

# CHALMERS



## Visualizing Database for Better User Interaction

Using data visualization approach to visualize Volvo Common Logging and MISP database

*Master of Science Thesis in the Programme Interaction Design*

ZHONGHAO CAI

Chalmers University of Technology  
University of Gothenburg  
Department of Computer Science and Engineering  
Göteborg, Sweden, October 2011

The Author grants to Chalmers University of Technology and University of Gothenburg the non-exclusive right to publish the Work electronically and in a non-commercial purpose make it accessible on the Internet.

The Author warrants that he/she is the author to the Work, and warrants that the Work does not contain text, pictures or other material that violates copyright law.

The Author shall, when transferring the rights of the Work to a third party (for example a publisher or a company), acknowledge the third party about this agreement. If the Author has signed a copyright agreement with a third party regarding the Work, the Author warrants hereby that he/she has obtained any necessary permission from this third party to let Chalmers University of Technology and University of Gothenburg store the Work electronically and make it accessible on the Internet.

Visualizing Database for Better User Interaction

Using data visualization approach to visualize Volvo Common Logging and MISP database

ZHONGHAO CAI

© ZHONGHAO CAI, October 2011.

Examiner: FANG CHEN

Chalmers University of Technology  
University of Gothenburg  
Department of Computer Science and Engineering  
SE-412 96 Göteborg  
Sweden  
Telephone + 46 (0)31-772 1000

Department of Computer Science and Engineering  
Göteborg, Sweden October 2011

## **Abstract**

Over the last decade, as the amount of information has exploded, data visualization has been used as one of the most important presentations for displaying information in an easy to understand way. From the traditional approaches like tables and bar charts to modern approach that a lot of graphics are used, the development of data visualization reflects the changing requirements that people perceive data. Nowadays, graphics are used widely in data visualization and the presentation of data is more elegant and descriptive than before.

This thesis work completed at the Integration team at Volvo Information Technology aims at visualizing complex database in graphics, and users who work with the database can view the connections between components clearly from the interfaces and benefit by its user-centered interactions and graphical presentations.

The current practices were analyzed, and the theories and methods were studied. The thesis work started with finding out stakeholders' requirements, followed by several designing iterations. During the process stakeholders actively participated in providing fruitful suggestions and helped to test the prototypes for better revisions. One project has been deployed successfully in the production, while the other is still in progress. Future work is needed, but the concept of the thesis work has been handed over to the team for later implementation.

**Key words:** Data Visualization, Prototype. Microsoft Expression Blend, Silverlight.

## Acknowledgements

First and foremost, I would like to express my deepest gratitude to my supervisors at Volvo IT, John Kaber and Patric Kronberg, who help me understand the thesis work and guide me along the process; Anders Wikström, who has patiently shared his experience and taught me the knowledge in the real-life environment. Without their support and encouragement, the thesis would not have been possible.

Many thanks to all the current members in the integration team at Volvo IT for their participation in helping me complete my work over the year: Mats Andersson for introducing me to the thesis work and explaining the complex tasks in an easy to understand way; Mats Erkenstam, Lars Grestam, Roger Andersson, Ing-Marie Perneblad for their active participation in the interviews and fruitful discussions.

I am grateful to Ramon Nurman and Hua Qu for their support and friendship. Sincere thanks to them for giving me advice on the way of thinking about my career development and also the essence of life. Thanks for them treating me like friends and cheering me up when I encountered problems during the work.

I would also like to acknowledge my examiner at Chalmers, Fang Chen for guiding me the matters I should pay attention to both in the working process and in the report. Her suggestions have allowed me to make the work more qualified in the scientific field. Thanks to all the people in the Department of Interaction Design for giving me a solid background in the field.

Finally, sincere thanks to my friends for all the support and care they have shown me. I would like to extend my thanks to friends all over the world. In particular, my better half, Datong Li deserves special thanks for comforting me when I was depressed and encouraging me to move on. Warmest wishes to Hongming Cai families who have made my stay in Sweden a pleasurable experience.

Zhonghao Cai

Oct, 2011

## Table of Contents

Abstract.....	1
Acknowledgements .....	2
1. Introduction .....	5
2. Background.....	7
2.1 Other Practices.....	7
2.1.1 Flickr Time .....	7
2.1.2 Fidge't Visualizer .....	7
2.2 Theories .....	8
2.2.1 Ambient Findability.....	8
2.2.2 The Laws of Simplicity .....	9
2.3 Methods .....	9
2.3.1 Agile Methodology.....	9
2.3.2 M-V-VM Pattern .....	11
2.3.3 User Requirement Analysis .....	12
3. Work Process.....	14
3.1 Preparation.....	14
3.1.1 Review of Internal Documents .....	14
3.1.2 Software.....	14
3.1.3 Subversion .....	15
3.2 Interviews .....	15
3.2.1 Identifying Stakeholders.....	15
3.2.2 Conducting Interviews.....	16
3.2.3 Analysis & Discussion .....	16
3.3 GUI Design.....	17
3.3.1 Preparation.....	17
3.3.2 Icons for Different Components .....	18
3.3.3 First Iteration for VCL Visualization .....	21
3.3.4 Second Iteration for VCL Visualization .....	25
3.3.5 Summary and Final Design for VCL visualization .....	27
3.3.6 First Iteration for MISP Visualization .....	29
3.3.7 Second Iteration for MISP Visualization.....	33
3.4 Implementation.....	35
3.4.1 Technical problems.....	35
3.4.2 VCL visualization in Silverlight.....	36
3.4.3 MISP Visualization in Silverlight .....	39
4. Results .....	40
4.1 Final Design for VCL Prototype in Silverlight .....	40
4.2 Final Design for MISP Prototype in Silverlight .....	41
5 Discussion.....	43
5.1 Method.....	43
5.2 Results .....	43
5.3 Limitations.....	44
5.4 Collaboration .....	44
5.5 Scientific Contribution .....	45
5.6 Future Work.....	45

References .....	47
Appendices .....	50
Appendix I. Outline of Stakeholder Interview .....	50
Appendix II. Stakeholders' Requirement Wish List .....	51
Appendix III. Sample MISP Visualization Use Case.....	53

## 1. Introduction

Volvo Information Technology (Volvo IT) has a large amount of data stored in a database called MISP (Managing Integration Solution Provisioning), and the data is stored with all the information regarding servers, names, users, etc. Authorized users are able to access the data through MISP, add new users to existing applications, or order new applications and configuration items (CI) [1]. Several roles used in MISP include

- Application Support
- Integration Specialists
- Infrastructure Specialists
- Configuration Administrators

Each role works in different fields and is responsible for one or more specific services. Application support works with applications and connections to the integration backbone by VCOM and WebSphere MQ (WMQ); integration specialists work with integration tools like message broker (WMB) and process servers; infrastructure specialists handles the integration infrastructure; while configuration administrators have high authority in MISP and work with infrastructure specialist as well.

Because different components are carried with different information and the amount of data is massive, it is difficult for users to see the complex relations among the components or the landscapes of a service or application. Therefore, the thesis work MISP Visualization is proposed with the purpose of visualizing the data in a clearer way, which allows users to be able to view the connections and dependencies of different components in MISP.

To get a better understanding of what functions stakeholders wish to have and how they would like them to be implemented, interviews were conducted at the first place and twenty-six requirements were gathered from one-to-one interviews, live meeting, and a sum-up from a workshop. After that, the requirements were analysed and grouped into different use cases. However, due to the shortage of time, not all the requirements were able to be implemented. In a meeting with one main stakeholder and two developers, two use cases were put in the highest priority and were decided to be carried out in the first stage. The reasons to have them implemented first were that they were the base for other use cases, and a large number of users could benefit after the implementation.

Ambient Findability and The Laws of Simplicity were the two main theoretical books for directing the design concept, and some other books and guidelines were also used as references. Iterative design methodology was applied during the design process in order to show the concept to the stakeholders, get feedback in time and refine the prototype accordingly to meet users' expectation. From flat pictures to interfaces with navigational functions, the stakeholders' requirements were taken into consideration in each designing iteration, while improvements and refinements were made to meet these requirements.

The designing process lasted for three months, and then the work moved onto another stage: implementation. This required interface designer and software developer to work together and convert the work to functionally usable. Problems occurred such as

different versions being incompatible and some controls not working as expected. Under such circumstances the developing team had to decide either to compromise by removing some of the functions in order to make it work, or to find solutions to solve the problems. It was hard to determine which functions deserved to live and which deserved to die, but the core concept remained the same, that was designing what stakeholders wanted and making them feel comfortable to use.



## 2. Background

### 2.1 Other Practices

Data visualization, as its name indicates, means visualizing data. According to the description of Vitaly Friedman, data visualization aims at seeking ways to present data effectively and convey the information clearly [2]. The traditional ways of visualizing data include line, bar, and pie charts and tables for example, which are functional but relatively boring. However, the modern approaches do not mean to make the data look fancy or sophisticated. Instead, Friedman suggested that the balance between functionality and aesthetic need to be considered carefully during the design. Graphical presentation is a means to deliver the information, and the main purpose shall not be misled by the complicated and fascinating design.

In Friedman's article, *Data Visualization: Modern Approaches*, he listed some examples showing what kinds of approach can be applied in graphical presentation [3]. As the thesis was about displaying data and its connections, I focused on reviewing the examples of these two kinds. Two typical examples of using modern approaches to present data are as follow.

#### 2.1.1 Flickr Time

Flickr Time is a tool of showing a collection of uploaded images in Flickr and presents them in the form of a clock with current time [4]. To get a large view of the image, simply move the mouse over it and the image will be displayed in a tooltip window as Figure 2-1 shows. If users want to see more information about the image, like who took it or other works from the same photographer, just click on the thumbnail and a new window will open, navigating to the normal flicker page that the image belongs to.

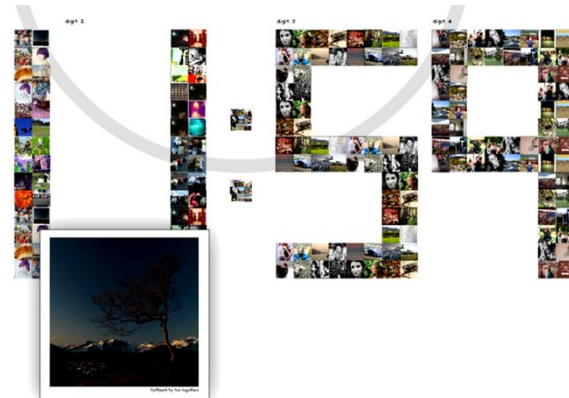


Figure 2-1. Screen shot of Flickr Time.

Resource from: <http://www.hottoast.org/convexstyle/flickertime/>

This approach of showing data is unique and the data, which in this case are images, are clear to the eye. Unlike the normal way of putting images in rows and columns or under separate albums, it finds a new and interesting approach of collecting the thumbnails into a form of clock and all the images are real-time. However, because the images are displayed randomly, users are not able to view which categories they belong to or choose their preferred category.

#### 2.1.2 Fidg't Visualizer

Fidg't Visualizer is an application to display connections between users and their friends. Users create magnet tags through the tags in Flickr and LastFM, and these magnet tags will generate a network of all the friends with the same tags, as the example demonstrates in Figure 2-2 on next page [5]. By using this application, users can see who share the same interest with them in music and photos, what the most popular

music type is and which kind of photos people prefer more. It helps discover the common interests in the same network, and allows users to learn the unfamiliar areas from each other as well.

Fidg't Visualizer displays data connections in a simple and direct way. The friends' list and their interests are visualized graphically. However, one concern about this type of visualization is that once the number of data increases, the network will become much more complex and the lines connecting between friends and the magnet tags will become closer and hard to see eventually.

One possible solution to that is adding a zooming slider to control the view size, so when the connections are complicated, users can zoom in and get a bigger view.

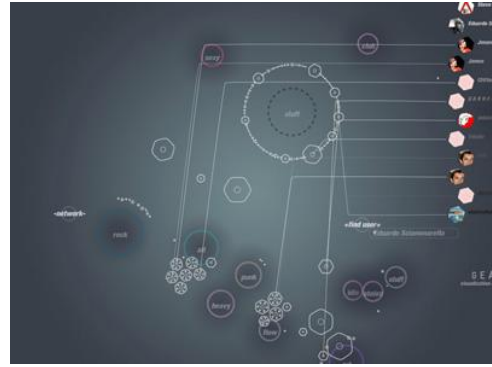


Figure 2-2. Screen shot of Fidg's Visualizer.  
Resource from <http://www.fidgt.com/visualize>

Based on the two examples above, it can be noticed that graphical approaches have been applied to data visualization in both websites and applications. They are becoming more and more popular, because users are no longer getting satisfied with only the pure data, but they also require it to be displayed in a good manner, in graphics, for example. Just like what Friedman described, 'data presentation can be beautiful, elegant and descriptive,' and the means of delivering the message to the end users do not have to be simple tables, pie charts and bar graphics that have been used for years [2]. Therefore, graphical approaches are adopted widely, seeking for creative ways to visualize data.

## 2.2 Theories

### 2.2.1 Ambient Findability

The book *Ambient Findability*, published in 2005 by Peter Morville, offers insights of how to help people find their way through an age of information overload and combine streams of complex information to filter out only the parts they want [6]. To be more specific, it explains some principles of information interaction that make designers start to think what people need and in which form they would like to perceive the information. To start the discussion, Peter first defines three terms that has been mentioned frequently: data, information, and knowledge. According to him, the definitions of the three terms are as follows (obtained from Chapter 3 in the book):

- Data. A string of identified but unevaluated symbols.
- Information. Evaluated, validated, or useful data.
- Knowledge. Information in the context of understanding.

These definitions reveal the interconnections between these terms and then the author brings up the concept that with effective and efficient communication the human society is tightly connected. As the information is ambient it is therefore important to choose appropriate ways for communicating the information in an understandable approach. Moreover, only delivering the information cannot satisfy people; a higher requirement comes out with the demand that the information carrier should be user friendly.

A critical element of designing in a user-centred method is making things easy to find, or in other words, enhance the usability in terms of the ease of finding what people need. Several approaches are highlighted by Peter, and to conclude from the book, ambient findability means to provide insights into human behavior and design from this perspective. Seeking for what people need most in the most appropriate way will make users feel comfortable when using the products or navigating on the interface. Therefore, this book is beneficial for the thesis work in the way of designing from users' perspective by applying these approaches.

### **2.2.2 The Laws of Simplicity**

A review of the book *The Laws of Simplicity* commented that 'simplicity is the soul of design while the author, John Maeda uses the concept of simplicity to get at the nature of human thought and perception' [7]. At the beginning of the book, John brings the argument that technology has made people's life full, but in some cases it is overloaded, therefore it is necessary to simplify the information by providing only the most important.

John introduces ten laws and three keys to achieve simplicity, of which the most helpful ones for the thesis work are described in the following:

- Reduce. The simplest way to achieve simplicity is through thoughtful reduction.
- Organize. Organization makes a system of many appear fewer.
- Learn. Knowledge makes everything simpler.
- Differences. Simplicity and complexity need each other.
- The one. Simplicity is about subtracting the obvious, and adding the meaningful.

It is normal that during the process people want to put the cool and fancy functions in the design but ignore the key fact of whether these functions are necessary. John describes a lot of scenarios that unused functions are a waste of space in design and money in production, which should be avoided through effective reduction and organization. However, not all could be removed; the most important ones should be kept and carefully designed in the way that suits users' perceptions and usage.

The laws are especially helpful for distinguishing which designing elements are meaningful and which are not necessary. Moreover, the principles discussed in the book direct the basic concept of the design to be simplified. Instead of making the interfaces look seemingly functional, it is more important to remain only what users need most to accomplish their tasks. In addition, with the guidelines of the first book, *Ambient Findability*, all the objects in the interfaces will be designed from the consideration of making users comfortable to use in an easy-to-navigate manner.

## **2.3 Methods**

### **2.3.1 Agile Methodology**

The traditional methodology, waterfall, is a linear and sequential approach that is largely based on predictability. When the users know what they want, the requirements are clear and rarely changing in the later phases, waterfall process can be an effective tool to guide the process. However, as Martin Fowler discussed in his article *The New Methodology*, the software development is unpredictable, and it is normal that the

requirements always change [8]. Since ‘changes are the norms’, how people cope with them appropriately and timely becomes important. In such cases, the late deployment in the waterfall process hides ‘lurking risks’ [9]. For instance, technological problems may occur that different parts cannot work together, or conceptual problems may happen that the final product is not what users exactly want, as users do not see anything real until the very end.

The development methodology the project followed is agile method, which is people-oriented and specification changes are allowed. The agile method works in cycles that the project is divided into tasks that need to be implemented in different iterations as Figure 2-3 demonstrates. The iterations are short time frames, each one lasting for one to four weeks. However, these iterations involve a full developing cycle that includes planning, requirement analysis, designing, etc. [10], and at the end of each iteration, users are able to see part of the functionality and give suggestions or demand any requirement changes. This timely feedback minimizes the risks and allows the plans to be adjusted in order to adapt to the changes. The agile method solves the problem of software development being hard to predict as Martin Fowler pointed out and helps reducing overheads from project management perspective as well.

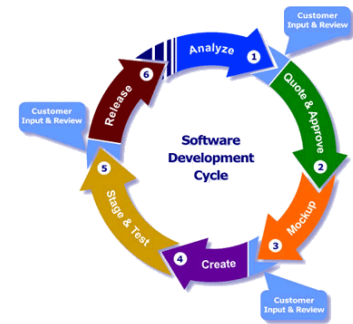


Figure 2-3. Model of agile iteration design methodology

By applying it to the data visualization project, several benefits were revealed. First of all, the schedule was flexible to change and easy to be adapted. When making schedules for the project, unexpected situations were not taken into consideration and once unpredictable problems occurred, the schedule was delayed. Under such circumstances, a longer period of time was needed to accomplish the tasks in that iteration. As the project was broken into several iterations at the beginning, it was possible to prolong one or more of the working duration, solve the problems in time and postpone the following iterations.

Secondly, changes that users wanted to make regarding the design or functionality and requirements that users came up during the process could be implemented in the next iterations. In the working cycle, after an iteration users were involved to provide direct feedback to the prototype. The input from users was beneficial for designers to make changes accordingly, and displayed to the users in the next iterations. Same to the changes, any new requirements that users wanted to add were feasible, because each iteration included a complete process and the new added functionality did not interfere the existing ones as long as they could technologically work together.

Thirdly, at the end of each iteration a self-adaptive process was conducted as the regular reviews of the project. Questions such as ‘what did we do well’, ‘what have we learned’, ‘what can we do better’, and ‘what puzzles us’ helped the developing team proceed in the next iterations and avoid making the same mistakes [8]. Moreover, the process was beneficial for self-improvement as well as keeping the project as scheduled, because, as discussed before, not all the situations were considered carefully and under control. In

such cases, every improvement within an iteration could help increase the maturity of the team, and reduce the chances of going to the wrong path.

### **2.3.2 M-V-VM Pattern**

The user interface development involves a series of activities: data, interaction design, visual design, connectivity, security, validation, unit testing, etc. [11]. However, interaction designers are focused more on the graphical and visual perspectives and do very less coding or data binding, while the software developers work with data and coding behind the interface. In order to let designers and developers work independently but keep the cooperation as a team as well, it is important that a platform is available to support all the functions. Microsoft was using Model-View-ViewModel (M-V-VM) pattern internally to develop Windows Presentation Foundation (WPF) applications such as Microsoft Expression Blend, which provides a platform to build user interfaces through a simple, independent and developer-approval pattern [11].

The M-V-VM design pattern is an architectural pattern used in software engineering, and it was introduced by John Gossman on his blog in 2005, who worked as a WPF and Silverlight architect at Microsoft [12]. It was developed in a way to help create user interface easier by its data binding infrastructure, which will be explained later in this part. The Model in M-V-VM represents the data that stores the real state content with codes or data. The View stands for all the elements displayed in the user interface, the buttons, graphics, windows, for instance. By creating the elements on the interface, XAML will be generated automatically behind, which means that designers do not need to care about the coding. Instead they can focus on the graphical parts.

The ViewModel, which can be interpreted as ‘the Model of the View’, serves in data binding function between the Model and the View. According to Gossman, the ‘ViewModel contains data-transformers’ that could transfer the View type to the Model type and allow the View to interact with the Model. In other words, the ViewModel is bound with properties on the View, and once the value of elements on the Model is changed, the data is exposed to the View through ViewModel. He then gave an example to further explain the role that ViewModel acts in the pattern. In the Model, there is some pre-defined data that has been bound with the View, but there are also certain types of data that cannot be controlled directly. If complex operations are needed to perform on the user interface or the View, some code is required for implementation. However, the non-predefined code may be too specific to put in the Model, and as a result, the ViewModel plays a role of connecting the controller between the View and the Model. Through this data binding infrastructure, the user interface can be bound with the code and interface designers do not need to take care of the code anymore.

In the data visualization project, the earlier prototypes were built in WPF, where the M-V-VM pattern played an important role for the designer and developers to work independently. The designer focused only on the graphics, the elements and how they could be placed in the interfaces without any disturbance of coding or programming. While the elements were created as the View, XAML was generated in the Model at the same time. For complex functionalities such as selecting, showing and hiding, it was controlled in the ViewModel by some more advanced controllers. All of these could be achieved in the Expression Blend under WPF application.

Later in the implementation phase, it turned out that prototypes built in WPF could not be converted into Visual Studio; the Silverlight projects were supported but not the WPF ones. Silverlight is another application platform developed by Microsoft, initially released in 2007, and it is powered by .NET framework that is compatible in different browsers [13]. The downside of using Silverlight instead of WPF to build projects in Expression Blend is that there is no animation-editing panel in Silverlight, and the only similar functionality is navigation. In such cases, the solutions were either to remove some of the animated functionalities or write code in the ViewModel to trigger animations in the View.

### **2.3.3 User Requirement Analysis**

To get full understanding of users' requirements and design new interfaces that fit their needs and benefits neatly, it is important to gather and analyse their requirements carefully in advance before moving to the designing phase. Several methods used to gather information and analyse user requirements include:

- Review of useful documents
- Stakeholder interview
- Use cases
- Prototyping
- Usability testing

#### **2.3.3.1 Review of Documents**

Review of useful documents in this case refers to studying the internal documents that provide information for the existing system, the specifications of the components, their functions, and how they work together. This process is helpful for knowing the background knowledge and forming a general picture of what kind of system the designer is dealing with. Though graphical designers do not have to know the technical theories like which component plays in what kind of role, it is better to have a rough idea in mind so when designing the icons or interfaces, the graphics can exactly reveal the most typical characteristics and let users distinguish them easily. For instance, VCOM products are used as basic transport and connectivity [14], and therefore the design of the VCOM icon should reflect the function and show these two characteristics.

#### **2.3.3.2 Stakeholder Interview**

Stakeholder interview not only means talking to the persons or getting feedbacks, but it also includes a series of preparation work such as stakeholder identification and interview structure. Reading background documentation also helps with the preparation. The stakeholders that participate in the interview need to be carefully selected, as they need to represent other users and they have the current working knowledge of the area to be discussed. The interview is better to be semi-structured with some prepared questions to start up the mind, and then the interviewers and interviewees continue with the subjects they are most interested. The process of conducting the interviews will be found in detail in the next section.

#### **2.3.3.3 Use Case**

Use cases give detailed realistic examples of how users may carry out their tasks in a specified context [15]. In other words, they are used to describe the steps that users need to perform in order to complete a task. By showing each step, designers can see if any unnecessary steps can be removed or if it is too complicated to use. The value of use

cases, as Craig Larman indicates in his book, is that they focus on the user goals and perspectives, and it is more user-centric [16]. However, in this thesis work, the use cases were firstly considered as a way to analyse users' requirements but then decided not to use, because users were very aware of what they wanted to be implemented.

#### 2.3.3.4 Prototyping

Instead of writing use cases, paper and computer prototypes saved more time and stakeholders could give instant feedbacks, which was more beneficial to the designers and project preparation; therefore, were widely used for the usability testing in the designing iterations. Prototyping is the process of building models by designers in order to let users manipulate. Prototyping comes in many forms, according to Professor Sauter, from low technique sketches such as paper screens to high technique systems like computer-aided software engineering, and generally speaking the low techniques are used in the initial stages to get user feedbacks and later on based on the comments designers develop operational prototypes during the design stage [17]. The advantages of using prototyping include having user involved, getting immediate feedback, fulfilling higher satisfaction, etc.

#### 2.3.3.5 Usability Testing

Usability testing is a technique used to ensure that users are able to perform intended tasks effectively and efficiently [18]. Participants include the designer and a representative user, or in this case stakeholder in each testing session, and the purpose is to find the difficulties users encounter while carrying out the tasks. In this thesis work, four usability testing sessions were conducted, and the comments regarding improvements were helpful for the final design of the projects. Examples of project prototypes and revisions after usability testing in each developing iteration will be included in later sections.

### **3. Work Process**

#### **3.1 Preparation**

##### **3.1.1 Review of Internal Documents**

Like mentioned above in the User Requirement Analysis section, review of internal documents helped with understanding the existing system, the specifications of the components, their functions, and how they work together. This process was beneficial for knowing the background knowledge effectively, and through reading these documents a general impression of the system was formed. The reviewed documents are stored in TeamPlace, which is a web-based solution that deals with the collaboration within a team, enabling team members to share documents, links, discussions and other forms of information regardless of time and location [19].

The Global Integration Infrastructure was the mostly used TeamPlace for searching document and retrieving information [20]. It includes the links to the archives that all the released notes and manuals of different components can be found. Moreover, it displays the architecture of these components in different layers, which makes the abstract concepts very easy to understand.

Another useful TeamPlace is ADT (Application Development Techniques), and it includes various materials regarding the methods and Volvo GUI design principles. Even from school methods like prototyping and usability testing have been lectured, it is good to read the principles from the company's perspective. The methods are basically the same, but the designing principle is detailed described. For instance, in the document named *Volvo Group Icon Visual Style* [21], it defines the general guidelines of creating new icons, their types (clickable or non-clickable), styles, colors, and sizes. In the tips on how to develop new icons, it has one tip of 'using existing icons as a foundation'. It is very helpful, because on one hand it indicates that existing icons are more familiar to the users and on the other hand it also reminds me to create something easy to be distinguished.

##### **3.1.2 Software**

The software applications used in this thesis project for designing the graphics and interfaces are Adobe Illustrator and Microsoft Expression Blend. The first one is well known among Interaction Design students as it is a professional tool to create vector graphics. One of the important parts in the project is to create icons for the components, and during the designing process Adobe Illustrator was widely used, with the reason of vector graphics being able to be scaled without degrading quality.

The latter one is not very popular at school, or at least new to me, and because the project is in close relation to the production, the finished designs should be able to be compelled in Microsoft Visual Studio for launching on the website. With this demand, Microsoft Expression Blend is the ideal software for user interface design tool. The benefits of using it are various. First of all, it is an interactive design tool, and it includes all the basic graphical functions for rapid prototyping and complex design, which gives designers 'What You See Is What You Get' instant feedback [22]. Secondly, it generates XAML (Extensible Application Markup Language) automatically, so



graphical designers do not need to bother about writing codes. Thirdly, as it is an application for web interface design, it can be compelled with Microsoft Visual Studio that is also a Microsoft developed product for interactive user experiences based on WPF (Windows Presentation Foundation) or Silverlight application, so software developers are able to implement the designs to the production [13].

Microsoft Expression Blend was not difficult to use in my first expression, as the interface was similar to the default graphical editing software Paint, with the graphical editing panel on the left and property panel on the right as Figure 3-1 presents. However,

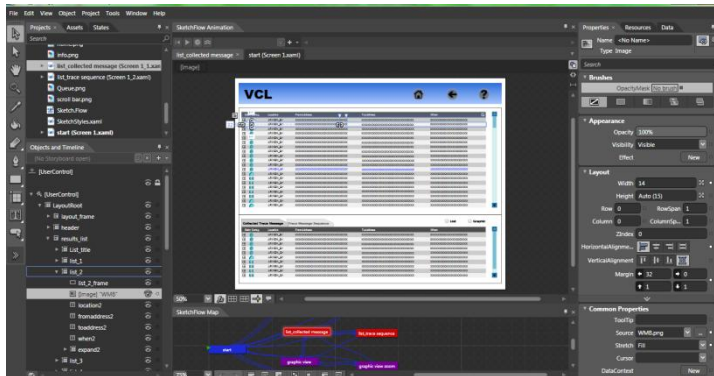


Figure 3-1. Screenshot of the interface in Expression Blend

it was not enough to use only the fundamental functions for high fidelity prototypes, and it was important to have something realistic for end users to try out and discover what was valuable and what was not useful [23]. From the graphical part it should have more pixel effects that are close to the final design, and from the interactive part the prototype

should be partly clickable. Therefore I started to dig deep to the functions that I was not familiar with, first by tutorial videos and then experienced designers' blogs and forums. It took some time to explore. While learning I created prototypes simultaneously, and looked for answers in online postings and tips when problems occurred. In the end, although I was still not an expert in this software, the final prototype was able to be compelled to Microsoft Visual Studio and to the production.

### 3.1.3 Subversion

Subversion is an open-source and centralized version control system that enables users to upload files and share them with other authorized users and they can also edit the files by checking them out to their local disk and update the revisions with a traceable history [24]. The biggest advantage of using it compared with other network disk is that subversion allows users to trace the changes in the 'show log' so users are able to see the changes in different revisions and compare them. Once the files are uploaded to subversion they are kept in a network disk that will not get lost if the computers crush. Every time if users need to edit anything they only need to check the files out and then check in again after editing. Updated files will replace the old ones automatically in the systems, and when other users check out the files, they are synchronized with the latest version. Because of these advantages and ease of use, it was used for sharing files and applying changes in the thesis work.

## 3.2 Interviews

### 3.2.1 Identifying Stakeholders

According to TrueSolution, identifying stakeholder is critical, and stakeholder refers to 'any person or organization that is actively involved in a project, or whose interests may be affected positively or negatively by the execution of the project' [25]. The topic of

the thesis is to visualize MISP database, and therefore the stakeholders include all the users that have impacts on MISP database and also the ones that might be impacted by.

The identified stakeholders in this project represented a diversity of users involved in the project, from product manager to maintenance manager, from application owners to security administrators. The selection was done by the supervisor in the company, who had rich knowledge of the stakeholders' influences and responsibilities on MISP. In total, five interviews in person with one sum up from a workshop of thirteen attendees and one phone interview were made. The interviewed stakeholders represented roles of two product managers, one maintenance manager, one security administrator, and one software architect. They not only have great involvement in MISP database, but they are also experienced users who are aware of the work process and its functions. Different from ordinary users, they are in a relatively higher position and have the right to look from a bigger perspective and have connections from both the application owners and developers. Therefore they were considered to represent the users' group as well as the main stakeholders' group.

### **3.2.2 Conducting Interviews**

The interviews were semi-structured, with some prepared questions to help understand the interviewee's role in the project and the tasks he/she needed to perform. The advantage of semi-structured interview, according to the in-class lecture of Method of Interaction Analysis, is that certain topics are guaranteed to be covered but the exact order or formulation depends on the actual situation. The prepared interview outline is attached in Appendix I. After asking some structured questions, I was clearer to the background information, and based on it I started to ask more specific questions such as their daily tasks on performing the system, the steps in order to accomplish a goal, the user experience for the current system and suggestions for improvements.

During the interviews, it was very beneficial to me that the stakeholders demonstrated their performing processes, because the processes clearly reflected the steps they took to accomplish familiar tasks. Some repeated actions indicated that unnecessary steps could be removed, and some steps could be improved to be simpler. Meanwhile, while performing the tasks from the beginning to the end, stakeholders could be able to point out more existing problems that might be ignored or forgotten if they only recalled from memories, and they could also show their wished improvements at some specific circumstances. Moreover, it was a great opportunity for me to observe the stakeholders' habits, understand their preferences, ask questions, and make clarification immediately. In average, each interview took one hour and five to ten requirements were collected, of which some were from a general perspective and some were from the demands of specific roles.

### **3.2.3 Analysis & Discussion**

After conducting interviews with the stakeholders, twenty-six requirements were gathered in total. It was impossible to implement all the requirements in the period of thesis work; therefore it was necessary to determine the priorities. A table was made to display the stakeholders' corresponding requirements in Appendix II. Similar requirements were coded in the same colour as the attachment shows. However, it was still not well organized for categorizing and analyzing these requirements.

Card sorting was applied in the process of grouping and defining which requirements belonged to the same category. By printing out each requirement on a piece of paper, requirements that were considered to be under the same category were put together. This process was finished together with the supervisor, as some technical information was involved and it was more accurate to work together with a person who has professional knowledge. Five categories were created as follows:

- General requirements, such as choosing between a list view and a graphic view, and being able to export data to local computer in Excel files.
- Visualization, such as visualizing the message path between components, and Volvo Common Logging.
- Application landscape, such as the objects belonging to the same application, and displaying the changes in the landscape and tracing them.
- Security, such as what approved accesses a user has, and who has authorized to a certain application.
- Application network, such as the connections between infrastructures and components.

A discussion was made to determine which tasks to be implemented in the first phase, and two stakeholders, a product manager and a global manager together with me and the supervisor attended the meeting. Two requirements were decided in a high priority in the thesis work: visualizing Volvo Common Logging (VCL), visualizing application landscape and visualizing application network in MISIP database. The reasons to choose them were that they were the basis for other requirements, and a lot of users could benefit from their implementation.

### 3.3 GUI Design

#### 3.3.1 Preparation

Before coming to the actual designing part, I started with analyzing the three requirements first. The purpose was to estimate how much work was needed and what characteristics were common and different among them. Visualizing VCL required a traceable message path to be displayed in graphics. In addition to the current list view, users would be able to view the components in sequences in a more direct graphical way. Application landscape required more work; the connections of one application to other components and (or) to other applications should be displayed. Compared with the message path in VCL, it was more like a collection of putting all the related paths together and showing the relations between each other. Application network would be a much bigger collection of having all the related applications organized in a network and displayed their connections. Figure 3-2 demonstrates these relations in a simple way.

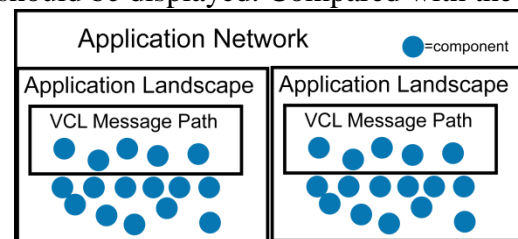


Figure 3-2. Relations of three use cases

Based on the complexity, I decided to start with VCL. It was easier because it did not involve too many items in the design, and the message path could be used in the other two cases as well. The style should be consistent in all the designs, and therefore it was very important that the message path was clear and simple so as to fit into complex relations.

As users were familiar with the current VCL system, it would be a challenge to change what they already got used to. It took time to adapt to changes, and I personally felt the same way too. For instance, I used to go to a website for weather forecast every day, but one day I found that a new version was launched to replace the one I had been checking for over a year as Figure 3-3 shows. My first reaction was that the old version worked fine, why did they change it?

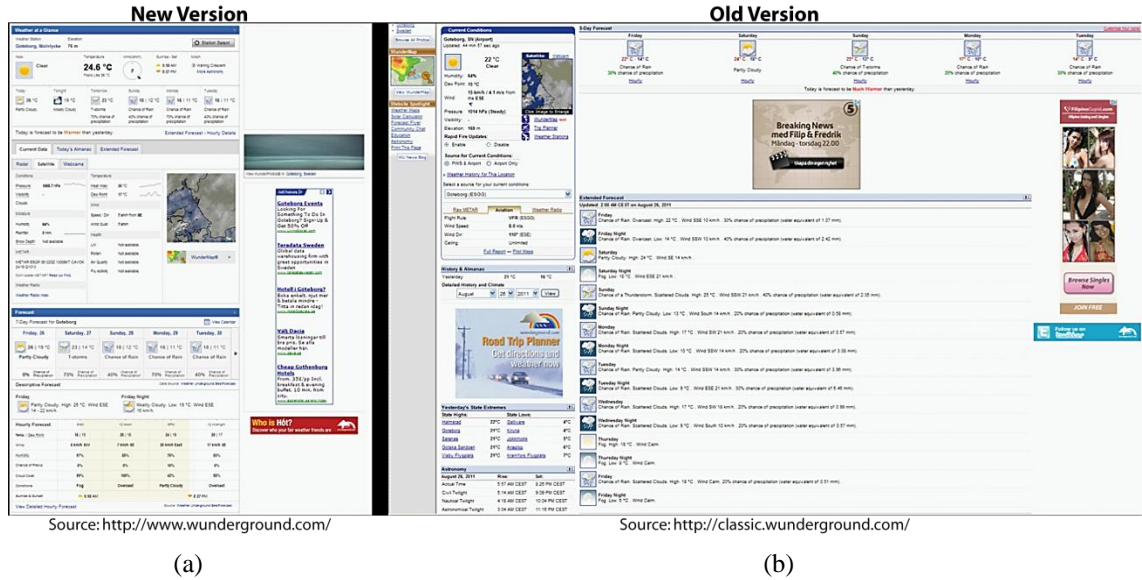


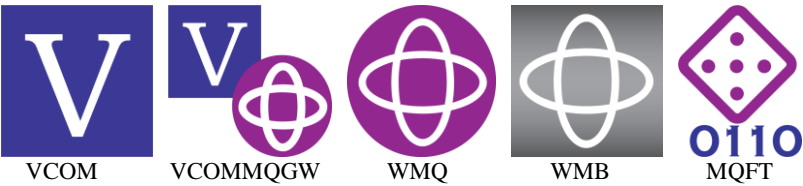
Figure 3-3. The comparison of the old (a) and new (b) versions of a weather forecast website

Then I spent some time looking through the new version and found that the new version organized information in a more reasonable way. For example, the new version put the most important data, current temperature, wind speed and direction, and the time for sunrise and sunset on the first row, following with the next five days' forecast on the next row. The sequence was logical for most users. The old version, in contrast, took up the whole page with two columns presenting the information, today's data on the left and the future's on the right. Since the right column was wider, it was more visible to the eye. The left column, however, presented the information that most users needed to see, so the focus of the interface was not arranged appropriately.

From the example above, it can be seen that users' habits are hard to change, and according to the book About Face 3, the authors also mention this by listing a design principle that 'significant changes must be significantly better' [26]. Therefore, it required cautious consideration about changing current features or adding new functions to the existing interfaces. My concept was that new graphical views could be added to the current VCL interface by changing as minimum as possible.

### 3.3.2 Icons for Different Components

One important task of the thesis work was to create icons for different components. The



current icons were shown in Figure 3-4. Because Volvo IT uses software from IBM, WMQ for instance,

Figure 3-4. Current icons for different components

and as a result, some icons were directly taken from IBM products while others were made up. The good part of these icons was that they could be distinguished by different colors and shapes even in a very small size, as Figure 3-5 shows a screenshot from the system. The image presents the actual size displayed in the screen, and users could easily tell what components are by looking at the icons.

However, to show a message path graphically, it would be better to have the icons in the same style and when putting them together it would look consistent. Moreover, the icons should have reflections on their functionalities. For instance, WMB works as a routing and transformation tool and the icon should have it reflected somehow.








+		WMB	ARWEN_B1
+		WMB	ARWEN_B1
+		WMQ	EOMER_B3
+		WMQ	EOMER_B3
+		WMQ	FRVU5PV1
+		VCOMMQGW	T214D103.ARWENB1
+		WMQ	ARWEN_B1

Figure 3-5. Screenshot from VCL system

For VCL visualization project, five main components were used in the system and needed to be presented graphically. Their basic functions, which are retrieved from Global Integration Infrastructure TeamPlace [20], are listed as follows:

- MQFT: a Volvo made file transfer tool, which uses WMQ (details can be found on the following section) as a transport layer to receive and send messages between applications or customer files. A scenario of MQFT is demonstrated in Figure 3-6.

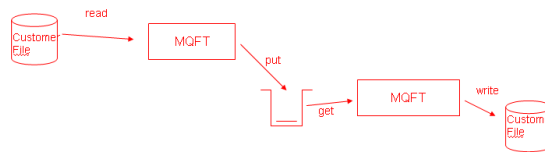


Figure 3-6. MQFT sends a file and another MQFT receives it.

- VCOM: a Volvo made data transfer tool, which allows processes to exchange data without concerning the different transferring technologies or operating systems. The scenario of VCOM is similar with MQFT, sending data from an application and then transferring it to others.
- VCOMMQGW: a gateway between VCOM and WMQ as the name indicates. It functions as a gate that allows information to be passed from one to another.
- WMB: an IBM product used for delivering transformation, routing and connectivity services. The functions include, for instance, transforming between different formats, and intelligently routing a message to a destination based on its content.
- WMQ: an IBM product used to exchange information on either an IBM platform or other platforms and integrate new applications to the existing systems, according to IBM official website.

Generally speaking, the functions and relations of these components can be grouped as Figure 3-7 shows on the next page.

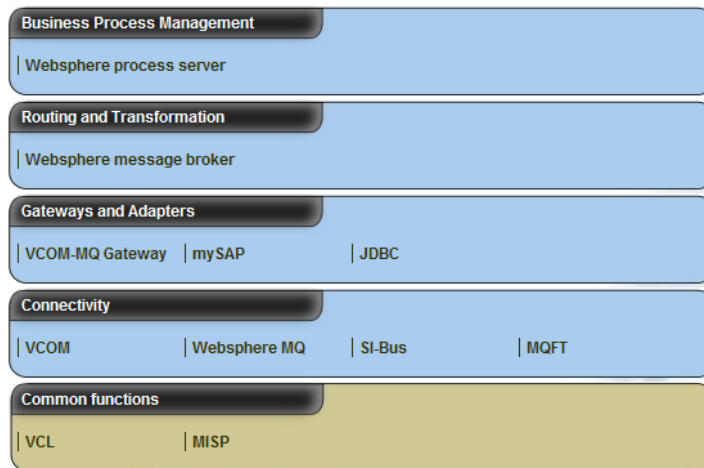


Figure 3-7. The Integration Infrastructure Reference Architecture  
Source from: <https://teamplace.volvoc.com/sites/global-integration/default.aspx>

Based on the description above, I analyzed these components' characteristics and functions, and endeavored to show them in a simply but direct way in the graphics. It was a tough task because the icons should not look too complicated, while many messages need to be delivered properly. Table 3-1 illustrates the final design compared with the current ones.

Table 3-1. Comparison of new designed icons and current icons

	New designed	Current
MQFT		
VCOM		
VCOMMQGW		
WMB		
WMQ		

above. Therefore the design for WMB was like a gear wheel, which implies that WMB offers more accurate engagements at work. Data is being processed by VCOM and WMQ. A vertical cyan cylinder represents databases, and the two small grey cylinders in WMB indicate the process of data being transferred from one database to another.



### 3.3.3 First Iteration for VCL Visualization

For problem determination, a tool called VCL is used in order to track messages [21]. In the current logging system, the collected messages and their tracing sequences are displayed in tables as Figure 3-8 shows. The task was to visualize message path in

The screenshot shows the 'Message Trace Result' window. It contains two main sections: 'Collected Trace Messages' and 'Trace Message Sequence'. Both sections display a table with columns: ID, Details, Main Comp., Location, From Address, To Address, and When.

**Collected Trace Messages Table:**

ID	Details	Main Comp.	Location	From Address	To Address	When
1926458822	WMB	ARWEN_B1	VTCVSS.CDB.CUST.LOW.IN	VTCVSS.CDB.CUST.LOW.IN	VTCVSS.CDB.CUST.LOW.IN	2011-01-31 10:55:00.000 +01:00
1926458824	WMB	ARWEN_B1	VTCVSS.CDB.CUST.LOW.IN	VTCVSS.CDB.CUST.LOW.IN	VTCVSS.CDB.CUST.LOW.IN	2011-01-31 10:55:00.000 +01:00
1926458826	WMB	ARWEN_B1	VTCVSS.CDB.CUST.LOW.IN	VTCVSS.CDB.CUST.LOW.IN	VTCVSS.CDB.CUST.LOW.IN	2011-01-31 10:55:00.000 +01:00
1926458828	VCOMMQOW	VSSPD104.ARWENB1	VSSPD104	VSSPD104	VTCVSS.CDB.LOW	2011-01-31 10:55:00.860 +01:00
1926458830	WMQ	ARWEN_B1	VTCVSS.CDB.CUST.LOW.IN	VTCVSS.CDB.CUST.LOW.IN	VTCVSS.CDB.CUST.LOW.IN	2011-01-31 10:55:00.860 +01:00
1926458832	WMQ	ARWEN_B1	VTCVSS.CDB.CUST.LOW.IN	VTCVSS.CDB.CUST.LOW.IN	VTCVSS.CDB.CUST.LOW.IN	2011-01-31 10:55:00.860 +01:00
1926458834	WMQ	ARWEN_B1	VTCVSS.CDB.CUST.LOW.IN	VTCVSS.CDB.CUST.LOW.IN	VTCVSS.CDB.CUST.LOW.IN	2011-01-31 10:55:00.860 +01:00
1926458836	WMQ	ARWEN_B1	VTCVSS.CDB.CUST.LOW.IN	VTCVSS.CDB.CUST.LOW.IN	VTCVSS.CDB.CUST.LOW.IN	2011-01-31 10:55:00.860 +01:00

**Trace Message Sequence Table:**

ID	Details	Main Comp.	Location	From Address	To Address	When
1926458834	WMQ	ARWEN_B1	VTCVSS.CDB.CUST.LOW.IN	VTCVSS.CDB.CUST.LOW.IN	VTCVSS.CDB.CUST.LOW.IN	2011-01-31 10:55:00.873 +01:00
1926458836	WMQ	ARWEN_B1	VTCVSS.CDB.CUST.LOW.IN	VTCVSS.CDB.CUST.LOW.IN	VTCVSS.CDB.CUST.LOW.IN	2011-01-31 10:55:00.873 +01:00

Below the tables, there is a section for 'Unsorted messages' which also displays a table with similar columns and data.

Figure 3-8. A screenshot from VCL

graphics. As the new icons were ready, the next step was to find a way to put all the information together. My direct reaction was to display components in a row. Then a question came up: how to deal with the texts? It was apparently not possible to fit everything in a small size and keep them readable; therefore, a solution needed to be carried out to make the texts clear to the eye.

#### 3.3.3.1 First Prototype

I firstly did simple sketches on paper to show the concept before drawing any graphics. It was beneficial at the initial stage, as discussed in User Requirement section; the low tech prototype saved more time and still could get some instant feedback. Viewed from

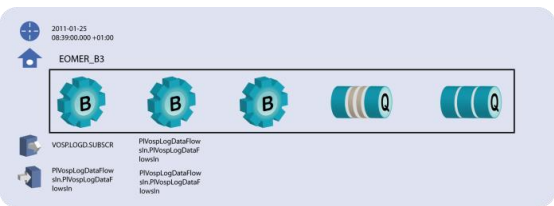


Figure 3-9. First VCL prototype

the sketches, it seemed possible to put the icons in a row, while the texts were distributed both above and below the icons. Then I tested the idea in Illustrator, which is shown in Figure 3-9. The intention was that the texts were not displayed initially; only the collected components were shown. If users wanted to know more, they could click on the four buttons representing Time, Location, From Address and To Address on the left column for detailed information.

Although the message path was now displayed in graphics, it had many downsides. First of all, it did not make much sense to users if they only saw the components; there were millions of the same components that belong to different applications. Secondly, there were too many clicks before users could see all the texts, which in most of the cases were useful and important to see. Thirdly, the graphics in the middle were separated from the texts, but they should be related together. Based on the analysis, the first prototype was rejected.

### 3.3.3.2 Second Prototype

Instead of creating the second prototype in a completely different concept, I checked the first one and sought other possibilities to have the texts displayed in a more clear way while remain the graphics in a row. I then remembered an online store I used to visit, and their method of presenting items on one page was very inspiring as Figure 3-10 illustrates. The red circle highlights the zooming function, which allows users to choose the image size they prefer to view on a page, three rows, two rows or only one row. It provides options for users to view based on their preferences, which highly reflects the user-centered approach. If there was a zooming function for the message path, would the unreadable small texts be possible to be solved? With that question in mind, I started my second prototype.

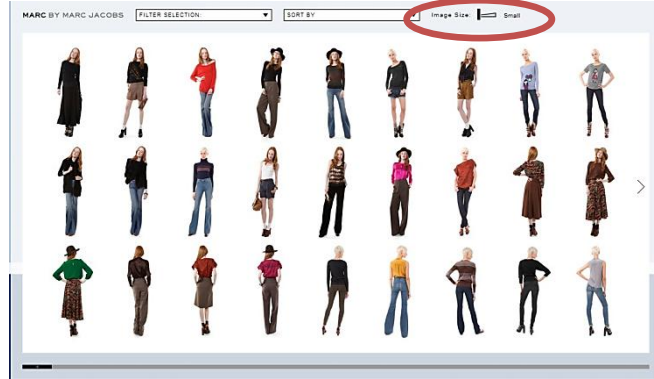


Figure 3-10. Screenshot from an online store using slider to enhance user interaction.

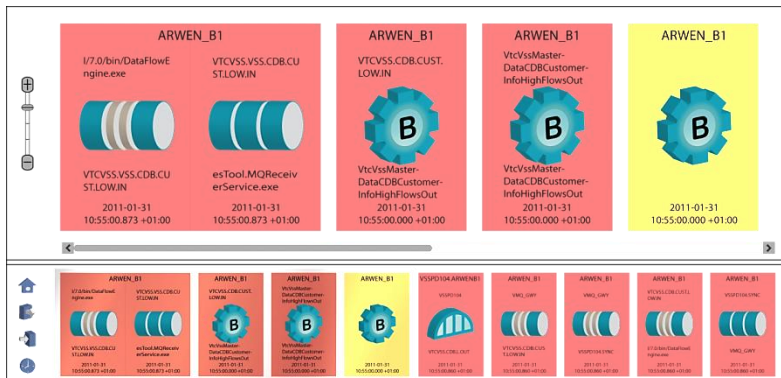
Source from Marc by Marc Jacobs.

<http://www.marcjacobs.com/marc-by-marc-jacobs/>

The basic layout of putting components together was similar with the first one, but the major changes were that a slider was added to the interface so that users were able to zoom in for details or zoom out for a complete view. The interfaces of the second prototype are presented in Figure 3-11. The reason that the interface was split in two rows was that in the lower part, users could see the complete message path.



(a) Initial state



(b) Magnified view

Figure 3-11. Interfaces of second prototype built in Illustrator

While in the upper section, a slider was available on the left to adjust the size. The first picture demonstrates the initial state. Once the users decided to zoom in, they could move the zoom bar to get up to four times bigger images as figure 3-11 (b) shows. Apparently only a part of the message path could be fit into the upper section after zoomed in; therefore the horizontal scrollbar was functional.

Moreover, in the lower part a grey shade was covered on the first five



components, corresponding to the upper area and indicating that they were currently visible. Users could also drag the grey shade left and right, and see the next components. Both the shade and scroll bar worked as a navigation tool, and users could choose either one according to their preferences.

A few things also needed to be pointed out in this prototype. First of all, the first two components were placed tightly next to each other, while the others were in separate frames. This was because sometimes messages could not be successfully traced. Referring back to the screenshot from the actual system in Figure 9, it had only two traceable messages and others unsorted. The connected and unconnected frames revealed this information in the graphics. Furthermore, the different background meant that another message path also passed through the same component. For instance, mostly the message paths are in a row, from a starting point to a destination. However, there are exceptions that more than two paths pass through the same component. Figure 3-12 gives a sample of the different types of paths.

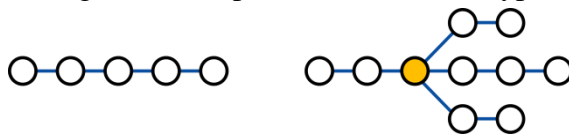


Figure 3-12. Sample of different types of paths

The left image illustrates the most common path, while the right one also exists in the system. Therefore, another background color could make users be aware that this particular component can also be found in other message paths.

Compared with the first prototype, some improvements could be noticed in the second prototype. First of all, the graphics was presented in a row, which was correspondent with the concept of showing the message in a graphical flow, while the details were included in the grids as individual components, so there was no need to click on icons for details. Second, the texts could be read by dragging the slider for zoom-in, and if users want to see the whole path they did not need to zoom out, as the complete path was always available on the lower part. Third, with different colored background, users could see if it was a linear path or a complex one. However, the graphics took large space, and the idea of putting traceable components together and unsorted ones separate did not seem very clearly. Revisions of the prototype were still needed, but the slider for large view was considered a solution to deal with the texts.

#### 3.3.3.3. Third Prototype

After the analysis of pros and cons for the second prototype, I decided to make some improvements by removing the constraints such as large space required. How to show the message path with zoom functions while the graphics still remained in one row? Then an idea came to mind, which was the 'mouse over' function. If users placed the cursor on top of one component, it could enlarged automatically; while users removed the cursor to other places, the enlarged component changed back to the original size. This solution did not require users to click the mouse; only by moving the mouse around they could get details in a bigger view. To make the prototype work with the mouse over function, flat pictures could not fully reflect the advantages and disadvantages. Therefore Expression Studio was used to create a simple prototype that allowed users to interact with.

Although in the first iteration of design the primary purpose was to choose the best design concept that could fulfill user needs, it was necessary to let users partly interact

with the interface so it gave a more real design of what users would use in the future, and also could get more thorough feedback on what was good and what needed improvement. In such case, the prototype built in Expression Blend was not for implementation, but only for presenting the design and generating feedback. With the function of importing files from Adobe Illustrator, the software allows users to directly import the graphics designed in Illustrator to Blend as a WPF application. The drawback of it is that even the imported files are still grouped the same way as they are in Illustrator, the items are transferred into paths, and a result of it is that a square can be stored as more than ten paths, which is difficult for further modification. However, with only showing the mouse over function to each component, it did not need much change to each item. By adding the animation to the whole group of paths, the mouse over could be achieved. Figure 3-13 demonstrates the different screens: Figure (a) is the initial path, and Figure (b) is when the cursor is placed on top of the second component.

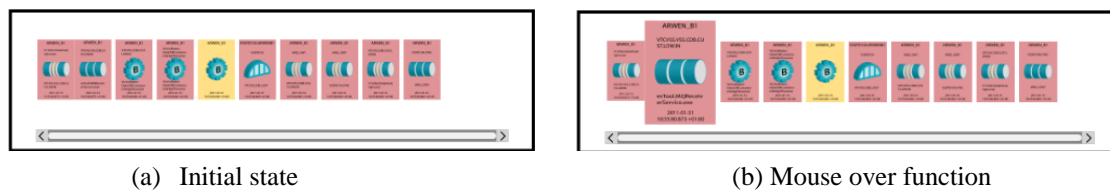


Figure 3-13. Screenshots of third prototype from Expression Blend application

This prototype did not need two rows to present the message path, and users were able to view the complete path directly while get an enlarged image of a particular component in detail. But an obvious constraint was that there was no way to have two or more components in enlarged views together. If users wanted to see details of the whole path, they had to move the mouse on top of every component and had a good memory; otherwise they had to move around many times until they found what they wanted to see.

#### 3.3.3.4 Feedback Interview

Although each prototype had its advantages and disadvantages, it was important to know what stakeholders thought useful and what they wanted to improve, and some more requirements may be acquired based on the current possible designs. Therefore a stakeholder interview for generating feedbacks and suggestions was conducted after three prototypes were created. Participated stakeholders included a global manager, a product manager, an application architect and they all share the rich knowledge of the thesis work and its impact to the whole users.

The first thing to discuss was the new icons. Comparing with the original ones, these icons were considered more consistent, and both stakeholders participating in the interviews had no difficulties identify the components that the icons presented. They did not give any comments on improvements, so the icons were kept the ways they were.

Next I discussed with stakeholders about the prototype individually and the feedback I got included in the following aspects: the layout of message path, and the functions that might be useful. They were satisfied with the way that the components were displayed in a row, and the zoom function was especially helpful for those who had small screens, or needed to read the texts. Although the mouse over offered a similar function, stakeholders were more comfortable with the whole path enlarged.

Furthermore, by color coding the different components users could be able to see the differences between components. However, one thing that could cause confusion was that users may understand the colors in a different way. They might think components in the same locations were grouped in the same color, for instance. Therefore the usage of background color should be carefully reconsidered.

### 3.3.4 Second Iteration for VCL Visualization

The overall feedback from the interview was positive, and they were satisfied with the layout that the components were shown in rows to represent the concept of message flow. The most difficult part was still the display of the text information in a readable and clear way. To seek for more inspirations, I checked a website named ‘Visual Complexity’ where many complex data has been visualized graphically [27]. There are many examples of data visualization, and some are quite revolutionary. However, they cannot be fit into the VCL context, as the VCL interfaces need to be formal, and corresponding to other systems. Therefore I decided to focus on presenting what users needed most in a plain and clear way rather than creating fancy graphics.

What was the most frequently used method to deal with text information? One possible answer would be tables. Tables have been used in presenting large amount of information for many years, and although it is not a modern method for data visualization, it is still considered one of the most effective ways. In the current system the result was shown in a big table, which can be reflected back in Figure 3-8. If the table could be somehow embedded with the graphical path, then the texts could be readable in a clearer way. With this thought in mind, I did some simple sketches on paper, trying to find a good combination. There seemed to have two ways of presenting tables in the graphics, one was to place the components horizontally and the other was to place vertically. I started with the horizontal one for the fourth prototype and the vertical one for the fifth.

#### 3.3.4.1 Forth Prototype

The idea of this prototype, as described above, was focused on showing information in an easy-to-read approach. Meanwhile, the mode of graphical path that had been created in the previous prototypes would be kept with slight modifications so as to adapt to the new design. Furthermore, the table would be embedded with user interactions so that the traditional method would fit in the overall design neatly. To discover the pros and cons of the interactions, Expression Blend was used for the development, which partly allowed me and users to interact with the prototype.

Figure 3-14 illustrates the basic idea of how the graphical path would look like. In the initial state the graphics were displayed in a row, representing the message flowing from the start to its destination. When users placed

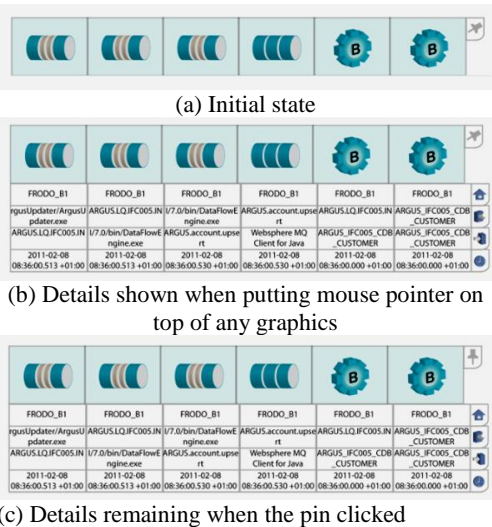


Figure 3-14. Screenshots of fourth prototype built in Expression Blend

mouse pointer on top of any graphics, the details were shown in a table, with each column corresponding to the component above. If the pointer was removed from the graphical area, the table would be folded with only graphics remaining in the interface. For some users they would need to work with the texts in order to detect the path or solve problems. In such cases, having to keep pointer within the graphical area could be very annoying. To avoid this situation, I added a ‘pin’ button on the right top corner of the interface, which allowed the tables to remain when the mouse was moved away from the graphical area. The purpose of this ‘pin’ function was that if users only needed to get an overall idea of the message flow, then they could control whether the table was displayed or not; for those who needed to deal with the details, it was convenient to have them always visible.

The primary benefit of this prototype was that it could fulfill the demands of all the users, needing or not needing to read the details in a plain and easy way. With the combination of table and graphics, the size of the texts was big enough to read, while the message flow was still clearly presented. Moreover, the pin function offered users an option of dealing with the table by clicking or not clicking on the button. On the other hand, in its initial state without having mouse over, only the components were shown. As discussed in previous prototypes, components were stored in different locations and only seeing the type of component did not give much information. In addition, although the tables were a means of visualizing data, new approaches could be found for better replacement. Therefore, this prototype, together with the next one provided different concepts for stakeholders to choose.

### 3.3.4.2 Fifth Prototype

This prototype, similar with the fourth one, was designed in order to have clear data visualization in a table. The layout of the components, unlike the previous one, was displayed vertically. However, instead of the mouse over and pin function, in this case I

used zoom function together with expanding command. The purpose was to try different types of interactions and seek for the most satisfied one.



(a) Initial state

	FIG000_01	ARGUS/IQ/PC005.IN	2011-02-08 08:36:00.513 +01:00
	FIG000_01	ARGUS/IQ/PC005.IN	2011-02-08 08:36:00.513 +01:00
	FIG000_01	I7.0/Doc/CatalFlowEngine.exe	2011-02-08 08:36:00.530 +01:00
	FIG000_01	ARGUS/accountexpert	2011-02-08 08:36:00.530 +01:00
	FIG000_01	ARGUS/accountexpert	2011-02-08 08:36:00.530 +01:00
	FIG000_01	ARGUS/IQ/PC005.IN	2011-02-08 08:36:00.530 +01:00
	FIG000_01	ARGUS_IPC005_CDB_CUSTOMER	2011-02-08 08:36:00.530 +01:00
	FIG000_01	ARGUS_IPC005_CDB_CUSTOMER	2011-02-08 08:36:00.530 +01:00

(b) Details shown by clicking on the expand arrow

Figure 3-15. Screenshots of fifth prototype built in Expression Blend

The interfaces of this prototype were illustrated in Figure 3-15 on the next page. The first one presented the initial state when users got a sorted message sequence. On the right column in the center where the area was highlighted by a red circle, an expand arrow indicated that more information could be found when the arrow was expanded. Once clicking on the arrow, a table was revealed with all the details shown in each column as the second image demonstrated. The magnified glass on top of the graphics was the zoom button, which could enable the whole interface to be enlarged to 4 times bigger. Different from the slider in the second prototype, the zoomed in size was

predefined and users could not control the size they wanted the interface to be. Furthermore, the 'from address' and 'to address' were combined in the same column in two rows, the top one referring to the 'from address' and the bottom one referring to the 'to address'. Because the addresses were usually long and if they were placed in parallel, the table could be very wide.

Both the fourth and fifth prototypes were built in Expression Blend as WPF applications, which supported all the interactions mentioned above (mouse over, pin, expand, and zoom) for stakeholders to navigate. From my experience of using Expression Blend to build prototype, the most beneficial thing was that the intended interactions could be practically tried out, which could better simulate the actual working situation and get more real feedback compared with the flat pictures in Illustrator.

#### 3.3.4.3 Feedback Interview

After finishing two more prototypes, I started another stakeholder interview for feedbacks on VCL visualization. It was important to know how stakeholders felt about using tables for visualization and which type of interactions they preferred before more time was spent on similar solutions. Two stakeholders, the global manager and a product manager who also joined the previous iteration participated in the interview, and they offered effective responses based on their experiences.

The first issue being discussed was the manner of displaying the components. Both stakeholders considered the horizontal way was better in terms of reading and conventional habits. The next one, regarding the interactions of showing data, they thought that the zoom function was functional as people had the feeling of controlling it themselves. Moreover, as the screen size varied, in a small screen it might be hard to read texts even in tables. The good part in the fourth prototype was that with the pin button people could save the current page. That could be used in a different way, snapping the current screen for instance. Compared with tables displayed while mouse over or clicked by the expand button, they still thought that having all the components and information shown together was easier and more direct.

During the interview I asked if all the information was equally important, and my intention was that if not all the four categories had to be shown to the users, then some texts could be hidden in some way and when users needed, they could click, a button for example, to display all the information. The response to the question was that some users might just get an overview of the path so they would just look at the time, but for application or system administrators they would read the details, especially when problems occurred. Therefore all the information had to be presented directly.

A function that could be added to the graphical view was to visualize the paths that went from one component to different destinations. It was not enough to just point out the particular component, and users could benefit more if the other paths could somehow be visualized too.

#### 3.3.5 Summary and Final Design for VCL visualization

Based on the feedbacks and suggestions, some functions should be kept and improvements should be made so they could fit together. The most favorable way to

present the message path was placing components horizontally, with texts presented in a readable way. The interactions should be simple; unnecessary clicks or navigations should be removed. To conclude from the analysis, the layout of the message path was in a row, with the texts around the component icon, while the appropriate function for better visual effect was the zooming slider, which allowed users to adjust the size of the path for larger views.

The next step was choosing where to put the graphical path. Currently the on the result page, 25 results were shown at one time, leaving the bottom of the page blank, as shown in Figure 3-16 (a) on the left. If that space could be used to display the message

Index	Source	Destination	Subject	Time	Size	Status
1	192.168.1.1	192.168.1.2	Test Message 1	2013-10-10 10:10:10	1024	Success
2	192.168.1.1	192.168.1.2	Test Message 2	2013-10-10 10:10:11	1024	Success
3	192.168.1.1	192.168.1.2	Test Message 3	2013-10-10 10:10:12	1024	Success
4	192.168.1.1	192.168.1.2	Test Message 4	2013-10-10 10:10:13	1024	Success
5	192.168.1.1	192.168.1.2	Test Message 5	2013-10-10 10:10:14	1024	Success
6	192.168.1.1	192.168.1.2	Test Message 6	2013-10-10 10:10:15	1024	Success
7	192.168.1.1	192.168.1.2	Test Message 7	2013-10-10 10:10:16	1024	Success
8	192.168.1.1	192.168.1.2	Test Message 8	2013-10-10 10:10:17	1024	Success
9	192.168.1.1	192.168.1.2	Test Message 9	2013-10-10 10:10:18	1024	Success
10	192.168.1.1	192.168.1.2	Test Message 10	2013-10-10 10:10:19	1024	Success
11	192.168.1.1	192.168.1.2	Test Message 11	2013-10-10 10:10:20	1024	Success
12	192.168.1.1	192.168.1.2	Test Message 12	2013-10-10 10:10:21	1024	Success
13	192.168.1.1	192.168.1.2	Test Message 13	2013-10-10 10:10:22	1024	Success
14	192.168.1.1	192.168.1.2	Test Message 14	2013-10-10 10:10:23	1024	Success
15	192.168.1.1	192.168.1.2	Test Message 15	2013-10-10 10:10:24	1024	Success
16	192.168.1.1	192.168.1.2	Test Message 16	2013-10-10 10:10:25	1024	Success
17	192.168.1.1	192.168.1.2	Test Message 17	2013-10-10 10:10:26	1024	Success
18	192.168.1.1	192.168.1.2	Test Message 18	2013-10-10 10:10:27	1024	Success
19	192.168.1.1	192.168.1.2	Test Message 19	2013-10-10 10:10:28	1024	Success
20	192.168.1.1	192.168.1.2	Test Message 20	2013-10-10 10:10:29	1024	Success
21	192.168.1.1	192.168.1.2	Test Message 21	2013-10-10 10:10:30	1024	Success
22	192.168.1.1	192.168.1.2	Test Message 22	2013-10-10 10:10:31	1024	Success
23	192.168.1.1	192.168.1.2	Test Message 23	2013-10-10 10:10:32	1024	Success
24	192.168.1.1	192.168.1.2	Test Message 24	2013-10-10 10:10:33	1024	Success
25	192.168.1.1	192.168.1.2	Test Message 25	2013-10-10 10:10:34	1024	Success

(a) Search result page

Index	Source	Destination	Subject	Time	Size	Status
1	192.168.1.1	192.168.1.2	Test Message 1	2013-10-10 10:10:10	1024	Success
2	192.168.1.1	192.168.1.2	Test Message 2	2013-10-10 10:10:11	1024	Success
3	192.168.1.1	192.168.1.2	Test Message 3	2013-10-10 10:10:12	1024	Success
4	192.168.1.1	192.168.1.2	Test Message 4	2013-10-10 10:10:13	1024	Success
5	192.168.1.1	192.168.1.2	Test Message 5	2013-10-10 10:10:14	1024	Success
6	192.168.1.1	192.168.1.2	Test Message 6	2013-10-10 10:10:15	1024	Success
7	192.168.1.1	192.168.1.2	Test Message 7	2013-10-10 10:10:16	1024	Success
8	192.168.1.1	192.168.1.2	Test Message 8	2013-10-10 10:10:17	1024	Success
9	192.168.1.1	192.168.1.2	Test Message 9	2013-10-10 10:10:18	1024	Success
10	192.168.1.1	192.168.1.2	Test Message 10	2013-10-10 10:10:19	1024	Success
11	192.168.1.1	192.168.1.2	Test Message 11	2013-10-10 10:10:20	1024	Success
12	192.168.1.1	192.168.1.2	Test Message 12	2013-10-10 10:10:21	1024	Success
13	192.168.1.1	192.168.1.2	Test Message 13	2013-10-10 10:10:22	1024	Success
14	192.168.1.1	192.168.1.2	Test Message 14	2013-10-10 10:10:23	1024	Success
15	192.168.1.1	192.168.1.2	Test Message 15	2013-10-10 10:10:24	1024	Success
16	192.168.1.1	192.168.1.2	Test Message 16	2013-10-10 10:10:25	1024	Success
17	192.168.1.1	192.168.1.2	Test Message 17	2013-10-10 10:10:26	1024	Success
18	192.168.1.1	192.168.1.2	Test Message 18	2013-10-10 10:10:27	1024	Success
19	192.168.1.1	192.168.1.2	Test Message 19	2013-10-10 10:10:28	1024	Success
20	192.168.1.1	192.168.1.2	Test Message 20	2013-10-10 10:10:29	1024	Success
21	192.168.1.1	192.168.1.2	Test Message 21	2013-10-10 10:10:30	1024	Success
22	192.168.1.1	192.168.1.2	Test Message 22	2013-10-10 10:10:31	1024	Success
23	192.168.1.1	192.168.1.2	Test Message 23	2013-10-10 10:10:32	1024	Success
24	192.168.1.1	192.168.1.2	Test Message 24	2013-10-10 10:10:33	1024	Success
25	192.168.1.1	192.168.1.2	Test Message 25	2013-10-10 10:10:34	1024	Success

(b) Collected message page

Figure 3-15. Screenshot of current search result and collected message pages

Resource from: <http://atlas-a1.it.volvo.net/vcl/Default.aspx>

sequence, then users would not need to be directed to a new page for a complete collected message as the right image illustrates. Tabs were a good choice for moving a whole page to a limited space. With two tabs, one for collected message and the other for traceable or unsorted message sequence, users could be able to see all the information on the result page, no need to go back and forth when trying to select different components. For switching between list view and graphical view, radio button could be used, which was different from having tabs to switch between two list views. In addition, a slider was automatically added when the graphical view was selected, so the graphical path could be zoomed in and out. The final design of VCL interfaces were presented in Figure 3-17.

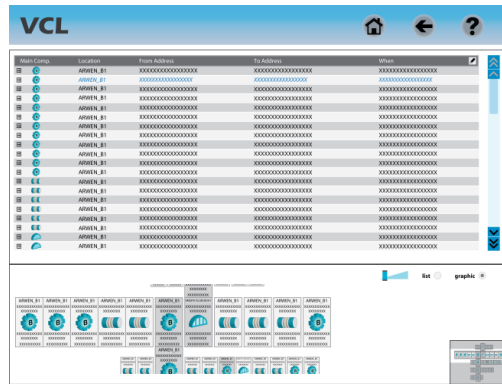
Index	Source	Destination	Subject	Time	Size	Status
1	192.168.1.1	192.168.1.2	Test Message 1	2013-10-10 10:10:10	1024	Success
2	192.168.1.1	192.168.1.2	Test Message 2	2013-10-10 10:10:11	1024	Success
3	192.168.1.1	192.168.1.2	Test Message 3	2013-10-10 10:10:12	1024	Success
4	192.168.1.1	192.168.1.2	Test Message 4	2013-10-10 10:10:13	1024	Success
5	192.168.1.1	192.168.1.2	Test Message 5	2013-10-10 10:10:14	1024	Success
6	192.168.1.1	192.168.1.2	Test Message 6	2013-10-10 10:10:15	1024	Success
7	192.168.1.1	192.168.1.2	Test Message 7	2013-10-10 10:10:16	1024	Success
8	192.168.1.1	192.168.1.2	Test Message 8	2013-10-10 10:10:17	1024	Success
9	192.168.1.1	192.168.1.2	Test Message 9	2013-10-10 10:10:18	1024	Success
10	192.168.1.1	192.168.1.2	Test Message 10	2013-10-10 10:10:19	1024	Success
11	192.168.1.1	192.168.1.2	Test Message 11	2013-10-10 10:10:20	1024	Success
12	192.168.1.1	192.168.1.2	Test Message 12	2013-10-10 10:10:21	1024	Success
13	192.168.1.1	192.168.1.2	Test Message 13	2013-10-10 10:10:22	1024	Success
14	192.168.1.1	192.168.1.2	Test Message 14	2013-10-10 10:10:23	1024	Success
15	192.168.1.1	192.168.1.2	Test Message 15	2013-10-10 10:10:24	1024	Success
16	192.168.1.1	192.168.1.2	Test Message 16	2013-10-10 10:10:25	1024	Success
17	192.168.1.1	192.168.1.2	Test Message 17	2013-10-10 10:10:26	1024	Success
18	192.168.1.1	192.168.1.2	Test Message 18	2013-10-10 10:10:27	1024	Success
19	192.168.1.1	192.168.1.2	Test Message 19	2013-10-10 10:10:28	1024	Success
20	192.168.1.1	192.168.1.2	Test Message 20	2013-10-10 10:10:29	1024	Success
21	192.168.1.1	192.168.1.2	Test Message 21	2013-10-10 10:10:30	1024	Success
22	192.168.1.1	192.168.1.2	Test Message 22	2013-10-10 10:10:31	1024	Success
23	192.168.1.1	192.168.1.2	Test Message 23	2013-10-10 10:10:32	1024	Success
24	192.168.1.1	192.168.1.2	Test Message 24	2013-10-10 10:10:33	1024	Success
25	192.168.1.1	192.168.1.2	Test Message 25	2013-10-10 10:10:34	1024	Success

(a) Search result page with collected trace messages

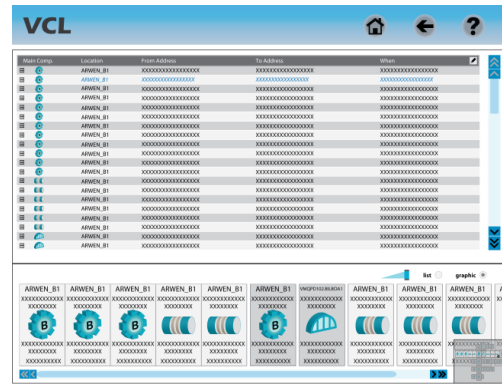
Index	Source	Destination	Subject	Time	Size	Status
1	192.168.1.1	192.168.1.2	Test Message 1	2013-10-10 10:10:10	1024	Success
2	192.168.1.1	192.168.1.2	Test Message 2	2013-10-10 10:10:11	1024	Success
3	192.168.1.1	192.168.1.2	Test Message 3	2013-10-10 10:10:12	1024	Success
4	192.168.1.1	192.168.1.2	Test Message 4	2013-10-10 10:10:13	1024	Success
5	192.168.1.1	192.168.1.2	Test Message 5	2013-10-10 10:10:14	1024	Success
6	192.168.1.1	192.168.1.2	Test Message 6	2013-10-10 10:10:15	1024	Success
7	192.168.1.1	192.168.1.2	Test Message 7	2013-10-10 10:10:16	1024	Success
8	192.168.1.1	192.168.1.2	Test Message 8	2013-10-10 10:10:17	1024	Success
9	192.168.1.1	192.168.1.2	Test Message 9	2013-10-10 10:10:18	1024	Success
10	192.168.1.1	192.168.1.2	Test Message 10	2013-10-10 10:10:19	1024	Success
11	192.168.1.1	192.168.1.2	Test Message 11	2013-10-10 10:10:20	1024	Success
12	192.168.1.1	192.168.1.2	Test Message 12	2013-10-10 10:10:21	1024	Success
13	192.168.1.1	192.168.1.2	Test Message 13	2013-10-10 10:10:22	1024	Success
14	192.168.1.1	192.168.1.2	Test Message 14	2013-10-10 10:10:23	1024	Success
15	192.168.1.1	192.168.1.2	Test Message 15	2013-10-10 10:10:24	1024	Success
16	192.168.1.1	192.168.1.2	Test Message 16	2013-10-10 10:10:25	1024	Success
17	192.168.1.1	192.168.1.2	Test Message 17	2013-10-10 10:10:26	1024	Success
18	192.168.1.1	192.168.1.2	Test Message 18	2013-10-10 10:10:27	1024	Success
19	192.168.1.1	192.168.1.2	Test Message 19	2013-10-10 10:10:28	1024	Success
20	192.168.1.1	192.168.1.2	Test Message 20	2013-10-10 10:10:29	1024	Success
21	192.168.1.1	192.168.1.2	Test Message 21	2013-10-10 10:10:30	1024	Success
22	192.168.1.1	192.168.1.2	Test Message 22	2013-10-10 10:10:31	1024	Success
23	192.168.1.1	192.168.1.2	Test Message 23	2013-10-10 10:10:32	1024	Success
24	192.168.1.1	192.168.1.2	Test Message 24	2013-10-10 10:10:33	1024	Success
25	192.168.1.1	192.168.1.2	Test Message 25	2013-10-10 10:10:34	1024	Success

(b) Search result page with message sequence





(c) Graphical path showing the traceable path



(d) Graphical path being magnified

Figure 3-17. Screenshots from final design for VCL visualization

When users select one component for more information, they click on it in the upper area and the message path that contains the chosen component is on the lower area. On the interfaces two tabs are used to display the message lists that were originally shown in a new window. By default ten tab items are visible directly in the lower area, and when there are more than ten messages being collected, a vertical scroll bar appears automatically on the right side. One tab enables users to see a complete list of the entire collected message while the other is sorted list, giving users different options to view the list based on their needs.

On the right top of the lower area, two radio buttons are designed for switching between list view and graphical view. By click on the graphic button, the message path is presented in graphics, while a slider appears automatically next to the two radio buttons, indicating that magnification is enabled. Moreover, the different background colors imply that the message flows to different destinations, and in the default view the other paths can be found on top of or under the main path. On the right bottom corner a navigation panel provides users the option to see the whole structure, with a highlighted square corresponding to on the current visible view, and users could drag the square to change the visible area. When the graphical path is magnified, the main path takes up the whole height, but the other paths can still be observed from the navigation panel.

The final prototype reflected stakeholders' requirements gathered from the interviews, the message path displayed in graphics, and the paths to different destinations visible. Not so many interactive functions were used here, as the design should be simple and easy to use. Therefore, unnecessary interactions were removed, only the most functional remained, which confirmed the theories described in the book *The Laws of Simplicity* [7].

### 3.3.6 First Iteration for MISP Visualization

The second use case, as discussed in previous section, was to visualize MISP. As a complex database with components connecting to each other in different ways, it was a very difficult task to present it in organized graphics while fulfill user requirements. To get a better idea of what visualization has been used, I started the brainstorming with the help of projects in visualcomplexity [27].

### 3.3.6.1 Brainstorming

A project called Graphopt on the website is an example of visualizing multi-domain representation [28]. It builds models for connecting different nodes and presenting it in a big view so that users are able to see the connections between nodes, as illustrated in Figure 3-18.

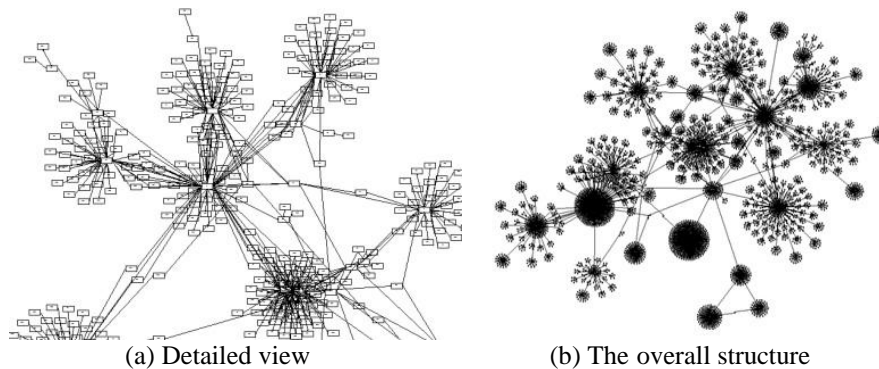


Figure3-18. Graphopt project for visualizing complex data connections

Resource from: <http://www.schmuhl.org/graphopt/>

The concept of this project is similar with the MISP visualization. Both aim at visualizing complex data and showing the relations between the nodes and the applications, and how the applications relate to each other. The left picture is a landscape view while the right one is a detailed view. Technically all the relations are displayed in the graphic, but the question is how to find the ones users want to see in detail? One possible answer is to use a slider for magnification and minification, but when there're too many nodes, the landscape picture could be only dots and lines, which are meaningless and unorganized, and require many more actions to be taken before finding the part users need most. Therefore, this method of visualization does not fit in this situation.

### 3.3.6.2 First Prototype

After reviewing current visualizing projects, I noticed that most of the projects were focused on delivering the new concepts, but the practical purposes were not valuable. Considering the relations between MISP and VCL message path, which is illustrated in Figure 3-1, graphical path that was used in VCL visualization could be used as the base for the intricate connections.

To combine the new concepts with the current system, I started to analyze the characteristics of the project. The most important part was to visualize the connections, and at the same time information such as components' name and location, and the group they belonged to had to be shown in a readable way. With so many details to be presented, my first design was a trail of using 3D models for visualization.

Displayed in Figure 3-19, the first prototype designed in Illustrator was to show the connections spatially. To be more specific, the information was displayed

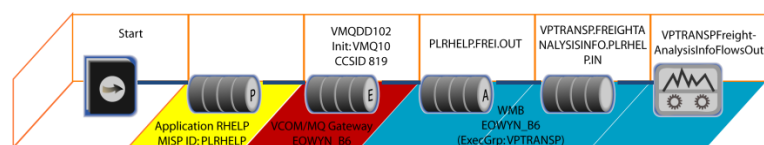


Figure 3-19. Concept demo for MISP Visualization



in three dimensions, each one representing one type. For instance, on the bottom of the layer the color coded groups that components belong to were shown, and on the back the components' information could be found. The components were placed in the middle of the layer connected by blue lines, indicating there were connections between them.

When the components were structured in a linear dimension, the 3D modeling method could be useful. However, when components were connected in a complex network, it was difficult to organize them in multiple layers while keeping the texts visible. Moreover, it would require a proper way to navigate within the network so users could see the landscape clearly. To achieve this, animation could be a possible solution that supports functions like rotating so that users will be able to view the structure in different angles. The first prototype was not feasible both technically and practically, nevertheless it demonstrated a way that had potential to improve.

### 3.3.6.3 Second Prototype

In order to make a doable design that reflected user requirements, I referred back to the current document for showing the application landscape. A sample use case for MISP Visualization is attached in Appendix III, which is an existing landscape created by

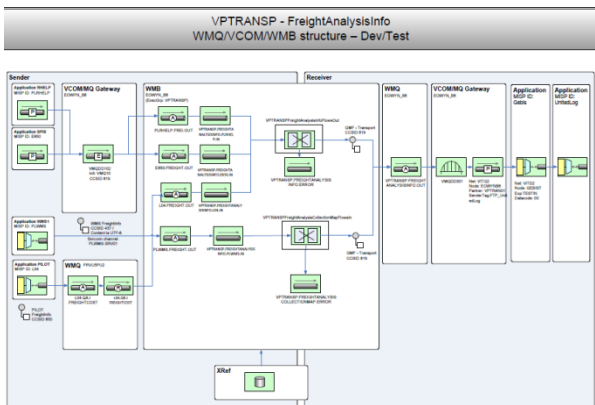
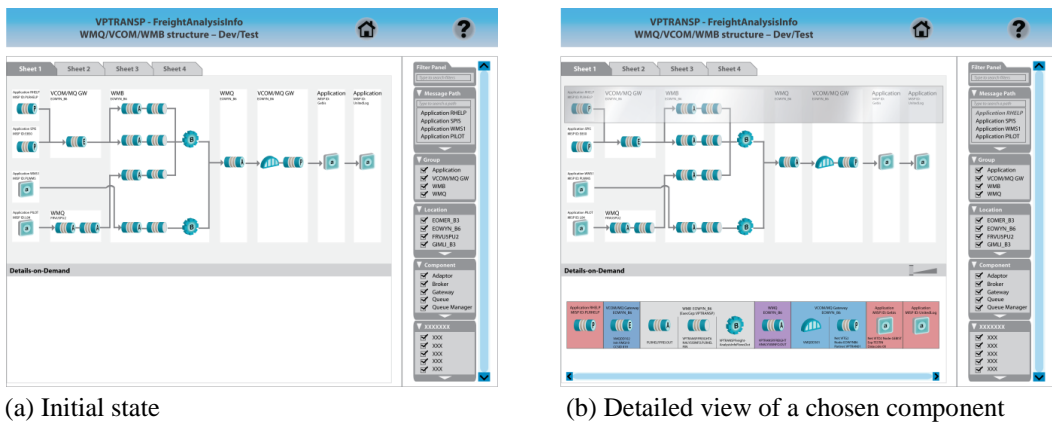


Figure 3-20. Screenshot from a sample MISP visualization use case

Microsoft Visio for demonstrating the connections and between different components under an application. Figure 3-20 presents a screenshot from the use case. All the components and the connections are drawn manually and it will be very beneficial if all the relations could be displayed automatically and presented in interfaces that allow users to navigate and filter based on their demands. Focusing on that part, I started designing the second prototype with the current design as a reference.

The basic structure was similar with the current one, while new functions were added in the prototype. Figure 3-21 presents the interfaces created in Illustrator, and these



(a) Initial state

(b) Detailed view of a chosen component

Figure 3-21. Interfaces from second prototype of MISP visualization

interfaces were based on the sample use case. The overall style is similar with VCL visualization, with the purpose of being consistent. The left one illustrates the initial status of a chosen application, which name is on the top of the page. The whole image of connections is on the left part, while at the bottom of this part Details-on-Demand is where a specific message path is shown. Once users click on any of the component, a complete message flow that contains the selected component is displayed, which is similar with the graphical view in VCL visualization project.

On the right of the interface is a navigation panel with different categories. The categories were designed based on the contents of different attributes, but it required further discussions with stakeholders to find out their requirements and most proper way to divide them. Search boxes and check boxes are two assets used here, because for experienced users they know the name and attributes of the components they want to see and by typing in the values they could quickly find what they look for, and for inexperienced users they could scroll down the list and find which one seems familiar and then select by checking or not checking the boxes in front of each option. Furthermore, as some categories might contain many options, it is not possible to show the long list directly. Therefore, expanding buttons are embedded at the bottom of the frames in categories with long lists. By default four to five options are visible on the screen depending on the frequency users need to select, and a complete message can be viewed by clicking on the expanding buttons. Consequently, a vertical scroll bar appears on the right, which enables users to see all the categories when one or more lists are expanded.

Some details in the prototype need to be mentioned. First of all, when none of the components are selected, the space for detailed view is empty; while any component is clicked, the message path is displayed together with a slider on the title bar, allowing users to zoom in and zoom out for detailed views. Meanwhile, a gray shade on the landscape view covers the part that has been selected, indicating which groups these components belong to. When the detailed path is magnified, the shade is changed accordingly, which could be referred to in Figure 3-22. As in the detailed view only six

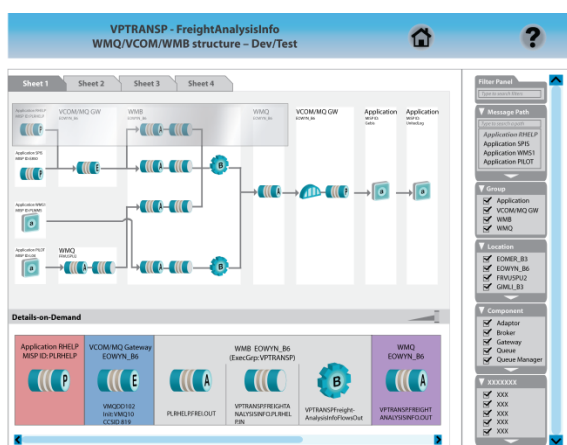


Figure 3-22. Interface from second prototype of MISP visualization with magnified path

stakeholders, I also added the colors to the frames in this prototype for comparisons for later stakeholder feedback interview.

#### 3.3.6.4 Feedback Interview

Instead of conducting feedback interview after creating different versions of prototypes, in this project I decided to get instant feedback from stakeholders in order to save unnecessary time as well as keep the design as the way they wanted. Therefore, I invited two stakeholders, who actively provided sufficient feedback in previous iterations for reviewing current design and gathering their thoughts towards the visualization project.

Their impression of the visualization work was positive, and they both thought that the way of presenting the landscape was clear. Because the overall layout did not change much compared with the current drawings created in Visio, they believed that users would not have any difficulties of using it. Furthermore, the global manager considered the detailed view was beneficial when the landscape was complex. With the help of slider they would be able to extract things they needed out of the complexity. In short they were quite satisfied with the design.

They also offered some suggestions to improve the prototype. The first one was regarding the navigation panel. They did not think the categories I divided into were appropriate according to their experience, and consequently they gave me better opinions on how to define the categories in the navigation panel, which will be reflected on the next prototype. As for the colors of the frames, they did not have special preferences and considered either way had its advantages. However, if there were many colors for different groups, the choice of colors should be careful; otherwise people might get confused about why this specific color was connected with the specific group.

#### 3.3.7 Second Iteration for MISP Visualization

After the feedback interview, I got some suggestions on the navigation panel as well as the way of presenting the graphical path. The ways of presenting the application landscape was approved by the stakeholders, so in the next prototype I focused on improving the interface by combining their suggestion in the new design.

##### 3.3.7.1 Third Prototype

I created the interfaces in Illustrator first, as it was easier to use a graphical designing tool to see how the graphics could be arranged. After that I imported the files to Expression Blend and added some interactions to allow users navigate through different functions. The interfaces of the prototype built in Blend are illustrated in Figure 3-23.

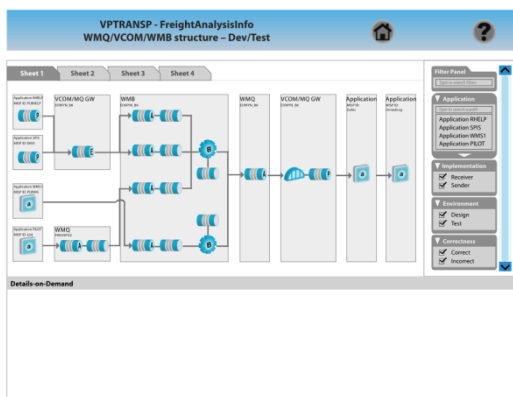


Figure 3-23. Screenshots from prototype 3 built in Expression Blend

The basic concept of this prototype was similar with the previous one, while the layout was reorganized in a way that every part could fit more properly. For intense, according to the stakeholders' suggestions, the navigation panel is now divided into categories of Filter Panel, Application, Implementation, Environment and Correctness, and these categories takes up less space than the previous navigation panel. Rather than having to reserve the whole space on the right of the interface for navigation panel, the spare place could be used for the

detailed view. The detailed view is wider, which allows two more graphical components to be displayed at the initial stage and one more when the path is magnified. Although adding one or two more components does not seem to be a big change, it is helpful when the message path contains too many components and users feel more comfortable to work with by reducing the distance of horizontal movement.

By importing Illustrator files to Expression Blend with a function called WPF SketchFlow Application that supports navigations within different pages, interactions were added and reflected in a SketchFlow Map, shown in Figure 3-24. The lines connecting different tabs indicate that there is navigation between the two pages. The more navigation exists, the complex the lines become. For instance, when users click on the first icon in the first row, a graphical message path would appear in the bottom of the page, indicating which path the clicked component belongs to as the left image presented in Figure 3-25. Same as the second prototype, a

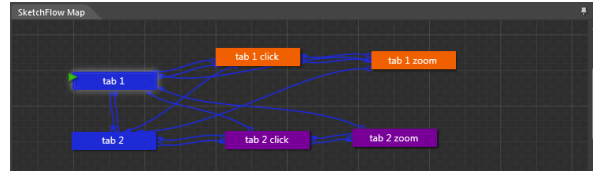


Figure 3-24. SketchFlow Map in Expression Blend

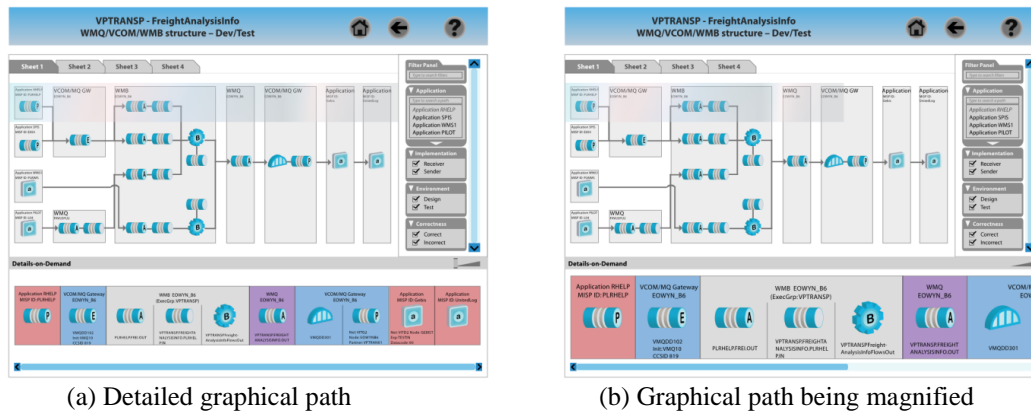


Figure 3-25. Detailed graphical path and its magnification in prototype 3

shade appears on the top area, corresponding to the detailed view by covering all the components contained in the message path. Furthermore, in the graphical path the detailed properties of each component can be found, and the texts being too small to read are solved by adding a slider to magnify the whole path up to 2 times bigger. Users could change the value of the slider to adjust the size of the path, and although this function is not achieved because everything is made of graphical paths, a substitute for it is that once users click on the slider they will be navigated to a page with a magnified path, illustrated in Figure 3-25 (b).

### 3.3.7.2 Feedback Interview

This feedback interview was very fast because the third prototype was a revision based on the second one, and in order to get more stakeholders' opinions about it, four people participated in the interview process, two from the previous sessions, the global manager and a product manager, and two new stakeholders, an application architect and another global manager who also worked with the application-to-application connections closely.

All the participants were very satisfied with the new design, and they considered that if it could be implemented it will bring many advantages to users and application owners. The only thing they were concerned was that because there was too much data in the database and it was not organized very well at this moment, problems might occur in the implementation. However these technical difficulties did not belong to the designing category, and they were software developers' job to find solutions to the problems. Thus the graphical design for MISP visualization reached to a close.

### **3.4 Implementation**

My main responsibility of the thesis work was to design interfaces, and apart from that I could also work with implementation team to help with integrating the design into production. In the original thesis proposal another student should be responsible for working with the implementation part together with me at the beginning, but for some reason the other student quit the project, so I did not have the opportunity to talk these thoughts with a software developer and ask her opinions from a developer's perspective. When the final designs were approved, the stakeholders decided to choose an in-house developer for the implementation and consequently, I got a chance to continue working with the projects, starting with VCL visualization.

#### **3.4.1 Technical problems**

After I uploaded the prototypes built in Expression Blend to subversion and the software developer checked them for compiling with Microsoft Visual Studio for developing Internet based systems, technical problems occurred. In Visual Studio elements should be ready to be programmed, but in WPF SketchFlow Application as its framework they were only paths and graphics, not real scroll bars or sliders. Even Expression Blend could generate code behind the interfaces and the codes could be recognized by Visual Studio, it was impossible to program with non-programming elements. Therefore I had to redo the prototypes with real elements under Silverlight Application in Expression Blend and make sure to use the actual elements for buttons, scroll bars, check boxes and the others.

The differences between WPF application and Silverlight application are various, both from the development's perspective and the GUI designing's perspective. As a non-expert in development field, the differences could be summarized as WPF can be deployed to desktop or run in Internet Explorer, while Silverlight can be deployed to more platforms. In short Silverlight has a 'broader reach' that could be accessed from many systems and browsers [29]. From the GUI designing side, WPF provides more functions than Silverlight; keyboard or mouse events are only available in WPF applications. Compared with WPF, Silverlight losses some functions and among these some could be achieved by developers manually adding in the control tree, but still Silverlight limits the choice of interactions by GUI designers.

As the discussion above indicates, WPF and Silverlight have their own advantages and disadvantages, but for applications run in such a big company, it is necessary to develop in a broader sense in order to avoid potential technical difficulties in other systems. Therefore, to be compatible with Visual Studio for development under Silverlight, the prototype had to be changed.

### 3.4.2 VCL visualization in Silverlight

To change the type of prototypes, I first sought the possibilities to convert existing projects into Silverlight application but it did not work, so I had to start over the entire projects. I first began with the VCL visualization coordinated by the experienced software developer, and along this process I also learnt a lot about how to make Silverlight prototypes compatible for production.

#### 3.4.2.1 Basic Structure of VCL Prototype in Silverlight

The first step was to create the initial interface in Silverlight that contained every element in the exact same size and color as the previous prototype. Instead of simply drawing rectangles or ellipses for buttons and scroll bars, this time I needed to choose the programming elements in the project. Expression Blend has a default set of assets and a toolkit of extensions prepared for UI designers so all I had to do was add the correct ones to the proper position on the interface. The elements were all in default color and shape, only the texts editable, as Figure 3-26 presented the most common ones. The styles could be changed through editing templates without coding, which is a convenient way for interface designers to focus on the style instead of bothering with the code.

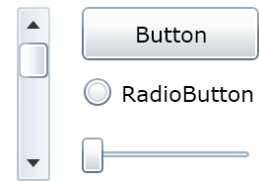


Figure 3-26. Default styles in Expression Blend

It was not a difficult task to put the elements in the right position, but the difficult part was how to organize them appropriately. Similar with Illustrator that users could group many paths, graphics or sub-layers into a layer, Expression Blend also offers this function. However, there are several layout types to choose, and some look quite similar on the interface. Therefore I chose the most common one, grid, which works as a big frame to have everything filled inside without changing any individual values, to group elements according to the categories they belonged to: the banner on top of the interface, the result list view, and the list/graphical view. Other layout types include Canvas, StackPanel, ScrollViewer, Border, and Viewbox, and the icons for each type show the typical characteristics, which could be found in Figure 3-27.

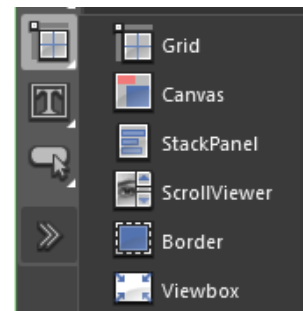


Figure 3-27. Layout types in Expression Blend

#### 3.4.2.2 Review of VCL prototype in Silverlight

The major changes in this prototype, as mentioned above, were to create the interface with real programming elements. Therefore, scroll bars, tabs, radio buttons, and sliders were used in VCL visualization prototype built in Silverlight as the replacement of the false ones made up of graphical paths. The initial design of building the structure of the prototype was almost finished, only remaining customized elements.

Then I asked the software develop to look it over, and some problems revealed. Firstly, grid was not always the choice when a better option was available. For instance, when placing the result item one after another, it was easier to use StackPanel instead of grid, because as the name indicates, StackPanel behaves like a stack that controls the items inside in a vertical or horizontal order. The advantage of having StackPanel was that the 100 items retrieved from each search could be automatically placed in a horizontal order,



so there was no need to manually place the items one by one. However, it was not enough to only use StackPanel for the result list, because not all the 100 items could be shown at the same time, and a scroll bar was necessary when the items went beyond the visible space. A suggestion to build the structure was to group the items in a StackPanel and place it in a ListBox, which could generate scroll bars when the list items were more than the displayed space.

Secondly, in the lower area where the collected list messages and graphical paths were displayed, the layout needed to be clearly defined. For example, in the list view all the items were collected in tab controls, while in the graphical view the graphical path should also be collected in a type of control that could be triggered by changing the value of the slider. To be more specific, the control that the graphics were in should be connected in some way to the slider so that once users moved the slider the size of the graphics could be increased or decreased. Grid did not support that function, and neither did StackPanel, so it was necessary to find a control that allowed a connection to be built so that the intended function could be achieved.

Thirdly, as the data was dynamic, the prototype should not contain the fixed data, but instead use templates for each item and then through programming to fit the actual data in the templates. This work required programming skills, and the software developer would be responsible for it. This one had no conflict with the first two, for the reason that choosing the proper ways of grouping items was the fundamental of building templates, and with fixed ‘fake’ data how the actual interfaces would look like could be simulated.

#### 3.4.2.3 Further development of VCL Prototype in Silverlight

After the review, I was clearly instructed about the elements that needed revisions and improvements, and my next focuses were to embed the first two problems discussed above into the prototype.

The characteristic of ListBox is that when the size the ListBox is set to a fixed number, all the items inside are stretched to fit the width, while the height is a default number that could be changed in the template. Once the height of items is more than the fixed height of the ListBox, a scroll bar is generated that allows users to scroll down and see all the items. A brief example is illustrated in Figure 3-28, in which nine items were added to the ListBox with the height of only showing seven rows directly. The function that ListBox provides could be applied to the result list, in which 100 results are displayed, as Figure 3-18 demonstrates in the upper area. Setting a proper height and width to the ListBox that exactly replaced the original grid and adding components to the ListBox made the upper area look the same as before, but the biggest difference was that the number of components added to the list could be dynamic while only 20 were shown at one time.

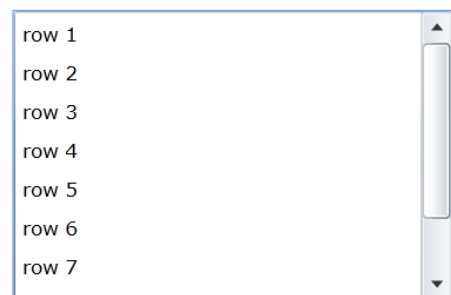


Figure 3-28. An example of illustrating how a ListBox works

Finishing with the result list in the upper area, the next task was to find a control to group the graphics that enables graphics to transform along with the change of value of the slider in the lower area. By looking up on the Internet a tutorial of Silverlight toolkits introducing the use of ViewBox was found that could be used to resize the graphics and therefore achieve the intended purpose [30]. ViewBox could resize itself based on the content, and as the number of components in a message path was dynamic, it was not certain how many images would be collected in the graphical view, therefore the characteristic of ViewBox precisely fit to the need.

During several pilot tests of using different controls to build the structure of the graphical view, the final one was that the detailed texts and the graphical icon were grouped in a grid as an individual component, and these components were grouped in a StackPanel, which was placed inside a ViewBox. The reason to build a structure like this was because a grid worked as a collection of objects that did not change any attributes inside it, and as a result the position of the grid was fixed. In this scenario the components should be placed one after another in a row, and with the help of StackPanel the components were collected in a stack, without the need to manually align them. Meanwhile StackPanel also worked as making the grids as a whole so they could be treated as one ‘image’ in a ViewBox. It avoided the trouble of linking the transformation individually to each grid for magnification.

In addition to the two major improvements, during the collaboration with the software developer, I found that a problem exists in the current system that should be solved appropriately. That was when the texts of the FromAddress or ToAddress in the list view were too long to be displayed in some cases, they were just cut off. In Figure 3-29 the long texts are highlighted. Even if the cut-

ID	Details	Host Name	Location	From Address	To Address	Offset
322896223	IS	WWW	ARVIL_81	172.16.1.100:80	172.16.1.100:80	2011-09-01 12:29:00.000 +02:00
322896227	IS	WWW	ARVIL_81	172.16.1.100:80	172.16.1.100:80	2011-09-01 12:29:00.000 +02:00
322896233	IS	WWW	ARVIL_81	172.16.1.100:80	172.16.1.100:80	2011-09-01 12:29:00.000 +02:00
322896240	IS	WWW	ARVIL_81	172.16.1.100:80	172.16.1.100:80	2011-09-01 12:29:00.000 +02:00
322896245	IS	WWW	ARVIL_81	172.16.1.100:80	172.16.1.100:80	2011-09-01 12:29:00.000 +02:00
322896246	IS	WWW	ARVIL_81	172.16.1.100:80	172.16.1.100:80	2011-09-01 12:29:00.000 +02:00

Figure 3-29. Example of list view containing long texts

off happened occasionally, it was not a good way to deal with the long texts. After discussing this issue with the software developer, we decided to use GridSplitter to solve this problem. GridSplitter is a control that allows users to resize dynamically the width or height of grid cells [31], and adding it to the columns between FromAddress and ToAddress would let users drag the border of the column left or right to increase the width of right or left column. It was not the best solution, because dragging requires horizontal movement that is in the opposite direction of scroll bars and users prefer one direction movements rather than moving around. However, as mentioned above this situation does not occur often, this design was the most convenient to develop based on the current prototype and not much extra work was needed.

By completing the changes and improvements on switching from the original WPF to Silverlight, the prototype was able to be compelled in Visual Studio for the preparation of implementation. The software developer built the templates of the structure of the layout based on the prototype, and then bound the data to the prototype so it was deployed in the production.



### **3.4.3 MISIP Visualization in Silverlight**

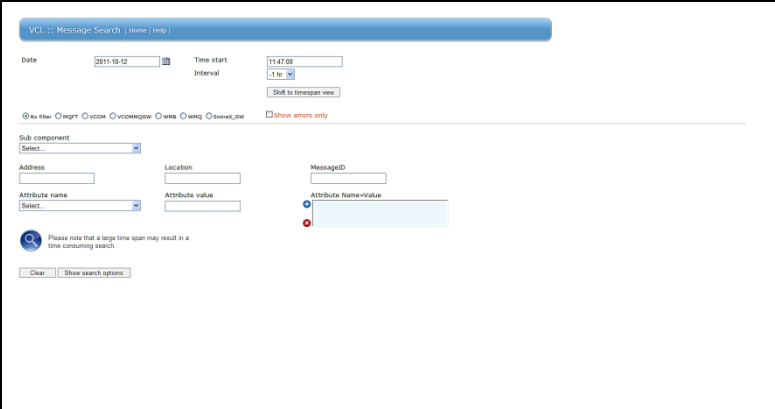
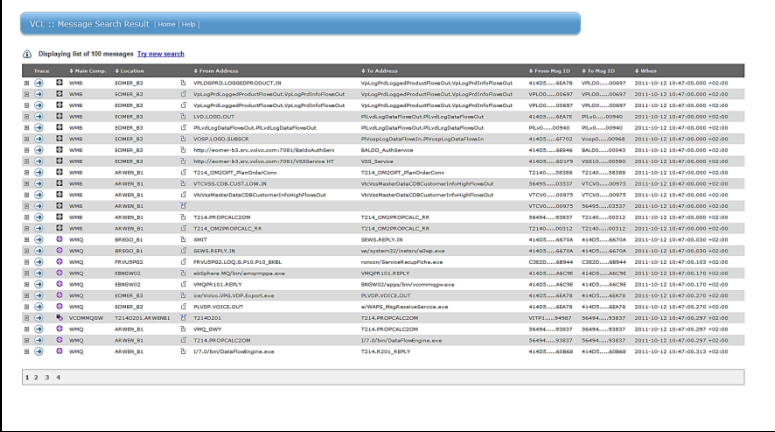
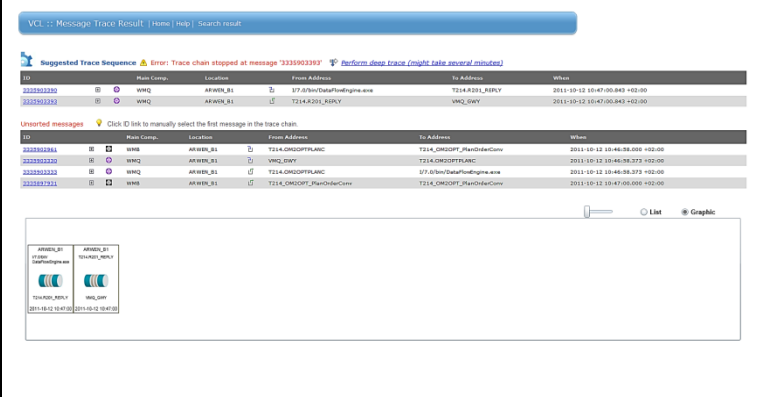
Unlike the finished VCL visualization project, MISIP visualization is still in process for the reason that so much data in the database needs to be sorted out before building the interfaces to show the relations and connections. Therefore my task in this one did not include the collaboration with the software developer for the production. However, I was still required to build the prototype in Silverlight with all the elements programmable, so that they could use it in the future when the data sorting process is complete.

With the experience of creating VCL project in Silverlight, the majority went smoothly. The only difficult part of shifting it to Silverlight was that the structure of showing the landscape, as the data was very dynamic so was the number of columns and rows. To build the logic behind the interface I firstly did some simple sketches to show the concept, and the final decision was that since everything could not be fixed, it had to be achieved with the help of coding. I discussed this issue with the software developer and he considered that it was doable but would need some effort to work on it.

## 4. Results

### 4.1 Final Design for VCL Prototype in Silverlight

After months of working on the project, the final design for VCL visualization was deployed in the system, which will be launched to replace the current one soon. Compared with the previous prototypes, the final design built in Silverlight did not have much difference, but behind the interfaces the structures and the use of programming elements enabled the integration to be carried out successfully. Figure 4-1 presents some screenshots from the Internet Explorer in the left column, which demonstrates how the interfaces will look like in the real working environment. The right column explains the screenshots in detail, with the focus on the designing area.

	<p>The home page of VCL system offers several options to filter the search criteria. The interface of the home page was designed by another in-house developer who is also responsible for changing the system from an old structure to Silverlight.</p>
	<p>A result page appears after clicking on the search button on the home page. One hundred results can be retrieved at each search, and on each page 25 is presented directly. The arrows on the second column of the interface will lead to the page where the message sequence will be displayed.</p>
	<p>After clicking on the arrows as mentioned above, the interface with traceable and unsorted messages is shown both in list and graphical view. In the left case, as only two messages were sorted, there were only two graphical components in the graphical view.</p>

(Continued on next page)

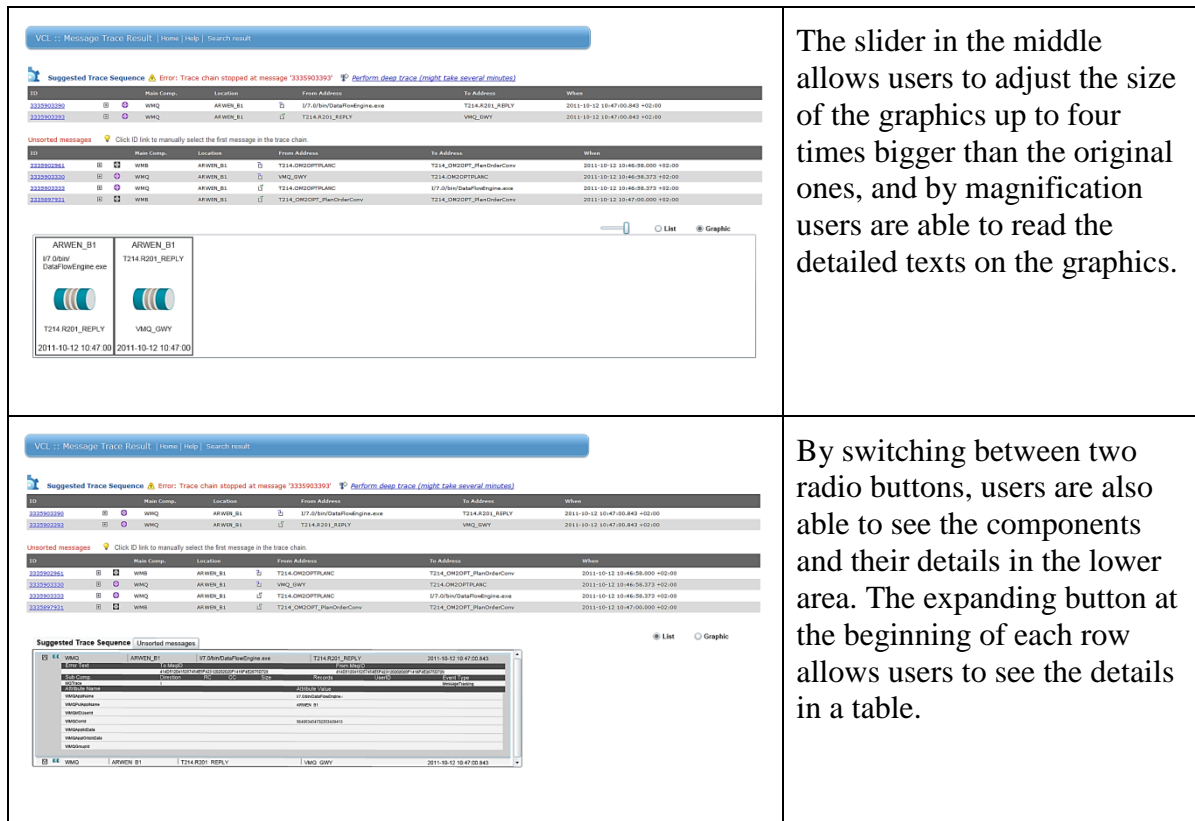


Figure 4-1. Screenshots of final design of VCL visualization project presented in Internet Explorer

The graphical path in the final design is placed differently from the original design, as the initial intention was by putting the graphics on the bottom of the page, users did not need to navigate back and forth to see the message path a selected component is in; however, during the implementation phase, it was decided to be placed on the next page, with the consideration that some users might not have screens big enough to show a complete view. At the current stage the graphical path was displayed together with the list of message sequence, while some modifications would be applied later to adjust the content of the interface so that the graphical path could fit into the screen.

## 4.2 Final Design for MISP Prototype in Silverlight

As discussed in previous section 3.4.3, MISP visualization project is still under progress, and the interfaces I created for the thesis work will be implemented later once the data is organized well. Figure 4-2 depicts the final design by a screenshot from the Internet Explorer with fake data in it. In the figure the landscape was not complex, so users were able to see everything clearly, but if there were over 100 components in a landscape, it was hard to see the connections, so by using the filter panel users could select which components they wanted to be displayed in the interface, and some

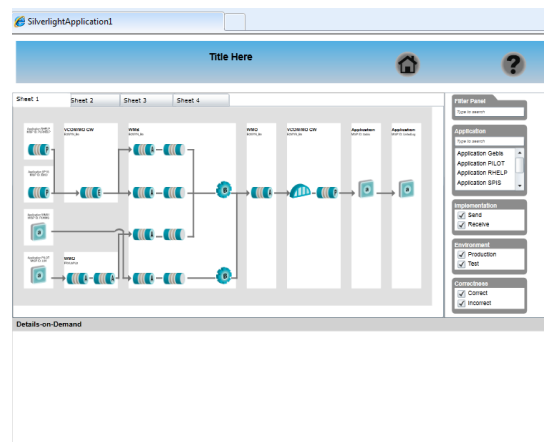


Figure 4-2. Final design for MISP visualization project built in Silverlight

unwanted data could hence be filtered out. To enhance the usability, four categories of filtering data were provided based on users' demands. Moreover, the detailed view in the lower area allowed users to select a specific component and see the connections the component has, and not only graphical path was shown, the attributes of the component could also be found in the detailed view.

There are no interactive functions in the current Silverlight prototype, as it is not built with templates and synchronized with actual system yet. After building the project in Silverlight and designing the elements to be integrated in the production, I handed it to the software developer and explained the concept of how the interaction should work. He will continue the work by compelling the project with Visual Studio, adding the logics behind the interfaces and using animations to show the complex relations.

## **5 Discussion**

### **5.1 Method**

Some methods have been introduced in classes of Interaction Design Methodology and Method of Interaction Analysis, and to sum up from these different methods, the basic idea is to find what users need in questions based or observation based methods. The most common ones are interviews, questionnaires, focus groups and participatory observations, for instance. Each method has its advantages and based on the course lecture, they are served to investigate needs and requirements, because in most of the cases users do not know what they really want, and it is the designers' responsibility to find out the most important information.

In the working environment while the thesis work is conducted, however, the situation is different from the course lecture, as the stakeholders in the company are clear about what they want to be implemented and how it should look like. Although they did not have a specific picture of every element, during the interviews they expressed their wishes in details and gave personal suggestions on what the interface could be like. It is not contradictory with the class work, as in this case they already had the topic of the thesis work in mind and the task was about their daily work which they are familiar with.

In addition, the experience of conducting interviews in the company is different from the ones in school for course assignment. The biggest difference is that when interviewing other students at school, they normally do not have a very clear focus and their answers can be whatever comes up to the minds. Yet in the company all the interviewees knew the topic and had plenty of experience, so their feedbacks were more from the perspective of the difficulties they encountered during the work.

Another major difference is that the interviewing process is mostly led by the interviewees in the company, while at school participants only answer to the questions. The reason to cause this difference is that because in the thesis work when the stakeholders attended the interview, they did not only answer questions, they also demonstrated their tasks and by doing that they could discover more problems they would like to be taken care of. In other words, they hold the initiatives in the interviews to provide information that is useful for the design and development of the thesis work.

### **5.2 Results**

The final designs of VCL visualization and MISP visualization are presented in section 4, and because there is not much time left for implementing the latter one, only the first one has been carried out in the production with the collaboration between me and the in-house software developer. Although it is deployed successfully, and the problems during the integration of the project from Expression Blend to Visual Studio have been solved finally, there are still some compromises in the VCL visualization project.

During the thesis work, I was not aware that the new VCL system was under development and due to the lack of communication, some inconsistency occurred on the interfaces, the expanding arrows for instance. It was not noticed until the integration was finished, and when we attempted to change the styles correspondingly, some technical errors happened in the style template and it might take longer time to fix them.

As a result the inconsistency is still there on the interface, and Figure 5-1 gives the example of the expanding arrows. On the two pictures highlighted by a red circle, the



Figure 5-1. Expanding arrows in different styles on the interface

left one is a square with a plus sign in the middle, and when it is expanded, the plus sign turns into a minus sign; while the right one is a square with an arrow pointing at the

bottom, and when it is expanded, the direction of the arrow is changed to the top as the image indicates. Although it is not a problem of using the function, all the elements should be consistent in styles so there are not confusions about the different functions these arrows would trigger [32].

### 5.3 Limitations

The major limitation is that data visualization, as a newly sprung method, does not have many relevant references to learn for this thesis work, especially in terms of visualizing massive data and its daedal relations, while at the same time making the interface in a user friendly way to reflect users' requirements. Although tables, bar charts and flow diagrams have been used for many years to show data in different forms, these traditional visualizations are used to show simple data, with the purpose of comparison or presenting information. However, in this work, the data is more complex and the purposes are various according to the users' role and their different requirements. Therefore, finding an appropriate way to present data is a challenge throughout the whole designing and developing process. In the specific working environment the requirements are relatively higher than concept designs, for instance the design shouldn't be over fancy or sophisticated. As a result, lacking of effective references makes the final results present the data in a graphical approach whilst still restricted within the traditional method of tables and flow charts.

Another limitation is to build prototypes in an unfamiliar application, Microsoft Expression Blend and work together with the production for integrating the project into practical use. As introduced in previous section, the application is not difficult for drawings simple graphics or creating interactions in animation, but the interactivities are restricted depending on which mainframe users choose. In the process the prototypes were firstly built in WPF that offers more interactive events. When integrating the prototype into production, it had to be turned into Silverlight framework to be able to deploy successfully, and some mouse or keyboard events had to be removed. As an interface designer, I did not focus much on the production side and ignored the fact that different frameworks would cause different result. For this reason, revisions were made at the beginning of implementation and after collaborating with the software developer the problems were eliminated. Lack of experience firstly became a barrier to deploy the project but as more practices were made, the limitation could be avoided.

### 5.4 Collaboration

Although the process of the thesis work has been delayed due to the absence of another student being responsible of the software development work, it was nevertheless a practical and valuable experience of working on reallife tasks together with an in-house

developer who joint in the work after the designing process finished. The collaboration was very helpful in terms of effective communication and patient instruction.

With little knowledge of Silverlight framework, I encountered many difficulties like choosing the right programming elements and arranging elements in different layout types. The help from an experienced developer was not only solving the current problem, but also let me know how to handle with similar situations. Moreover, because the application Expression Blend was used in the development for the first time, when some technical problems occurred during the implementation, we had to search for solutions on the Silverlight forums and fortunate enough most of the problems were resolved properly.

In addition, the effective communication allowed the collaboration to go smoothly and it avoided the waste of time caused by misunderstandings or miscommunication efficiently. The software developer is not only experienced in his profession, but he plays the role of being a mentor as well. The collaboration reflects the reallife situation when people work together on the task and it is an important experience I learned from the thesis work.

### **5.5 Scientific Contribution**

The biggest contribution the thesis work could provide to future scientific research is that it uses a modern data visualization approach to visualize complex data in an on-the-job environment, which could be used as a reference of how the modern approach could be applied in the real life tasks. Nowadays, most of the graphical designs stay in the concept designing stage, and it is time to promote them into practices. However, the problem of using the concept designing is that in most cases the designs are too sophisticated and could not be applied directly. How to make the designs more feasible and usable is an important topic in the field of data visualization, especially in terms of graphics and the complexity of data.

Although the final result of the thesis work did not fully reflect the concept of using pure graphics to visualize data, it could arouse discussions in the scientific field of how to balance the design to meet the requirements that companies may have, as well as whether the graphical designs should be a compromise between the usability and the production. As in most cases the graphical or interface designers and the software developers are different groups of people, and when the design is too complex to be implemented, the developers tend to seek easier ways to get work done. These issues require further research to be carried out in order to find better solutions when using data visualization for practical tasks.

### **5.6 Future Work**

Although VCL visualization project has been successfully deployed in the system, there are some modifications that can be applied to the interface look better. As all the styles of different elements are stored in the project, it is possible to make the elements coherent in styles and the interface will look more consistent.

The data in MISP needs to be sorted out in order before the MISP visualization takes into implementation. It requires a lot of time and effort, and unfortunately it cannot be

finished during the thesis work. The prototype has been handed over to the in-house developer, and once the data is organized logically, the project can be deployed and the user experience will be enhanced by visualizing the components, their connections and the application landscape in graphics.



## References

- [1] Help for Managing Integration Solution Provisioning. MISP Help Center. Volvo IT Internal Site. Retrieved on Sep. 20, 2011.
- [2] V. Friedman. Data Visualization and Infographics. 2008. Smashing Magazine. Available at: <http://www.smashingmagazine.com/2008/01/14/monday-inspiration-data-visualization-and-infographics/>. Retrieved on Aug. 23, 2011
- [3] V. Friedman. Data Visualization: Modern Approaches. 2007. Smashing Magazine. Available at: <http://www.smashingmagazine.com/2007/08/02/data-visualization-modern-approaches/>. Retrieved on Aug. 23, 2011
- [4] Flickr Time. Available at: <http://www.hottoast.org/convexstyle/flickrtime>. Retrieved on Mar. 3, 2011
- [5] Fidg't Visualizer. Available at <http://www.fidgt.com/visualize>. Retrieved on Mar. 3, 2011
- [6] P. Morville. Ambient Findability. 2005 1<sup>st</sup> edition. O'Reilly Media, Inc.
- [7] J. Maeda. The Laws of Simplicity. 2006 1<sup>st</sup> edition. The MIT Press.
- [8] M. Fowler. The New Methodology. 2005. Martin Fowler. Available at: <http://martinfowler.com/articles/newMethodology.html>. Retrieved on Feb. 19, 2011
- [9] Lecture 4 Process and Method: an Introduction to Rational Unified Process. ADT TeamPlace. Volvo IT Violin. Retrieved on Dec. 4, 2011
- [10] B. Kent. Agile Software Development. 1999. PM Briefcase. Available at: <http://www.pmbriefcase.com/methodologies/50-software-development/55-agile-software-development.html>. Retrieved on Dec. 7, 2011
- [11] J. Smith. WPF Apps with the Model-View-ViewModel Design Patten. 2009. MSDN Magazine. Available at: <http://msdn.microsoft.com/en-us/magazine/dd419663.aspx>. Retrieved on Jan. 9, 2011
- [12] J. Gossman. Introduction to Model/View/ViewModel Pattern for Building WPF Apps. 2005. MSDN Blogs. Available at: <http://blogs.msdn.com/b/johngossman/archive/2005/10/08/478683.aspx>. Retrieved on Jan. 9, 2011
- [13] About Silverlight. Microsoft Silverlight. 2011. Available at: <http://www.microsoft.com/silverlight/what-is-silverlight/>. Retrieved on Apr. 10, 2011
- [14] VCOM, Operation Level Specification. 2009. Volvo IT Violin. Retrieved on Feb. 5, 2011
- [15] M. Maguire and N. Bevan. User Requirement Analysis. 2002. Kluwer Academic Publishers. Available at: [citeseerx.ist.psu.edu](http://citeseerx.ist.psu.edu). Retrieved on Mar. 9, 2011

- [16] C. Larman. Applying UML and Patterns. 2005 3<sup>rd</sup> edition. Prentice Hall.
- [17] V. Sauter. What is Prototyping? 1999. University of Missouri-St. Louis. Available at: <http://www.umsl.edu/~sauterv/analysis/prototyping/proto.html>. Retrieved on Mar. 3, 2011
- [18] G. Gaffney. What Is Usability Testing? 1999. Information & Design. Available at: [www.infodesign.com.au](http://www.infodesign.com.au). Retrieved on Mar. 3, 2011.
- [19] TeamPlace: What? Why? When? Volvo IT Violin. Available at: <https://teamplace.volvo.com/info/default.aspx>. Retrieved on Dec. 2, 2011
- [20] Global Integration Infrastructure. TeamPlace. Volvo IT Violin. Available at <https://teamplace.volvo.com/sites/global-integration/default.aspx>. Retrieved on Aug. 5, 2011
- [21] Next generation integration OS platform. Volvo IT. 2010
- [22] Volvo Group Icon Visual Style. Design & Usability. Volvo IT Violin. 2007 Available at: <https://teamplace.volvo.com/sites/application-development/default.aspx>. Retrieved on Aug. 5, 2011
- [23] What Is Expression Blend. Microsoft Expression Blend. 2010. Available at: [http://www.microsoft.com/expression/products/Blend\\_WhatIsExpressionBlend.aspx](http://www.microsoft.com/expression/products/Blend_WhatIsExpressionBlend.aspx). Retrieved on Aug. 12, 2011
- [24] M. Cagan. High-fidelity Prototypes. 2008. Silicon Valley Product Group. Available at <http://www.svproduct.com/high-fidelity-prototypes/>. Retrieved on Aug. 5, 2011.
- [25] Apache Subversion. About Subversion. 2011. Available at: <http://subversion.apache.org/>. Retrieved on Oct. 5, 2011.
- [26] T. Bergmann. Identify Stakeholders. Ultimate PMP® Exam Prep Study Guide. Available at [http://www.truesolutions.com/pdf/identify\\_stakeholders.pdf](http://www.truesolutions.com/pdf/identify_stakeholders.pdf). Retrieved on Aug. 14, 2011
- [27] A. Cooper. About Face 3: The Essentials of Interaction Design. 2007 3<sup>rd</sup> edition. Wiley Publishing, Inc.
- [28] Visual Complexity. Available at: <http://www.visualcomplexity.com/>. Retrieved on Mar. 23, 2011.
- [29] Graphopt. Visual Complexity. Available at: <http://www.schmuhl.org/graphopt/>. Retrieved on May. 10, 2011.
- [30] J. Marsman. When Should I Use WPF vs. Silverlight? MSDN Blogs. 2008. Available at: <http://blogs.msdn.com/b/jennifer/archive/2008/05/06/when-should-i-use-wpf-vs-silverlight.aspx>. Retrieved on Aug. 13, 2011

- [31] J. Angel. Silverlight Toolkit: ViewBox. Justin myJustin. 2008. Available at: <http://blogs.silverlight.net/blogs/justinangel/archive/2008/11/06/silverlight-toolkit-viewbox.aspx>. Retrieved on Aug. 3, 2011
- [32] N. Raychev. Using the GridSplitter control in Silverlight. Silverlight Show. 2008. Available at: <http://www.silverlightshow.net/items/Using-the-GridSplitter-control-in-Silverlight-2-Beta-1.aspx>. Retrieved on Aug. 3, 2011
- [33] J. Tidwell. Designing Interface. 2005 1<sup>st</sup> edition. O'Reilly.

## Appendices

### Appendix I. Outline of Stakeholder Interview

#### General Questions

1. How often do you use MISP?
2. Do you consider yourself as an expert user?
3. Do you think it is easy to use?
4. Have you ever had any difficulties of ordering tools in MISP? If yes, what are they?
5. Do you think the current functions are enough? If not, what should be added?
6. What do you think should be kept on current interface?
7. What do you think needs to be changed?
8. What relevant information do you expect to see on the interface?
9. How would you want it to be visualized? (in text, graphics, or others)
10. What suggestions or comments do you have regarding MISP, its interface and ease of use?

Predefined tasks	Questions
• Visualize the sequences	<ol style="list-style-type: none"><li>1. Do you think it is necessary to view the message sequences/information flow? Why?</li><li>2. Can you describe under what circumstances you would need to view the sequence?</li><li>3. Will it be used frequently?</li><li>4. How would you wish it to be visualized?</li><li>5. What information do you think needs to be included and what needs to be visualized?</li><li>6. What sequences do you prefer to be visualized?</li><li>7. Can you prioritize them?</li><li>8. Do you prefer to have all the information visualized at the same time on the screen? Or do you prefer to have different views that you can choose which to be displayed?</li><li>9. Where do you prefer to have the sequences displayed? (In a new window, in a pop-up window, etc.)</li><li>10. What do you think is the benefit of visualizing the sequence?</li></ol>
• Visualize the infrastructure	<ol style="list-style-type: none"><li>1. Why do you think the infrastructure needs to be visualized?</li><li>2. When do you need to view the infrastructure? How frequently?</li><li>3. What do you expect to be displayed regarding infrastructure?</li><li>4. How do you like the infrastructure to be visualized?</li></ol>

	<ol style="list-style-type: none"> <li>Do you prefer to have all the information shown at the same time on the screen? Or do you prefer to have different views that allow you to customize?</li> <li>Where do you want it to be displayed?</li> <li>What is the benefit of visualizing the infrastructure?</li> </ol>
<ul style="list-style-type: none"> <li>Visualize the connections of components</li> </ul>	<ol style="list-style-type: none"> <li>What connections do you wish to be displayed?</li> <li>What information shall be included?</li> <li>When do you want it to be displayed? i.e. do you want it to be shown as a sort of instruction before you do anything or as a feedback after you select any components?</li> <li>Do you expect to see only the chosen components and their connections or all the available components and their connections?</li> <li>How do you want the connections to be visualized? (tables, pictures, texts, etc.)</li> </ol>
<ul style="list-style-type: none"> <li>Visualize the services and implementation</li> </ul>	<ol style="list-style-type: none"> <li>What are the functions/usages of displaying services?</li> <li>What information needs to be displayed?</li> <li>How do you prefer to visualize the services?</li> <li>What function is it to visualize the services? i.e. is it as a supplement to help users decide which service to choose?</li> <li>What do you think is the benefit of showing the implementation? (being able to trace what's been done, for instance?)</li> <li>How to show the implementations? (a series of pictures, tables, lists, etc.)</li> </ol>
<ul style="list-style-type: none"> <li>Create icons for components</li> </ul>	

## Appendix II. Stakeholders' Requirement Wish List

The stakeholders' requirements are summarized in a table as the next page illustrates. Similar requirements are color coded.

No.	Requirements	Application Support / Application Developers Erkenstam Mats	Application Owners / System Owners Erkenstam Mats	Infrastructure Support Erkenstam Mats	Configuration Administrators Erkenstam Mats	Security Administrators Grastam Lars	Product Manager Andersson Mats	Maintenance Manager Perneblad Ing-Marie	Application Software Architect Andersson Roger	Rundle Timothy
1	Choose between a list view/graphical view	X	X	X	X	X				
2	Export data to files	X	X	X	X	X		X	X	
3	Visualize the logging done in VCL	X	X	X	X	X				
4	View the configuration of the integration infrastructure components	X	X	X	X					
5	Get a graphical view of objects that belong to a certain application	X	X	X	X					
6	Display message routes and a chain of components	X		X	X					
7	Define a template for a certain application and display exceptions to the template	X	X		X					
8	Get a graphical view of the approved access for users accessing objects that belongs to my application		X					X		
9	See a change-log and MISP orders for the displayed MISP CI's			X	X					
10	Display dependencies between different parent CI's			X	X					
11	Add filter for parent CI's				X					
12	Display views that show how the parent CI's are connected to each other				X					
13	Display which CI's a certain access order resolves to					X				
14	Display which access a certain users have					X		X		
15	Display who has the accesses					X		X		
16	Display CI access orders that has not completed (failed for certain CI's)					X				
17	Create views of what deployment implemented/ failed					X				
18	Rearrange 'Application selection' list in alphabet order and easy to find							X	X	
19	Display the message path between two applications						X			
20	Display 'Application Landscape', i.e. connections of one application to other components/ other applications						X			
21	Display 'Network View', i.e. infrastructure relates to each other						X			
22	Display 'Domain View', i.e. connections of all components under one domain						X			
23	Display 'Integration Implementation', i.e. one part of application landscape						X			
24	Get an overview of a list of items								X	
25	Visualize the changes in the landscape and trace the changes									X
26	Visualize the impacts on interdependencies of making changes									X

### Appendix III. Sample MISP Visualization Use Case

This use case is designed in Microsoft Visio, showing the application landscape.

