AALTO UNIVERSITY

SCHOOL OF SCIENCE AND TECHNOLOGY

Faculty of Electronics, Communications and Automation

Miika Aho

Automatic Software Update of an Optical Measurement Device

Master's Thesis submitted for the degree of Master of Science in Technology

Espoo, May 28, 2010

Supervisor: Docent Timo O. Korhonen

Instructor: Tuomas Jylhä, M. Sc.

AALTO UNIVERSITY

SCHOOL OF SCIENCE AND TECHNOLOGY **Abstract of the Master's Thesis**

Author: Miika Aho

Name of the thesis: Automatic Software Update of an Optical Measurement Device

Date: May 28, 2010

Number of pages: 9 + 66

Faculty: Faculty of Electronics, Communications and Automation

Professorship: S-72 Communications

Supervisor: Docent Timo Korhonen

Instructor: M. Sc. Tuomas Jylhä

Abstract text:

An automated software update system was developed for eliminating most customer-side human errors associated with manual file replacement and related operations. A future version is intended to automate in-house update package building process to eliminate many supplier side human errors in configuring the update file package before it is sent to the customer.

The product to be updated is a measurement software delivered to customer factories. Any hardware issues are not dealt with in this project. This software provided by Mapvision Oy runs an optical measurement hardware also provided by the same company. The purpose of the software and hardware combination is optical measurement for quality control of industrial products.

Substantial savings are expected to be gained by deploying this update system. This is largely due to the fact that the customers are situated typically abroad – sometimes also on different continents. As a result Mapvision personnel need to fly to these locations to implement the current strategy of manual updates. The automated system would reduce travel cost, accommodation cost, extra salary for work days in foreign countries and total work time of an individual software update. The above effects are expected to occur when Mapvision personnel would no longer have to perform the extra work of manually installing software at a customer factory.

Keywords: Software update, software automation, usability, heuristic evaluation, remote desktop

AALTO-YLIOPISTO

TEKNILLINEN KORKEAKOULU Diplomityön tiivistelmä

Tekijä: Miika Aho

Työn nimi: Optisen mittalaitteen automaattinen ohjelmistopäivitys

Päivämäärä: 28.5.2010

Sivumäärä: 9 + 66

Tiedekunta: Elektroniikan, tietoliikenteen ja automaation tiedekunta

Professuuri: S-72 Tietoliikenne

Työn valvoja: Dosentti Timo Korhonen

Työn ohjaaja: DI Tuomas Jylhä

Tiivistelmäteksti:

Tässä työssä kehitettiin automatisoitu ohjelmistopäivitysjärjestelmä, jolla eliminoidaan inhimillisiä virheitä asiakkaan toimitiloissa suoritettavasta tiedostojen korvauksesta ja muista päivitystoimenpiteistä. Jatkoversioiden on tarkoitus jatkaa tätä virheiden poistoa automatisoimalla myös päivityspaketin muodostaminen ennen sen lähettämistä asiakkaalle.

Päivitystoimenpiteen kohde on mittausohjelmisto, joka on toimitettu asiakasyritysten tehtaisiin. Laitteiston päivitys ei kuulu tämän projektin piiriin. Ohjelmisto on Mapvision Oy:n tuote ja toimii teollisessa käytössä olevassa optinen mittalaitteessa, joka on saman yrityksen toimittama. Laitteiston ja ohjelmiston käyttötarkoitus on teollisten tuotteiden laadunvalvonta optisilla mittauksilla.

Työssä kehitetystä järjestelmästä odotetaan huomattavia kustannussäästöjä aikaisempaan toimintamalliin verrattuna. Tämä johtuu pääasiassa siitä tilanteesta, että asiakkaat sijaitsevat tyypillisesti ulkomailla – osa myös muilla mantereilla. Seurauksena tästä Mapvision joutuu lennättämään työntekijöitään näihin kohteisiin päivittämään tiedostoja käsin, koska tämänhetkinen ratkaisu ei sisällä muita keinoja. Asiakkaan tiloissa tapahtuvan ohjelmistopäivityksen automatisointi vähentäisi matkustuskuluja, majoituskuluja, menoja päivärahoina ja yhteen päivitykseen kuluvaa kokonaisaikaa. Näitä etuja odotetaan käsityön vähentämisestä ja työn siirtämisestä pois asiakkaan tiloista.

Avainsanat: Ohjelmistopäivitys, ohjelmistoautomaatio, käytettävyys, heuristinen arviointi, etätyöpöytä

Preface

It is a complex task to successfully integrate an update software with an existing target software to be updated and the staff of the system supplier company and the staff of the customer company. In addition this should happen in a cost-effective way with minimal human errors in system development and system application tasks. Paying attention to the usability of these products in supplier offices and in customer environment is a useful in shaping a usage process which will serve the needs of both supplier and customer.

Thinking oriented towards problem solving is something I often find interesting. Programming and more generally product development is a practical field of employment where this is useful if not often necessary. Mapvision Oy provided me with an opportunity to carry out a project employing interesting problem solving tasks and thus I found it a good choice for my master's thesis.

My job application had approximately four general areas of interest I found most preferable when searching for a thesis work project. The remote automatic update of an industrial measurement device got three hits out of four – remote operation, programming and automation – and the choice was easy to accept the job offered.

I would like to express my thanks to Mapvision Oy staff for providing a pleasant work environment in which to develop this graduation work. I found the people most friendly and helpful both in casual conversation and when discussing work matters.

Espoo, May 28, 2010

Miika Aho

Table of contents

Key concepts

- Update:

> Computers. to incorporate new or more accurate information in (a database, program, procedure, etc.).
>
> (update, n.d.)

- Automation:

> 1. *The automatic operation or control of equipment, a process, or a system.*
>
> 2. The techniques and equipment used to achieve automatic operation or control.
>
> 3. The condition of being automatically controlled or operated
>
> (automation, n.d.)

- Usability:

> *The effectiveness, efficiency, and satisfaction with which users can achieve tasks in a particular environment of a product. High usability means a system is: easy to learn and remember; efficient, visually pleasing and fun to use; and quick to recover from errors.* (usability, n.d.)

- Benchmarking:

> A standard by which something can be measured or judged: "Inflation . . . is a great distorter of seemingly fixed economic ideas and benchmarks" (Benjamin M. Friedman).
>
> (benchmarking, n.d.)

-State-of-the-art:

> n. The highest level of development, as of a device, technique, or scientific field, achieved at a particular time: "Forty or fifty years ago the state of the art in radio was represented by crackling noises coming from a console of . . . Aztec-temple shape" (New Yorker).
>
> (state of the art, n.d.)

-User interface

> (UI) The aspects of a computer system or program which can be seen (or heard or otherwise perceived) by the human user, and the commands and mechanisms the user uses to control its operation and input data.
>
> A graphical user interface emphasises the use of pictures for output and a pointing device such as a mouse for input and control whereas a command line interface requires the user to type textual commands and input at a keyboard and produces a single stream of text as output.
>
> A user interface contrasts with, but is typically built on top of, an Application Program Interface (API).

(user interface, n.d.)

-Interface:

Computer Science

1. The point of interaction or communication between a computer and any other entity, such as a printer or human operator.

2. The layout of an application's graphic or textual controls in conjunction with the way the application responds to user activity: an interface whose icons were hard to remember.

(interface, n.d.)

-Remote control:

n.

1. The control of an activity, process, or machine from a distance, as by radioed instructions or coded signals.

2. A device used to control an apparatus or machine from a distance.

(remote control, n.d.)

1 Introduction

1.1 Thesis concept map

This diagram is intended to illustrate the surrounding logical structure of this thesis work. The work context is sketched in a tree structure. The root or starting node of every tree is situated at the upper left corner of the respective colored rectangle. Each crossroads of straight lines represents a branch in the tree. Also each straight line ends in a node representing an individual topic related to this thesis work or it's related concepts.

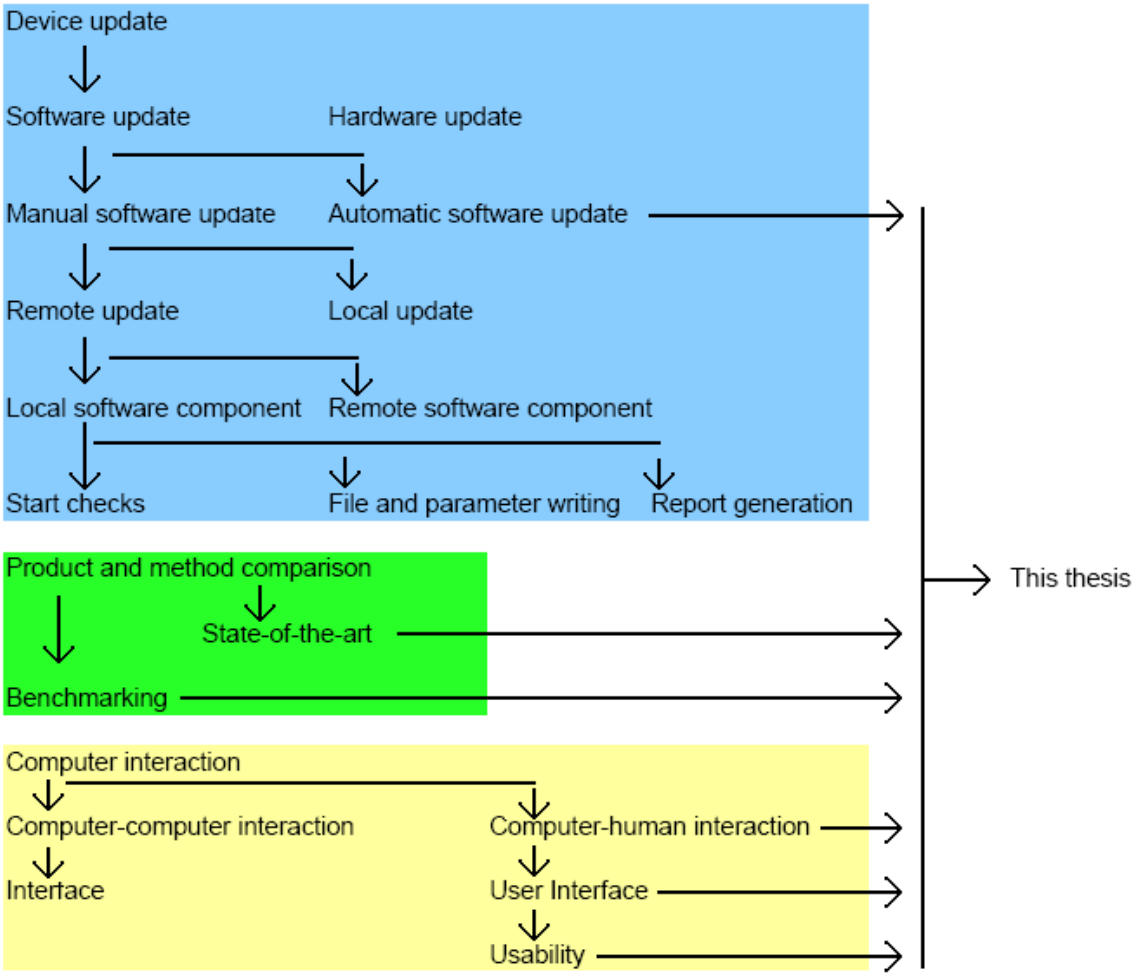

Image 1: Thesis concept map

1.2 The system to be updated

The current system targeted for a software update by this project is a combination of standard PC hardware, a standard Windows operating system, special optical measurement hardware and special software.

The PC hardware is selected mainly by cost. It's main goal is to serve as a platform on which to run a standard operating system and also to connect some external hardware to the custom software. Other functions are executed by custom means.

The operating system in use is standard Windows XP. It merely serves as a link between the PC hardware and the custom control software. No need to specially customize the operating system are known.

The custom control software is the only target of this update solution. The control software is the most sophisticated component of this Mapvision product. It serves both to control the relatively simple custom measurement hardware, to analyze the results obtained from the measurement hardware and to deliver the results to the customer user.

The software developed in this project is divided into two parts. The office side software is referred to as packager. This is the component that is used to produce the update package which is then delivered to the customer factory employee. The other software component is referred to as patcher. This is the software that runs the package and thus executes the final patching operation at the customer side factory computer.

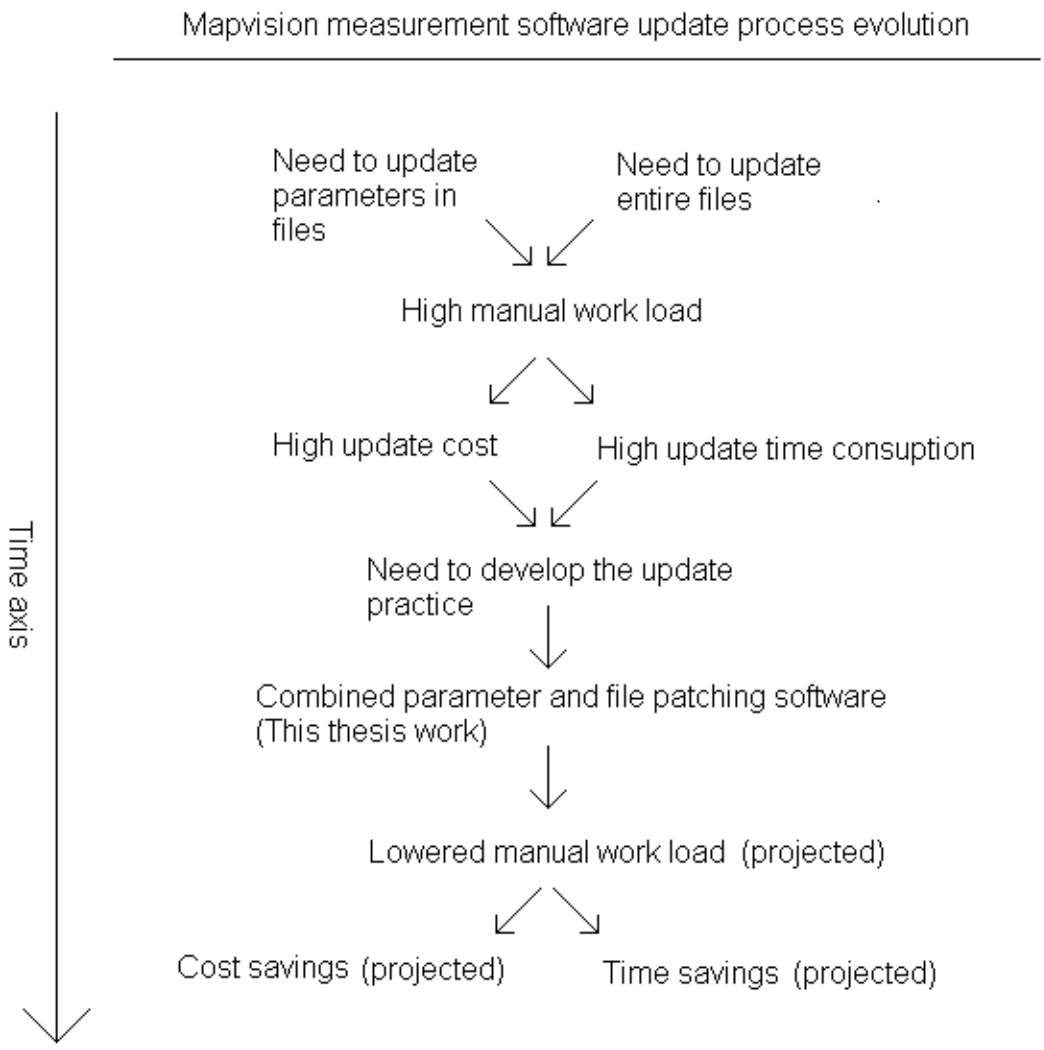

Image 2: The causal relationships of the major factors influencing current Mapvision update process development.

1.3 Historical review

Updating the measurement software of industrial measurement computers has been a recurring demand for Mapvision operations. The need for this has come from various causes. However the solution is always accessing the target computer to alter the software. This practice in turn has sent Mapvision staff onto repeated on-site calls to foreign countries.

The need to frequently travel across the different continents places an obvious and undesirable burden of cost to the company's expenses. This is caused mainly by financing the air travel of the relevant Mapvision employees and providing hotel accommodation near the target site. Another significant source of expense is the extra work time needed by the practice of visiting the production sites frequently. The added travel time and the noisy and otherwise uncomfortable work environment of a factory floor may also be considered undesirable factors by company employees. Thus removing or reducing the use of this practice may have a desirable effect on subjective work experience.

1.4 Research questions

The central research questions for this project are improving three important attributes in a software update operation. The first question is finding out how Mapvision could reduce the time spent by the customer. The customer staff is currently waiting for a software update to their measurement computer for an unnecessarily long time. This is due to the fact that a Mapvision sales person or a technical employee has to personally travel to the target customer site.

The second question is studying how they can reduce the cost of a software update which currently involves sending Mapvision personnel to foreign factories by plane. This is a considerable waste of money in the long run.

The final one is looking into how the staff involved in software updates can reduce human errors happening in a very noisy and distracting factory floor environment. Currently a Mapvision representative sent to a customer factory is not enjoying calm working conditions. This would be preferred given the technically complex nature of the system he or she is expected to reconfigure for production service.

2 Theory

Some general purpose concepts relevant to this project are defined here.

2.1

2.2 Usability

"The practice of designing usable products is called usability engineering.1 The book User-centered system design by Donald Norman and Stephen Draper [1] is a pioneering work. John Gould and his colleagues also worked with usability methodologies in the 1980s [2]. Dennis Wixon and Karen Holtzblatt at Digital Equipment developed Contextual Inquiry and later on Contextual Design [3]; Carroll and Mack [4] were also early contributors. Later, various UCD methodologies were proposed e.g. by [5–10]. The standard ISO 13407 [11] is a widely used general reference for usability engineering." (Jokela, 2005)

Heuristic evaluation and cognitive walkthrough are studied to be the most actively used and also most frequently researched methods. Pluralistic walkthrough and formal usability inspection methods appear to have been either

cannibalized into other techniques or mostly abandoned by users of practical development applications. (Hollingsed, 2007)

Usability inspection is a general purpose concept for certain cost effective strategies for analyzing proposed user interfaces to find problems in practical usability. These are rather informal approaches and easy to use. (Nielsen, 1995)

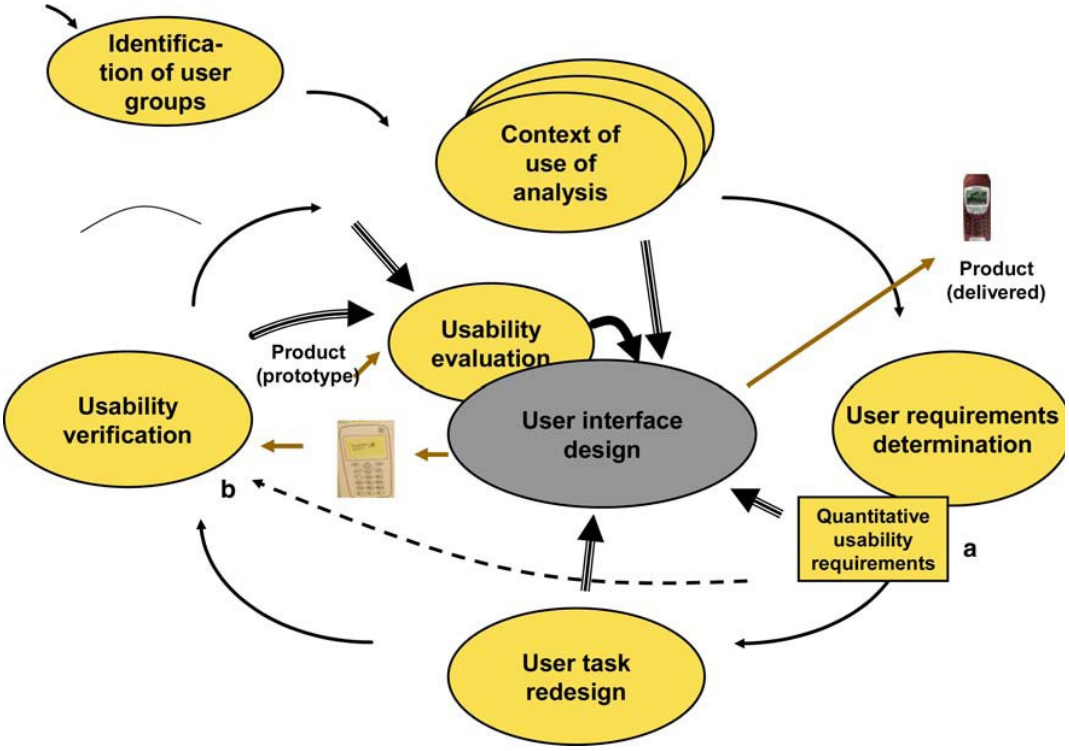

Image 3: A user-centric product development process diagram. (Jokela, 2005)

2.2.1 Potential benefits of usability

One definition of usability is the success of a targeted user in being able to use

the product in a targeted environment and use situation. Investment in usability engineering can:

- Raise sales volumes
- Produce an increased amount of satisfied users
- Lower costs related to development
- Decrease costs caused by product support functions

According to the Usability Professionals' Association Business benefits of usability include:

- Increased productivity
- Decreased training and support costs
- Increased sales and revenues
- Reduced development time and costs
- Reduced maintenance costs
- Increased customer satisfaction

(Usability Professionals' Association, n.d.)

2.2.2 Usability goals

Usability goals are preferably quantifiable measures chosen before testing a product, prototype or concept with typical users. The goals may be divided to four categories: Performance, accuracy, recall and emotional response. By measuring the system before any improvements are implemented one can gain a control test result. Next usability problems are observed and hypothesized solutions are implemented. Now one can run the experiments again and compare the new results to the original control results and look for improvement in the test score.

- Performance

A quantity describing the resources a user expends accomplishing a given task. This may be measured in continuous time or discrete steps with the user interface.

- Accuracy

A measure of the amount of mistakes a user made when trying to perform the given task. It is also often relevant if the mistakes made caused the task to fail or could be corrected with appropriate information and/or methods.

- Recall

A description about how much the user remembers about performing the task immediately after the performance or after a significantly long time without using the system.

- Emotional response

The subjective experiences of the user about the system. The user may feel nervous, happy or something else. It may also be valuable to know the way the user would praise, criticize etc the system to other people.

(usability testing, n.d.)

2.2.3 User profiling

User profiling is the act of dividing the projected users into groups. The dividing criteria should be meaningful to the usability assessing project. One should choose the criteria so that they affect the users' ability to use revelant products. The products being developed or evaluated are of course relevant. Also other products may be relevant if they could affect the usability of the evaluated product when used with it. Also such products may be relevant which employ user interface solutions which may be productively used in near future versions

of the evaluated product.

2.2.4 Heuristics

Heuristic evaluation is conducted by usability engineers examining whether or not the elements of the proposed user interface conform to generally accepted design guidelines. These principles or rules of thumb are called the heuristics. (Nielsen, 1995)

A study compared usability issues found in a user survey against the issues found by applying usability heuristics to the same system. The results showed that 91% of the issues found in the user survey were found in the heuristic evaluation by using competent evaluators. The tested system was an e-learning application. These results estimate heuristic evaluation to be a useful tool for cost-effective searching of usability problems. (Ssemugabi, 2007)

Nielsen's widely used heuristics (Nielsen, 1994) are listed by their original terms, with descriptions.

- *Visibility of system status*

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

- *Match between system and the real world*

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

- *User control and freedom*

Users often choose system functions by mistake and will need a clearly marked

"emergency exit" to leave the unwanted state without having to go through an extended dialogue. Support undo and redo.

- *Consistency and standards*

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

- *Error prevention*

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

- *Recognition rather than recall*

Minimize the user's memory load by making objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate.

- *Flexibility and efficiency of use*

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

- *Aesthetic and minimalist design*

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

- *Help users recognize, diagnose, and recover from errors*

Error messages should be expressed in plain language (no codes), precisely

indicate the problem, and constructively suggest a solution.

- *Help and documentation*

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

Gerhardt-Powals' cognitive engineering principles aim at the much same goals as Nielsen's and can be considered an alternative set of rules of thumb.

- *Automate unwanted workload*

  - free cognitive resources for high-level tasks

  - eliminate mental calculations, estimations, comparisons, and unnecessary thinking

- *Reduce uncertainty*

  - display data in a manner that is clear and obvious

- *Fuse data*

  - reduce cognitive load by bringing together lower level data into a higher-level summation

- *Present new information with meaningful aids to interpretation*

  - use a familiar framework, making it easier to absorb

  - use everyday terms, metaphors, etc

- *Use names that are conceptually related to function*

  - Context-dependent

  - Attempt to improve recall and recognition

- Group data in consistently meaningful ways to decrease search time

- *Limit data-driven tasks*

  - Reduce the time spent assimilating raw data

  - Make appropriate use of color and graphics

- *Include in the displays only that information needed by the user at a given time*
- *Provide multiple coding of data when appropriate*
- *Practice judicious redundancy*

2.2.5 User studies

User studies refers specifically to studies focusing on users. This should not be confused with other studies in usability engineering. More specifically user studies do not especially look into a usable system, tasks, goals or other aspects of usability engineering. Studying users may be done by employing interviews, queries, observations in a typical use environment or other methods of research. The goal of user studies is to establish useful understanding about the needs, desires, skills, relevant pre-existing knowledge, other abilities, preferences etc of the potential users.

2.2.6 User tests

User tests are controlled experiments using a functional prototype of the system under development. Real sample users are given tasks to accomplish by using the system. Their activities are monitored and preferably recorded for any later analysis. The developers or special usability experts then compare the chosen usability goals against real achieved performance of the product when used by

representatives of the chosen user profiles.

2.2.7 Pluralistic walk-through

Pluralistic walk-throughs are a type of user tests. They involve real sample users, developers and other stake holders. A meeting is held between members of the aforementioned parties. A use scenario is walked through by using drafts, screen shots or other samples of user interface elements. All steps are discussed between the participants. (Nielsen, 1995)

2.2.8 Cognitive walk-through

A cognitive walk-through is a rather ambitious enterprise. Here a target user is modeled as a machine like system. This system is considered having input data, memory content, previously known strategies for problem solving and output methods. In a cognitive walk-through developers simulate a target users actions. The user's current goals and relevant memory content are explicitly modeled. This information is used to estimate whether or not the user has the needed information at the right time to carry out the current task. (Nielsen, 1995)

2.2.9 Formal usability inspection

A formal usability inspection is divided into six steps. The roles in this process are strictly divided. The high level strategy is an effort to combine to other usability evaluation methods. These are a heuristic evaluation and a simple version of a cognitive walkthroughs. (Nielsen, 1995)

2.2.10 Feature inspection

A feature inspection revolves around the features of the proposed product like the name suggests. In a feature inspection the needed features for a given user task are listed. This is repeated for each user task that is chosen for evaluation. The lists are checked for certain potential problem sources. Long sequences are both wasting the user's time and increasing the probability of a human error. Awkward steps introduce mistakes by the user. Steps that would be unnatural for the chosen users or steps that would need knowledge or experience that the target users are not expected to possess both lower the probability of the user to try these steps. (Nielsen, 1995)

2.2.11 Consistency inspection

A consistency inspection collects developers from several projects. They then compare a suggested system's interface to their projects. The goal is to check whether or not the interface in question handles similar tasks in same ways as the other systems' solutions. This increases the likelihood that a given user will recognize the new system's interface elements and practices. Many users have previous experiences with other systems where these same choices may have been made in system design. (Nielsen, 1995)

2.2.12 Standards inspection

A standards inspection requires a special expert on a specific user interface standard. The expert goes through the proposed user interface material. During the course of this examination he or she is inspecting all elements and functionality for compliance with the interface standard he or she is familiar with. If several standards are to be inspected a group of experts may be required to

carry out the inspection. (Nielsen, 1995)

2.2.13 About usability in current research and industry

A study looked into the deployment of usability in software development organizations. It found that 29 out of 39 studied groups performed some method of usability evaluation. The most influential problems hindering usability evaluation were a working understanding the field of usability, allocation of resources and the mindset of developers. (Bak, 2008)

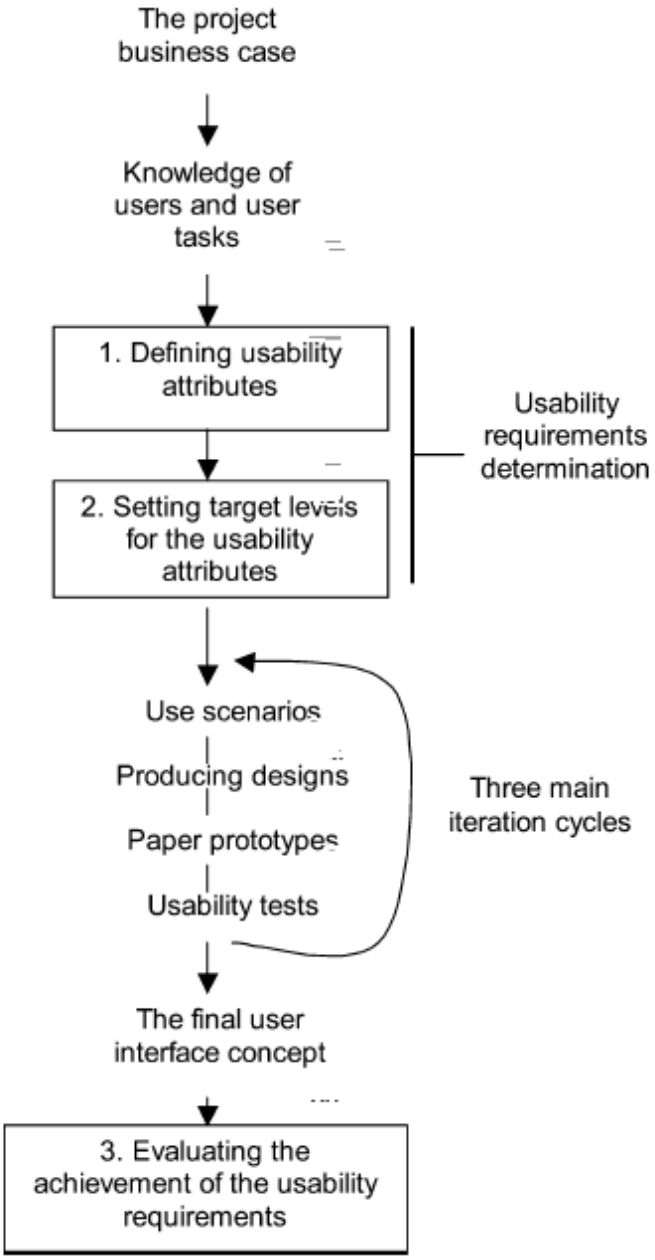

Image 4: A depiction of a user centric project flow. (Jokela, 2005)

The issue of time consumption in the development process has a novel advocate for a future solution. A study looked into the compatibility of current usability evaluation methods with possibilities for automating parts of the evaluation process. Future research may provide a viable alternative for completely manual usability testing. (Ivory, 2001)

In an organization which uses different staff for usability evaluation and product development the input from evaluators to developers may be in the form of usability problems or redesign suggestions. A study was made to estimate the relative usefulness of usability problems and design proposals to developers. The results reported the redesign ideas to provide more utility for the developers than usability problems. Usability issues were seen largely useful for choosing the relative priorities of redesign work. Also none of the participating developers found in the end that they would have wanted to receive only redesign proposals or only usability issues. (Hornbæk, 2005)

Choosing practical usability evaluation methods for a given project may yield very different choices from different developers. The factors to be considered include resource allocation and expected and required accuracy of the results. The mutual complementary and convergence of heuristic evaluation and user testing were confirmed to certain extent in a study. The paper in question compared the effectiveness of heuristic evaluation and usability testing. It was not however found out in the study whether or not the usability problems reported resulted in corresponding problems in actual use of the system. (Law, 2002)

Evaluating an interface with users and evaluating without users have traditionally been two rather distinct processes. This has been reflected in resource consumption in the design procedures and depending on the system potentially in results. An effort to bridge the gap has been made by applying psychological models of human thinking processes and behavioral patterns. These have been

used to produce a model of a user attempting to accomplish a given task with the given user interface. This way one could possibly evaluate a user interface without users and get results very similar to user tests. The proposed strategy is called metaphors of human thinking or MOT.

“The metaphors concern habit, the stream of thought, awareness, association, he relation between utterances and thought, and knowing.” (Frøkjær, 2008)

MOT is most importantly designed to combine a psychological model with thinking metaphors. This is an attempt to uncover relevant new pit holes in sequences of interface actions. These interface issues are expected to match sufficiently well with actual problems found with real users. The efficiency of MOT was tested in three experiments. The first of these experiments demonstrated that the usability issues found by using MOT are more serious than those found with heuristic evaluation. Also the problems found with MOT were found to be more complex to repair. Additionally the problems uncover with MOT were deemed more likely to affect experienced users. In the second experiment MOT was found to find more usability issues than cognitive walkthrough. It also covered a wider range of a reference collection of usability problems. The participants of the experiment reported preferring using MOT over cognitive walkthrough. An important reason for this was that the scope of MOT appeared to be wider. In the third experiment MOT, cognitive walkthrough and think aloud testing. It targeted non-traditional user interfaces. Although the participants preferred using think aloud testing it found few problems uncovered by the other two tested methods. In contrast MOT identified more usability issues than the other two tested techniques. Considering the usability problems found in the experiments applied to system design MOT succeeded better than the control strategies and the results it produced were comparable to think aloud testing. (Frøkjær, 2008)

2.3 Software automation

Software automation refers to programs developed to remove a costly, time consuming and error-prone human agent from a wanted process and replace the human with a cheaper, faster and more accurate computer program. Generally automation is the fourth step in grand scale work development – first being bare hands, second non powered tools, third mechanization by engines and fourth automation by adding some logic control system, typically electronics.

2.3.1 Requirements of automation

2.3.1.1 Environment check

The customer's PC must be checked for the preset attributes to ensure that the user has the right update package in the right computer. This is done by verifying the existence of the measurement software installation directory, the measurement software revision number and it's license validity. All of the above checks are individually optional and chosen at Mapvision offices before delivery of update package to customer factory.

2.3.1.2 Standardized file structure

To be able to perform a controlled software update the wanted update instructions must be saved in some standardized form. An XML format list was chosen for this purpose. The customer-side part of the update software may then follow the preset instructions. Deviations from the normal file structure would be likely to cause problems in the update process. The file structure is preferred by Mapvision staff to remain out of outside hands to avoid unwanted tampering with systems.

The customer side system briefly described at chapter 1.2 (the target system of the update process) must also conform to the expected folder structure. Similarly the expected files must be found in the expected locations. Any missing files will not be searched for by the update software. This would likely be unproductive as any files in unexpected locations would likely not function as expected.

2.3.1.3 Correct preset files and parameter lists

The preset files containing all the update instructions control the entire update process. Thus the integrity of the preset files and other options is a recognized error sensitive factor. It is planned to be automated in future versions of the update software. This is to remove many human errors and to reduce time consumption and memory load of the human operator. This part of automation was however decided to be excluded from the 1.0 version of the update software. The main reason for this decision was to fit the software development project and the master's thesis project to a practical and reasonably controllable time span.

2.4 Remote connection

The goal for using the remote connection is to provide an option to transfer and run the entire update process and retrieve the update result report data without using customer personnel. A customer company may choose this option if they do not wish to be bothered by taking part in update processes. They essentially only need to open the connection for Mapvision personnel. Also work time may be saved by using the same system for future software updates. Then the same network details need not be explained every time.

2.4.1 Chosen system

A Windows XP remote desktop software is to be run over an off-the-shelf VPN software. This was chosen as a readily available option. Also the financial impact was expected to be low. There was already in-house expertise on this piece of VPN software. This all contributed to a relatively easy decision in favor of this solution.

2.4.2 Security

Sufficient encryption is provided by the chosen underlying VPN software OpenVPN. By employing built-in security features (essentially encryption of transmitted data) we saved plenty of development time and increased the confidence level in the solution. This is considered as opposed to developing one's own remote connection and encryption software. Even with a modest set of features this would have imposed a significant delay to the already stretched time table.

2.4.3 Requirements for network access

The VPN software and remote desktop software require certain ports and connections protocols to be set to allowed state on both connecting computers. This is personally negotiated with a customer technical representative to agree an acceptable set of open access. Some customers are expected to allow different port access than others. Hence one provides customization to the customary set of ports suggestion.

3 User centric design work flow

3.1 Chosen methods

3.1.1 Initial problem definition

The goal of the project was to build a Windows program to automate the update procedure of an optical measurement computer in a Windows environment. The update was divided into two parts. The offline part consists of a program that will update files and parameters based on a pre-defined list. The list is chosen at Mapvision offices. The online part consists of creating an internet connection between Mapvision office and the measurement computer at the customer factory. Although the target computer of the update will usually be in routine production use it was not deemed necessary to perform the update with the target software running. This could be accomplished amongst others by adding a redundant computer to handle the measuring operations during the software update process. (Wei, 2003)

3.1.2 Stakeholder meetings

Several stakeholder meetings were held during the process of this project. The participants were among others the business manager, the project supervisor, a technical assistant, the developer and projected future users from Mapvision technical staff and sales staff.

The main objectives were quite clear from the beginning on – the system to be developed is expected to reduce the time spent for software updates into customer computers, reduce the financial cost and time spent for dispatching Mapvision staff to overseas customer factories to manually perform Mapvision

software updates and minimizing the number of human errors in the update process. The manual update strategy consists of using standard Windows tools for placing new files onto the customer measurement computer's hard drive and individually changing parameters in existing files. No fixed measurable conditions were set for the successfulness of the development project. The project is considered a success on the grounds that the system was functioning without any noticeable major technical or human related difficulties in limited office tests.

Other stakeholders should be identified as customer factory managers, whose preference for this product is indirectly influenced by it’s usability by the factory floor personnel. Also all Mapvision personnel should be considered indirect stakeholders as the usability of this product affects Mapvision sales in the long run.

The main high-level requirements from stakeholder meetings were a fast update process and checking for correct environment before performing any changes to the target computer to avoid accidentally manipulating the customer’s local software environment into a random inoperable state as a result of using the wrong update package file. More specific stakeholder requirements were the necessary software development to contain an entire update package in one standalone file to accommodate the needed ease of use at the customer organization in a typical offline update case without any online connection, and finally the option to offer an offline connection alternative to any customers willing to allow the direct use of their measurement computer over a secure VPN connection.

The technical and environmental requirements that arose during these meetings were the need to implement all update functions from start to finish by using only standard PC hardware running Windows XP as the common operating system. The physical environment is either a typical office space in the first step of the

update – the setup of the package file. In the latter part the environment is the factory floor at customer facilities. Here the user will be in an unavoidably noisy environment caused by loud production machinery and typically surrounded by complex distraction with the factory operating around him. This in part adds to the need to keep the customer user interface extremely simple.

3.1.2.1 Key functionality to support user needs

The key functionality to support user needs is in the setup part a diverse but straightforward access to any parts of the update package the user is setting up. The key functionality at the customer side is in a way the opposite of this: the software is programmed to run the update process autonomously once it is started by the local customer user and the user is not asked for any information, all input is limited to the expert user interface at Mapvision offices.

3.1.3 Typical use scenarios

The most common projected use cases are planned to be a “basic” local update process and a remote update process which is done from Mapvision headquarters. These are described in more detail in following paragraphs.

3.1.3.1 Local update

The local update scenario represents a typical situation where the customer organization has not granted Mapvision a permission to access the measurement computer directly through a computer network or has not connected the measurement computer into a computer network. Both of these cases may be the result of the customer company either deeming these actions unnecessary or choosing to avoid them because of carefulness with information security in

networks. Thus it is necessary to access the measurement computer indirectly with the help of a local customer employee carrying the update file into the measurement computer and carrying the report file back from the measurement computer to be emailed to Mapvision staff.

3.1.3.1.1 Configuration

At first an expert user belonging to Mapvision staff collects the relevant new versions of the updatable files and parameters onto his or her computer. The user then runs the Patcher software to build the update package file. The user interface is focused on a tree structure that describes the chronological steps that will take place in the customer side update process. A central part of this tree is the collection of file update operations. The user will configure these operations to add, remove or/and replace any files in the customer side measurement software installation path. This step consists of browsing the hard drive of the office computer for the new file versions and selecting them individually into the file lists of the Patcher software. The expert user also sets the target file paths on the target computer for each new updatable file. The user writes the parameter paths and names and their new values in the cases of parameter creations and replacements. In cases of parameter deletions only parameter paths and names are entered. The user may select to activate any number of the three available environment check procedures – checking the validity of the license of the measurement software to be updated, checking the software revision of the measurement software and checking the existence of the expected installation path of the measurement software. An expert user is offered an option to run any arbitrary number of external command lines during the customer side update process. These command lines are inserted into the desired chronological slot between other update steps in the graphical user interface of the Patcher program. The user may want to order the return of the entire file environment of the measurement software at the customer side measurement computer. This

can be selected to happen before the update process, after the update process or both of these times.

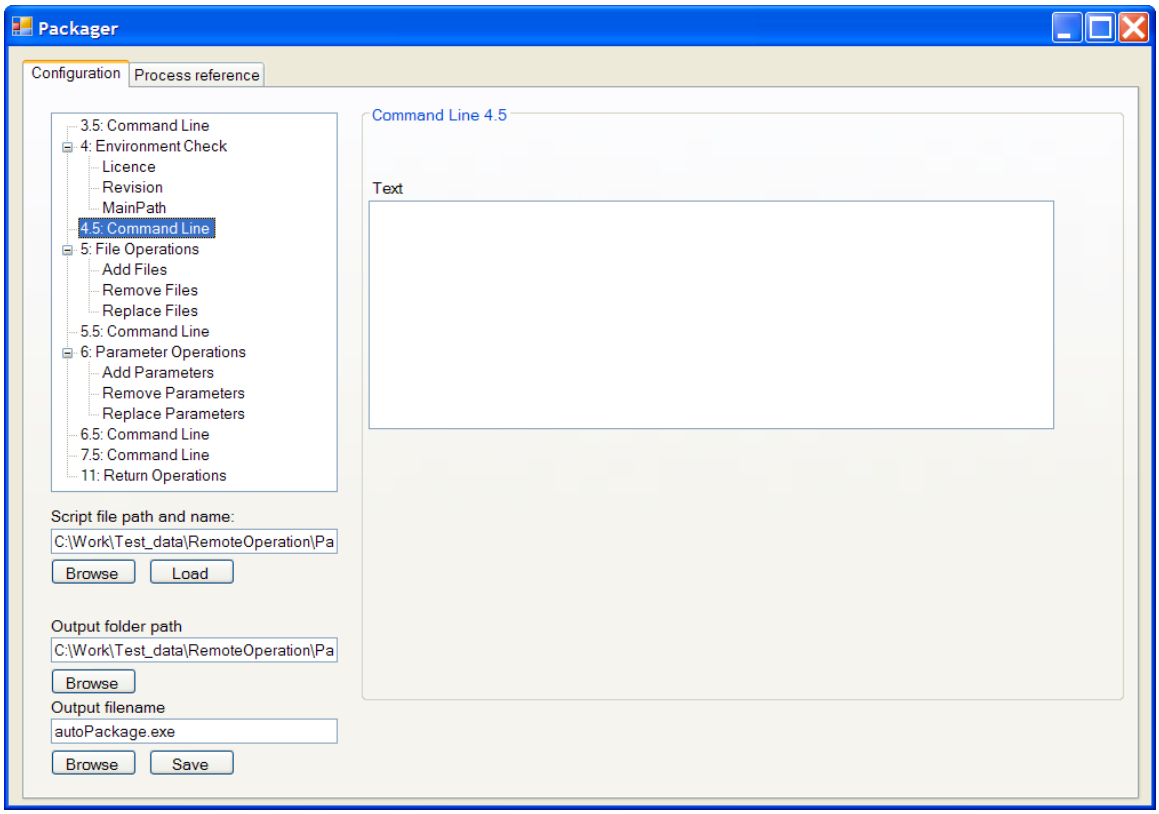

Image 5: Command line branch of the settings tree at the left side of the window. Command line settings at the right side of the window. This software was developed using an integrated development environment (IDE). This application was written in C++. The basic user interface design was based on wishes from Mapvision staff.

Finally the expert user clicks a button to instruct the program to create the self-extracting compressed file to be delivered to the customer. This file contains all updatable files and other needed utility files as a single stand-alone executable application.

3.1.3.1.2 Local procedures

The expert user then emails the newly generated update package file to a local customer user. The customer user moves the file onto the local measurement computer via a memory stick or other media of choice. He then simply clicks on the update executable file and the entire update process runs on the measurement computer without any need to interact with the local user. The update software displays progress information on the screen so that the local user may observe that the update process is indeed taking place instead of any software crash or other malfunction. The flow of information is a strict one way process as the local user is isolated from affecting the outcome of the update program running on the measurement computer. The update package generates a report file describing the actual steps executed in the update process. When every update step is finished, the update executable informs the local user that the process is complete and asks the user to return the update report file to Mapvision staff via email. The local user moves the report file onto the memory stick used earlier and emails the report file back to Mapvision headquarters.

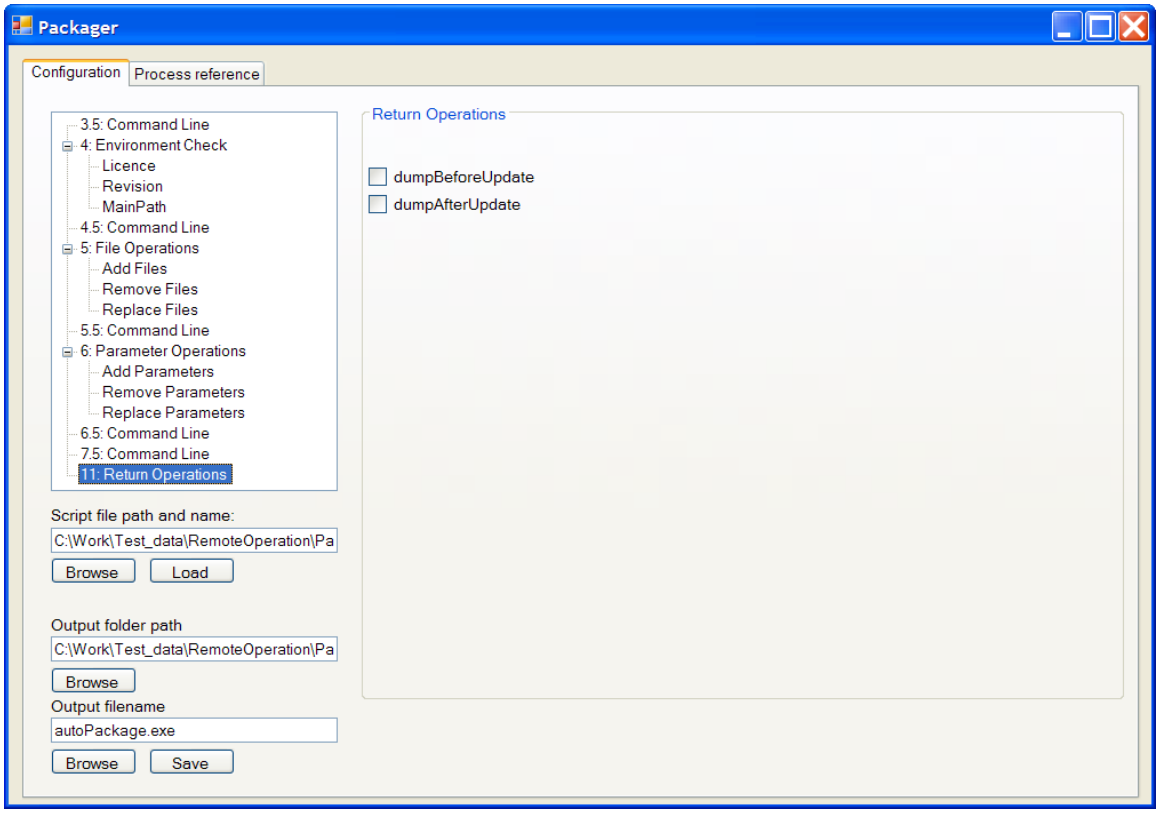

Image 6: Return operations branch of the settings tree at the left side of the window. Return operations settings at the right side of the window.

3.1.3.2 Remote update

The remotely executed update scenario is much akin to the local one. Actually the remote scenario contains most parts the locally executed update scenario as internal steps of the remote process. The additional parts are involved in accessing the customer computer from Mapvision offices or another remote site, thus avoiding bothering the customer users with potentially repeating update tasks.

3.1.3.2.1 Preparations

The first stage is generating the current update package file in the same manner as described in the local update scenario immediately above. The other preparatory phase is acquiring the relevant network access from the local network administration staff of the customer organization. Once the network admission has been negotiated and secured and the necessary connection data has been delivered in form of usernames, passwords, network ports and addresses, the actual remote connecting may be initiated.

3.1.3.2.2 Connecting

The expert user is situated for instance at Mapvision offices for the duration of the entire update process. There he or she clicks on the connection opening button of the VPN software and enters the connection data in the appropriate fields. After a short delay the VPN program informs the expert user of a successfully opened connection. Then the user may open the Remote Desktop Connections utility delivered along with Windows XP. The user again enters relevant connection data and the software forms a remote desktop connection operating on top of the VPN connection.

3.1.3.2.3 Using the connection

Having the active pair of piled connections the expert user is now at a position to replace the local user by conducting his or her role of copying the update package file onto the target customer computer over the remote desktop connection.

Finally the underlying VPN connection is ended by the user and thus the remote update process comes to a stop. The expert user can now open the returned

report file and evaluate the successfulness of the update process, the content of which does not lie within the scope of this paper.

3.1.4 Concept definition

At first an executable update package file is generated at Mapvision office using the packaging program developed during this project. The Mapvision staff user adds all wanted update operations using a graphical Windows interface. The main update work is divided to complete files and individual parameters inside files. Each of these is divided to adding a file, removing a file, and replacing a file, and the same three for parameters. The user may also activate the environment checks to help ensure the correct file will be executed in the correct customer computer.

Next the package file is sent to the customer using email or a remote desktop software over a VPN connection. The customer user then simply starts the received program on the target measurement computer and the program runs the entire update procedure without any input from the user. The user is then asked by the program to send the generated report file back to Mapvision by email. Update operations are logged into the report file for later inspection. In the case of the remote connection the customer user is not needed at all – the update package file is sent and executed by a Mapvision user and the report file is retrieved over the same connection.

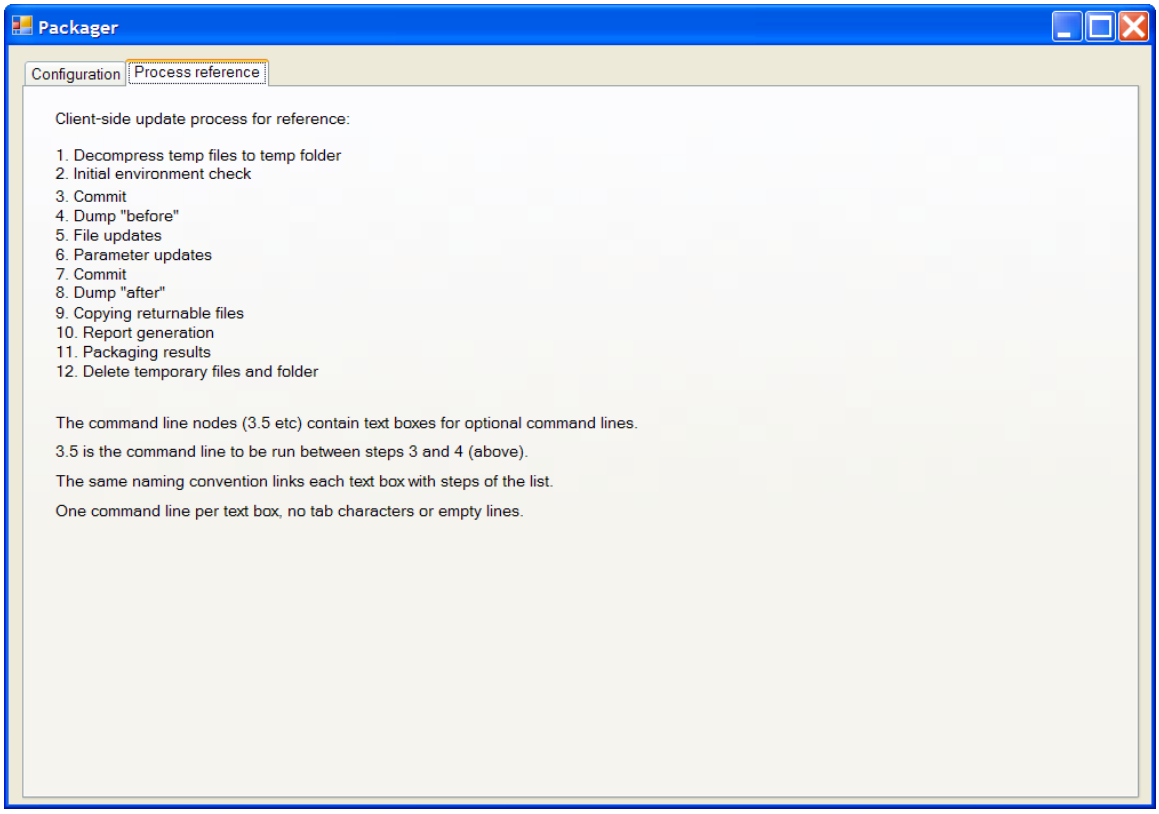

Image 7: The process reference tab selected. The tab displays the chronological progression of the client side of the update process.

By using appropriate functionality conditions studied by Murarka et. al. to analyze the inter-thread dependencies of the target software during the update sequence one could according to the writers ensure the avoidance of deadlocks. By this method a running multithread program could be updated without interrupting the running concurrent target programs. This would ensure the termination of the update program as well as the continued normal operation of the target programs as opposed to potential deadlocking threads caused by the update process. (Murarka, 2008)

3.1.5 Expert user group definition

The chosen expert user group consists of Mapvision staff relevant to software updates. This includes field sales personnel and office technical personnel, both of which are groups frequently setting up or executing manual software updates to customer measurement computers. They have sufficient technical knowledge to choose and replace individual files and parameters manually, as is done so far without this automated update system. Due to the low number of members in this group, all of them can also be specially trained to use the update configuration program at Mapvision offices.

A study suggests a measurable difference between usability preferences between Danish and Chinese users: Danish users valued effectiveness, efficiency and lack of frustration more than Chinese users. Chinese users reacted more to visual appearance, satisfaction and fun than their Danish counterparts (Frandsen-Thorlacius, 2009). The statistical differences of preference between domestic and foreign user groups might influence design decisions. However the available data provides information only on differences between Danish and Chinese users. Meaningful application of this data would not appear reliable.

3.1.6 Local user group definition

The chosen local user group consists of customer technical personnel. The main goal and motivation for this group using the system is to achieve a quick success in the update process with minimal time and effort diverted out of the usual day to day responsibilities. There is no reason to expect the local customer staff to have any special interest in the update process and thus the safest assumption is to expect them to consider it a somewhat alien product seemingly unconnected to usual work tasks. This group must be expected to have little to no understanding

of the inner functionality relevant to updating the measurement software. This is the reason to use the local users mainly only for starting the update program file. They are deliberately kept out of the inner functionality of the update process to avoid human errors and to reduce the need to train personnel. They however are expected to possess basic Windows using knowledge, more specifically moving the update program file to the target computer, starting it and sending the report file by email to Mapvision. This should not be a problem for almost any technical staff.

3.1.7 Expert interface building

The core functionality of the update process if the ability to perform file additions, file deletions, file replacements, parameter additions, parameter deletions and parameter replacements. The expert interface has separate lists for all of the above operations. Also the individual environment checks are activated and deactivated here, and their expected correct values are defined. This piece of the software provides maximal access to all settings of an update process. An expert user must have quick access to any supported features without having to manually edit files in any memorized way. This access is only found in the expert user interface.

3.1.8 Preventing users from updating the wrong computer

From discussions with the probable future expert users of the update system, it has been found that accidentally updating the wrong measurement computer - i.e. using the wrong update package - has happened quite easily. This causes confusion, a need for trouble shooting and extra work load when it happens. Thus this is a very undesirable scenario. To address this need the update system now has a built-in safeguard to help ensure that the running update package is

intended for the computer running it. This is called the environment check and it is contained in every update package.

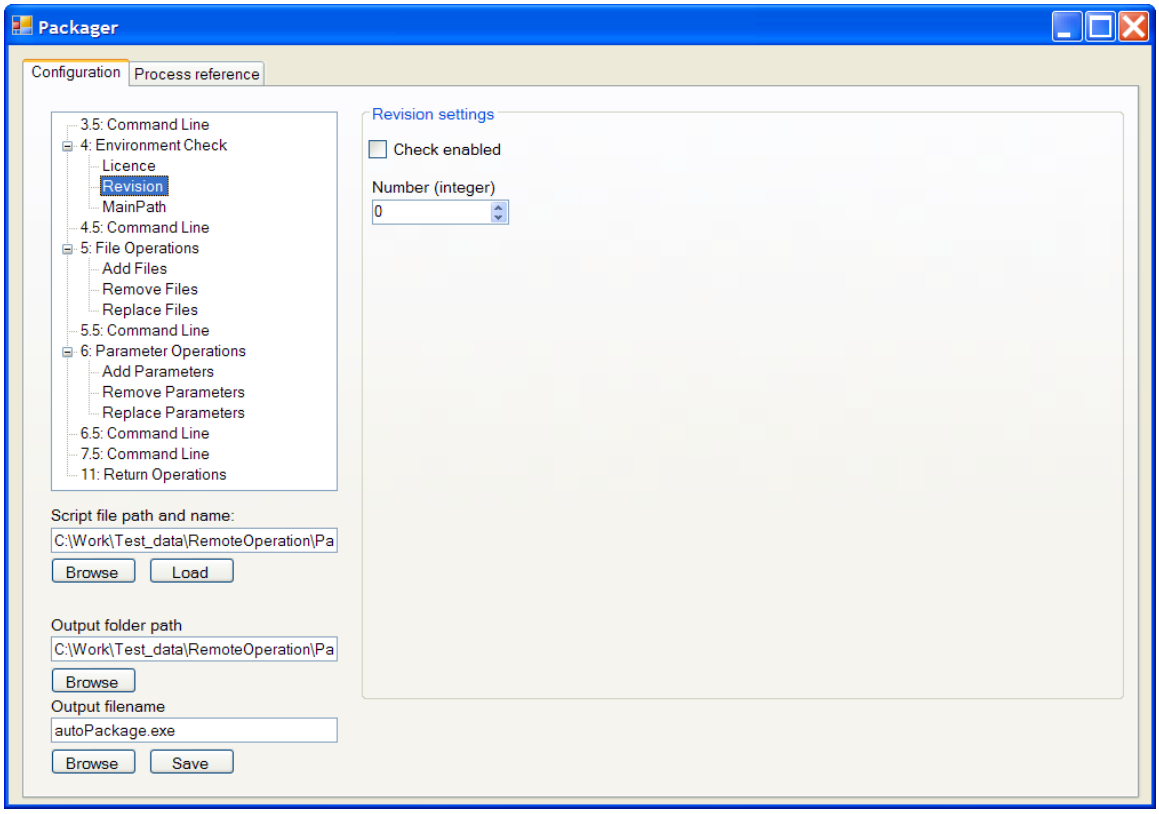

Image 8: Environment check branch of the settings tree at the left side of the window. Revision check settings at the right side of the window.

One of the important factors to be checked for would be whether or not the target software is still up and running or has it been properly shut down for the duration of the update procedure. This matter was not chosen as one of the environment check targets. It is assumed that either the customer's local technical staff is shutting the target software down or Mapvision staff shuts it down remotely if the customer has chosen to allow a remote connection all the way to the target measurement computer. This question however easy to take care off could still

be evaded by integrating appropriate software components into both the update software and the target software. For instance by applying a software system called dynamic C++ classes one could allow for controlled run-time updates of C++ software. (Myrén, 2001) Such techniques do not come free from the development time consumption point of view however. Lacking a sufficiently disruptive problem this solution newer came to a phase of implementation.

3.1.9 Addition, deletion and replacement of files

The file operations are the most influential part of an update process. Parameters may be used for example to store camera calibration data. While also parameters change the behavior of the measurement process the file operations may override the files containing the parameters. An expert user can exercise this control by choosing which files to add, delete or replace. He or she will also provide the path to the new file for addition and replacement. The user may choose to write the path with the file name, to copy and paste it from another program or press the supplied “browse” button to click the target file with a mouse.

3.1.10 Addition, deletion and replacement of parameters

The expert user can choose which parameters to add, delete or replace. He or she will also provide a new value for parameter addition and replacement.

3.1.11 Local interface building

Local interface design is guided by a reach for extreme simplicity to eliminate any realistic possibility for the local user to influence the outcome of the update procedure. This is performed by allowing the local user no control beyond the

choice to start the update program. The user is shown progress data during the update process and a completion notice after it, but no input is taken from the user at any point.

3.1.12 Expert user group prototype testing

The test version of the expert setup system is planned to be tested by a sample of Mapvision technical and sales staff in the future. Outcomes of the tests are estimated by the success rate of using the system to create a functional update package. Implementation of this did not fit the time table of the thesis employment time.

3.1.12.1 Practicality of early prototypes

Early prototypes containing expert side automation were developed and demonstrated and received feedback to postpone the expert side automation to a future version due to time constraints and currently unclear technical solutions. The expert side automation was removed for this version, and prototyping the user interface was decided to be executed at the time of completion of a functional test version of the core technical solution as the user interface was considered to be quick to change when needed when the underlying update functionality was at an operational state.

3.1.13 Local user group prototype testing

The local user group is typically located far outside Finland. For this reason it was decided that a controlled experiment was better executed at Mapvision office by using office personnel to simulate customer staff. To get more accurate results it would be preferable to eventually conduct studies with actual customer

staff in customer environments. People not familiar with the measurement system should be chosen in an effort to avoid a bias of expertise. Also this experiment did not fit into the time table of the thesis work employment and should thus be carried out in the future.

3.1.14 Remote connection testing

The VPN connection was tested first between two computers in Mapvision office. This produced a working remote desktop connection and did fit into the time table of the thesis work employment. The next step should be connecting between a computer at a home internet connection and a computer at Mapvision office. The final test should be a connection between a computer in Mapvision office and a real customer computer in actual customer premises.

3.1.15 Manual writing

The manual will be written for Mapvision use only, as the customer users are instructed to start the update application each time an update is needed. Necessary instructions are provided by the update program at run time. Local customer users are not expected to remember anything about the update system at the time of the next update operation. Recognition is preferred over recall. (Nielsen, 1994)

3.2 Applied heuristics

This section discusses the practical application of the aforementioned heuristic evaluation strategies. Both sets of heuristic models are applied separately to the developed system.

3.2.1 Applied Nielsen's heuristics

- Visibility of system status

The packager program displays all selected options as they are chosen by the user. This is however limited to the user selected sub set of options due to screen size limitations.

The patcher program updates process status information onto the screen in real time as events take place.

- Match between system and the real world

Packager options are organized and named after prevailing technical terms. These are taken from currently used production software. This is the one which current users are familiar with.

Patcher announcements are worded in simplistic, relatively non-technical phrases. The need for this difference between use of language in the office software and factory software comes from the steep gap of technical understanding of the product. Factory users are expected to be much less familiar with it.

- User control and freedom

All packager options are directly and individually erasable and changeable. These reversing controls are placed next to them in the graphical user interface. The update package itself can be re-opened for editing once it has been ordered saved by the user.

Patcher doesn't support any direct undo or redo functions. The factory side non-expert user is intentionally restricted to running pre-scripted update packages. The factory side user is not allowed to make any update operations on his or her own discretion.

- Consistency and standards

Packager offers the user selected options as radio buttons and editable lists as expected from a Windows based configuration application. When the user is happy about his or her choices he or she clicks the packaging button and is informed of the results of the final packaging operation.

Patcher functions in the manner expected from everyday installation and patching programs. It displays actions taken by the patching program and their results in real time. Sub-topics of the patching process are divided into separate windows which are opened and then closed as they become relevant and obsolete in the patching process.

- Error prevention

The state of the customer side computer is unknown to the expert side application. This prevents the recognition of error prone actions. The user cannot be warned or prohibited from making mistakes in the packaging phase.

The customer side application checks that the preset environment check variables match those in the system running the update patch. This is intended to reduce the chance of accidentally updating the wrong computer.

- Recognition rather than recall

The packager user is not expected to enter any information into sequential dialogs. All input is taken before any dialogs are displayed.

The patcher user is not allowed to enter any input in any phase of the update sequence. Thus no information is expected to be carried by the user from one dialog to a subsequent dialog.

- Flexibility and efficiency of use

One does not expect to see frequently recurring file replacements with identical file names or other similar update trends which would offer potential accelerators

for frequent users. However a packager user may still configure and save an update package in any frequently used basic configuration for future opening use should such a trend emerge.

Efficiency of patcher use appears sufficient and flexibility an irrelevant concept given that patcher usage is composed of running a single program file and giving the single result file back to the sender.

- Aesthetic and minimalist design

The packager graphical interface offers the conventional names of the options to be chosen. The user is informed of the completion of the packaging process. Any other information is provided in the separate manual.

The user of the patcher application is informed of the individual steps in the update sequence. This information is visualized by updating relevant event lists in appropriate windows. Finally the user is asked to return the result file to the sender of the patcher program file. This target person is the Mapvision employee who provided the file in the first place.

- Help users recognize, diagnose, and recover from errors

The packager application doesn't use error codes. Any information provided is displayed in plain language.

The patcher application doesn't expect the customer user to respond to errors. Any error situations are reported to a separate log which is returned to Mapvision for later analyzis. No error codes are used here either.

- Help and documentation

Any searching within the packager documentation is done by the users word processor of choice. This documentation specifically describes typical tasks to be carried out by the office user. The documentation focuses on the planned tasks targeted at the Mapvision staff user. The documentation lists explicitly described

steps for the user to carry out. This process of completing the steps is aimed directly at completing the planned tasks with the developed software. The supplied documentation is written to stay within a practical length that can be read by the user when performing the tasks. It is intended as a help material that is read fast enough to serve as a reference when needed. No separate training classes are expected to be needed.

Patcher does not come with separate documentation. The customer side user is instructed by a Mapvision contact person to run the supplied patcher program on the target computer and return the resulting file. Separate written documentation for this purpose is considered redundant. One could include a simple help file with the patcher program but this might also act as a distraction.

3.2.2 **Applied** Gerhardt-Powals' cognitive engineering principles

- Automate unwanted workload

The packager program doesn't expect the user to make other input than choose the settings, parameters and input files for the update package. This does not require mental calculations, estimations or other such easily automated cognitive tasks. A potential candidate for future versions is automating the input of automatically detected software environment changes into the update package. This would eliminate much of the manual configuration process in the packager program.

The overwhelming majority of tasks associated with the patcher program has been automated. All steps excluding the manual starting of the patcher program and manual returning of the result file have been automated. The main goal of this practice is however not reducing customer user's workload but eliminating

human errors when inputting data at the target computer.

- Reduce uncertainty

The data displayed by packager changes when the user changes it. Most of the time the user sees the same text he or she has entered manually. The rest of the data is composed of the multiple choice options also directly selected by the user. Thus the presentation of the relevant data would appear to be in a simple and clear form for the same user.

Patcher displays progress data in terms of accomplished or failed update steps. The steps are displayed in text form as lists. The lists are updated in real time as the update sequence proceeds to the next step. This approach is chosen as a natural way of informing the user that the update process is still running and a potentially present expert user may follow the progress of any points of interest. An alternative approach might incorporate some manner of graphical visualization of the update process. This was not deemed necessary nor an economical use of development resources.

- Fuse data

Any data to fuse would consist of file operations or parameter operations. They are fused in the sense of grouping them to additions, deletions and replacements. Further fusion of file and parameter operations into meaningful groups would require high level modeling of the target software system. Such functionality was not implemented nor considered productive in this case.

The data displayed by patcher is fused into relevant separate windows by topic. This is also a typical solution in commercial update software.

- Present new information with meaningful aids to interpretation

Data and terms presented by packager is known by the expert user. For instance setting files and parameters is a concept understood by the user in the first place.

Otherwise he or she would not know what the purpose of the developed application is.

The goal of patcher displaying progress data to a customer user is showing that the update process is still underway and not for instance complete or hasn't crashed. Therefore the user is not required to understand the progress data in any detail. A large portion of it is still probably understood however as most of the terms used a familiar from a general purpose Windows environment. Again any possibly present expert user is already familiar with the technical terms used. Otherwise he or she would not understand the work to be carried out with or without the terms used.

- Use names that are conceptually related to function

Names or terms used by packager are selected to match their counterparts in the environment outside the update process. Any renaming of concepts is avoided.

Packager doesn't group the package data in the temporal order in which the data has been entered. All packager data is grouped by it's function. For example external command lines in a particular time slot of the update sequence and file additions are grouped in two distinct lists. Inside these lists the user added actions are listed in the entered order.

Patcher uses the same terms as applied in packager. Any considerable change in naming conventions would not only be counter-productive for most users but a separate redundant authoring operation.

Patcher groups displayed data in a chronological order. This is chosen to present the real time progress of the update process. Inside the chronological lists the individual update actions are grouped into functionally related sets. An example of this practice is displaying all parameter replacements as a single continuous lists within the main action list. This is a consequence of displaying individual update actions as they actually occur in the target computer running the update

program in question.

- Limit data-driven tasks

Packager groups raw data by function. This is expected to help the user identify types of entries used. Coloring the data entries in a useful way would be difficult as they are already grouped by the same criteria that the coloring would use – their function. Coloring the repeating functional keywords in the data might be helpful though.

Patcher handles data in the same manner as packager. Coloring might be practical for example as a separation between different parts of the update process. For instance the displayed update events could be colored differently according to their category. For example file deletion event entries could be displayed in a different color from parameter additions. Another possibility would be coloring the update event texts according to the outcome. Such as coloring successful actions green with failed actions presented in a red font.

The update progress visualization of patcher is text based. Graphics could be used to make events easier to absorb, especially for the less technical customer side users. However producing and displaying graphics was not considered a desirable time investment within the development activities. At least this was the case with this first version of this update software.

- Include in the displays only that information needed by the user at a given time

The patcher real time update event announcement lists are somewhat unnecessary for most customer users. They usually do not need to know the specific steps in the patching process. They are however chosen to be displayed. This provides the customer user a feedback that tells him or her that the patching process is actually making progress in discreet visualized steps. Otherwise the user might easily begin suspecting whether the process has crashed or slowed down to defectively low rate of progress.

For an occasionally present Mapvision side expert user the patcher progress data does actually show whether the update process is making expected progress. Otherwise he or she may choose to commence debugging actions to find what has caused any unexpected progress data.

- Provide multiple coding of data when appropriate

Packager uses one set of terms for needed concepts. Most of the terms or names used are direct copies of the names already in use in manual updates. A future release of the packaging software may add alternative names or descriptions for the same concepts if it is considered useful. For example the environment check terms could be described in more detail as this is a concept not familiar from manual updating practices.

Patcher applies the same technical terminology as packager. As the technical goals to be achieved are the same there is little need for separate terms.

- Practice judicious redundancy

Packager displays for example file deletions or parameter replacements only in their respective lists. Displaying them in other context than their own lists has not been needed for the functionality of the software. For example individual file or parameter operations are not displayed in any dialog boxes or status reports. Thus a conflict of choosing between displaying the single operation and displaying the entire list of functionally related operations does not appear to arise in this application.

Patcher functions much in the same sense as packager also in this respect. The primary difference is that patcher displays the file and parameter operations and external command lines combined into the same list. The same situation with packager about the lack of the aforementioned conflict also stands here.

3.2.3 Solution evaluation

Most of the heuristic evaluation requirements were met in the chosen technical solutions – that is the two programs developed. Further usability evaluation might reveal more potential usability issues which could be met with potential solutions. This software is however designed from the ground up to a very small and selected group of users. Every user is always individually instructed to use the software. The emergence of a new user is an infrequent event. The product is not released to any kind of mass market or other groups of untrained users. Also the time constraints of the project were met and thus no further evaluation operations such as user tests are possible.

3.3 Alternate methods

This part discusses general purpose user centric design methods. In addition some notable attempted applications and technical solutions are described. These belong to a group that was not considered relevant to the final implementation stage of this development project.

3.3.1 Theory

3.3.1.1 Usability Testing

Usability testing was considered a useful method for evaluating the usability of the resulting update system. Unfortunately due to time constraints imposed by a tight schedule this method was not implemented.

Usability testing is a usability evaluation method which specifically uses actual

potential users. This is a clear difference from usability inspection methods which use usability experts to evaluate the system. The users are selected to represent the different user profiles deemed relevant to the product. As such the user sample's success at using the system is an excellent point to use as reference when judging usability.

A usability test would be planned around a set of tasks. The users would be instructed to carry out these tasks with the proposed system. The success of the users would be judged based on some predefined criteria. This could be time spent, ratio of successfully completed tasks to all given tasks, subjective satisfaction etc. The results would be a set of numbers - for example “80 % of given tasks successfully completed”.

3.3.1.2 Task Analysis

A task analysis is a strategy for modeling a given task by describing many of it's attributes explicitly and structuring it from smaller sub units – that is psychological and mechanical actions. The attributes described may be step durations, task frequency, task complexity, changes in the environment, equipment or any other relevant observations.

The resulting model may take the form of a hierarchical presentation with attributes and sub tasks forming nested structures. Results gathered from a task analysis may be applied for example to process improvement, automation, training or equipment selection. (task analysis, n.d.)

3.3.1.3 Think aloud protocol

As the name suggests the think aloud protocol is designed so that test users

think aloud as they try to accomplish given tasks using the proposed system. The people acting as test users are instructed to always say aloud what they watch, think, try to do, feel.

As opposed to observing the outcome of the effort to complete a task, the researchers have the opportunity to observe the individual steps taken to reach the goal. Video and audio recording is considered useful for analyzing the test later on. The think aloud protocol is used in usability engineering, psychology and social sciences.

3.3.1.4 Contextual inquiry

A contextual inquiry would most practically be conducted at an early phase of a product development process. The reason for this is that a contextual inquiry is a learning strategy for understanding a user's habitual way of trying to reach some goal in a work activity or in other context. This information can then be incorporated into a product development plan.

This learning method may be described as an extreme short time master-apprentice relationship. The user is the master who already knows how he or she is usually performs the activities being studied. The usability researcher takes a passive by-stander role. The initiative should be kept at the hands of the master with the apprentice trying his or her best not to interfere with the process. Interviewing the user may be conducted during the task execution or after it.

3.3.2 Application

3.3.2.1 Expert side automation

Initial feedback from stakeholder representatives called for extensive automation of the expert side functionality. More specifically this concept was projected to incorporate version tracking of available potentially updateable files and automatic generation of an update package suggestion. This suggestion was then planned to be evaluated be an expert user who would accept or dismiss any or all individual file and parameter update operations. A functional prototype of this system was developed and as it was evaluated it was deemed mostly operational but too time consuming to be incorporated into the rest of the relevant systems in time to fit the accepted timeline for this entire update automation project. With this in mind the project organization decided to place the expert automation functionality outside the scope of this research and development project.

4 Experimental application

This part describes the practical execution of the earlier needs, observations and plans. Probably the most distinctive feature of the implemented software update system is it's ability to directly modify individual parameter values inside files. The goal is to prevent transferring the entire new file and deleting the entire old file if it is not strictly necessary. By applying this method any customer made changes in these files will be preserved through the software update process. The direct parameter update is done to files of a known Mapvision format. Aside from this solution the overarching scheme of the designed program set is quite straightforward. The packager program is used to prescribe a list of update

actions. The implementation of the patcher program then runs this previously prescribed list of actions at the client computer side. Besides the direct parameter operations these lists of actions are composed most importantly of file additions and deletions.

4.1 Offline update

Offline update refers to the action of producing an update package at Mapvision offices or elsewhere by an expert user and then sending the update package to the customer staff typically by email and having them run the package at the target computer. After this the update report is returned to Mapvision by email by the customer staff.

Here an offline update as a term refers to an offline operation in relation to a telecommunication link. That is, an offline update does not use a telecommunication link. This is differentiated from the use of the term “offline update” meaning stopping the target process for the duration of the update. In this sense all update operation in this project would be “offline updates”. An online update in the sense of keeping the target software online would be possible amongst other ways by adapting the flow of data types from one running process to another. By using special software tools the update software would be able to keep the data flow between different running processes functional even when the data definitions change during the update. (Yajnik, 1997)

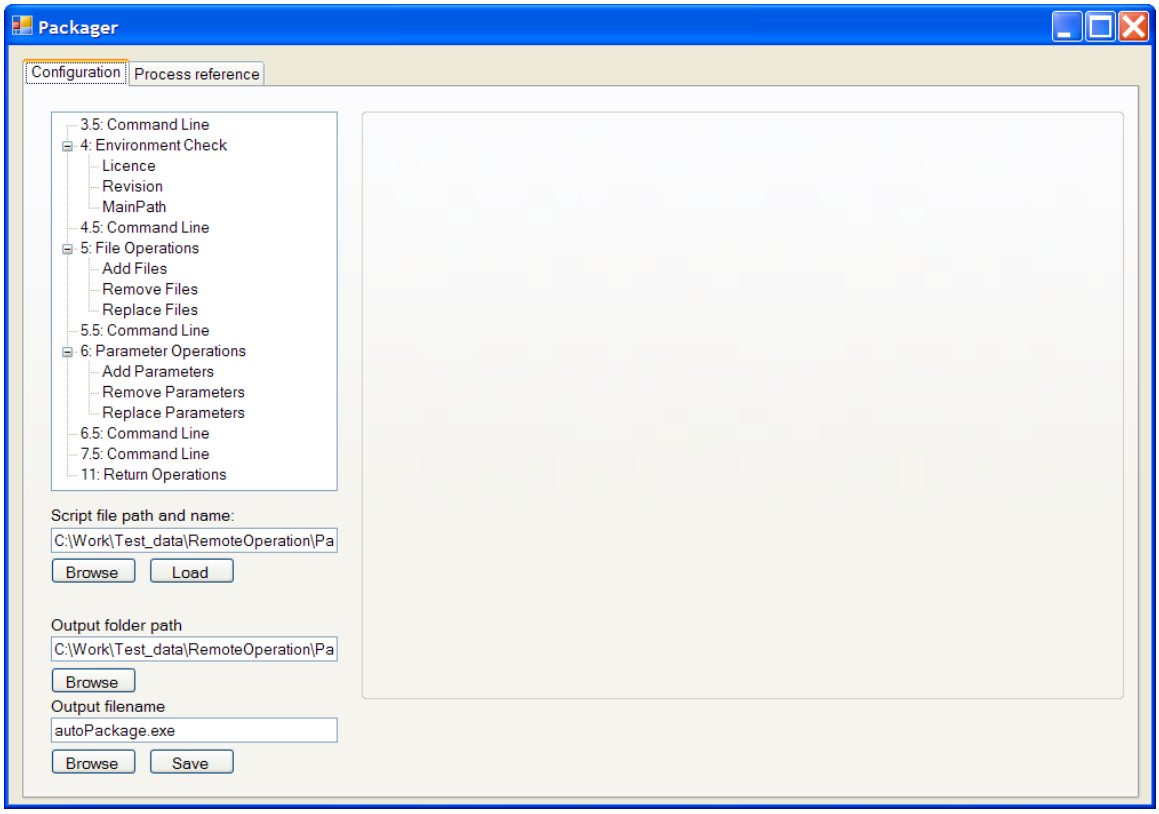

Image 9: The main screen of the Packager configuration interface intended for the purpose of producing and optionally later editing existing update package files. Such a file is then sent to a customer representative to be run at the targeted measurement computer.

4.1.1 File list

Files to be updated are stored in a data file divided into three lists by the operation type. Consequently file additions, deletions and replacements are executed as separate sequences. These three sequences always occur in the same mutual order. Thus one cannot mix parts of these lists to take place in an arbitrary order in time. As software update operations don't typically require files to be added, removed or replaced in any particular order this is not expected to

be any type of a problem.

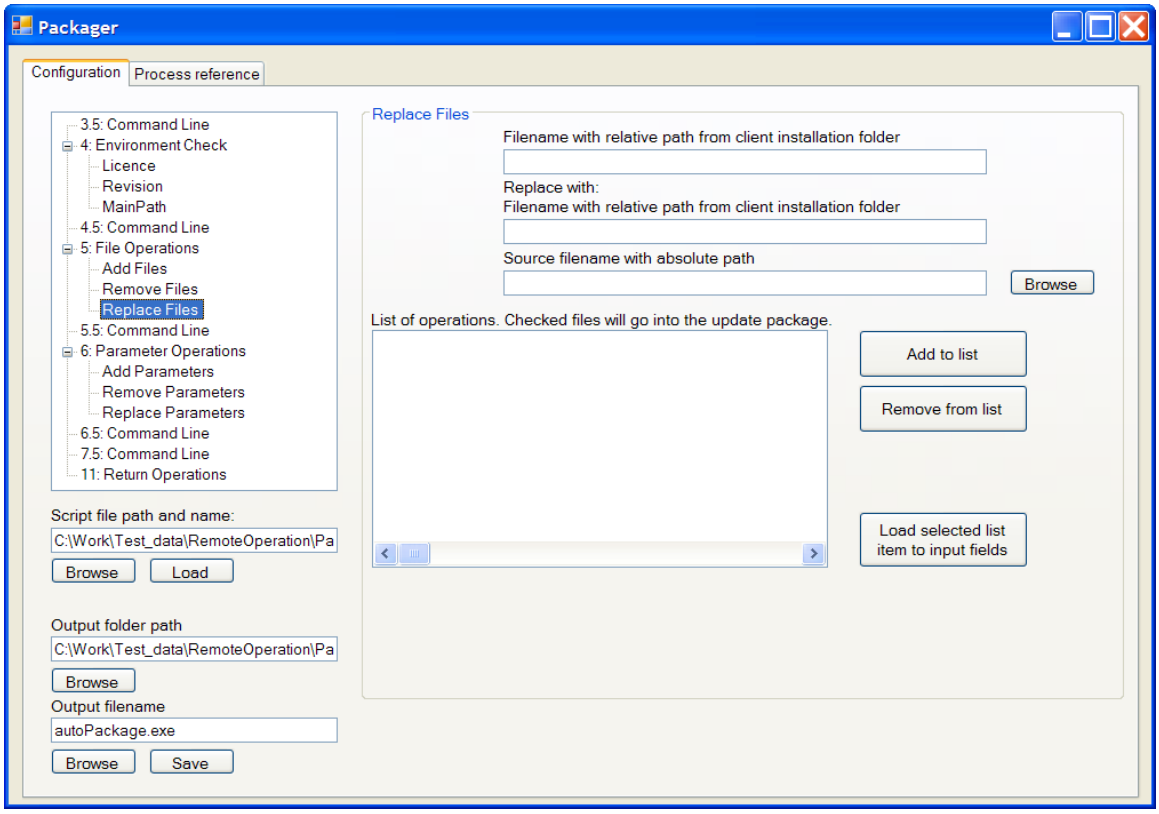

Image 10: File manipulation branch of the main settings tree user interface at the left side of the window. File replacements at the right side of the window.

4.1.2 Parameter list

Parameters to be updated are stored in a data file divided into three lists by the operation type. Thus parameter addition, deletions and replacement are performed as separate sequences like their file operation counterparts. The same logic applies as with file operations – parameters can be updated in any mutual order without causing any errors in the result of the update.

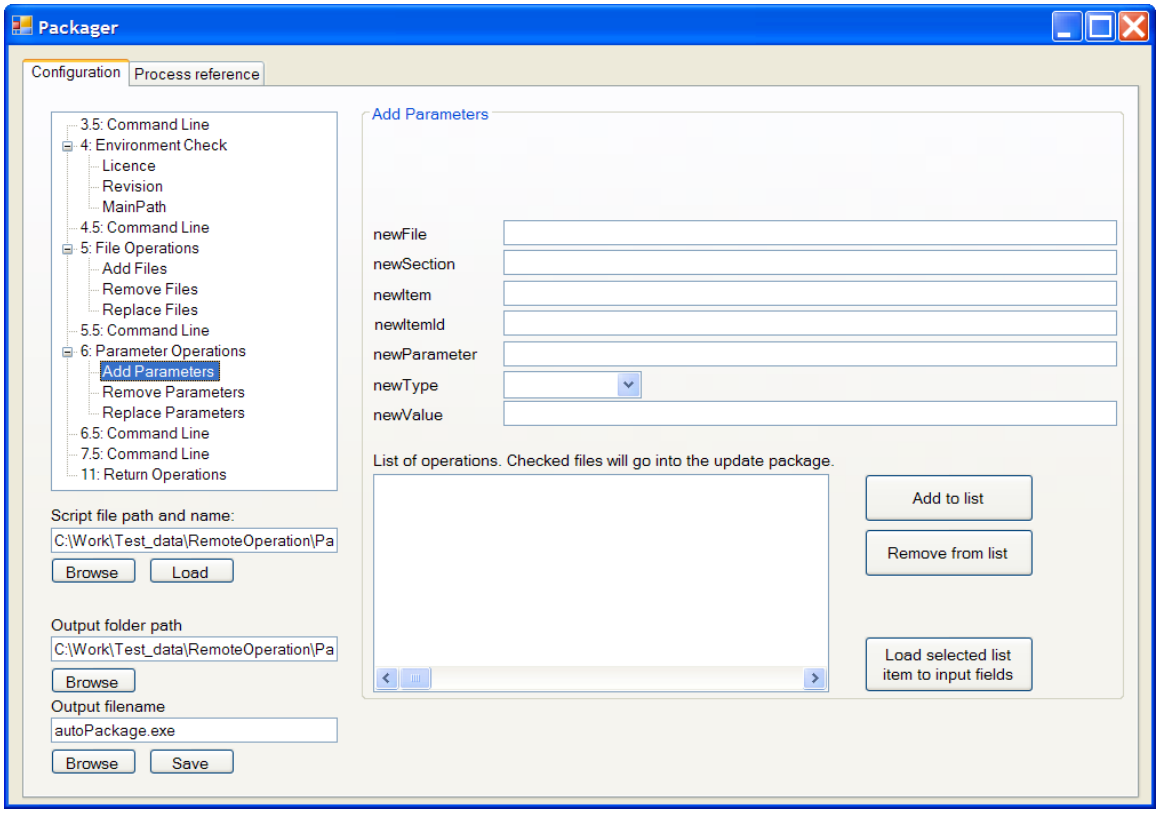

Image 11: The parameter operations branch of the main settings tree at the left side of the window. Parameter additions at the right side of the window.

4.1.3 Environment test programming

The customer environment – the measurement computer file system – can be tested in three ways. The measurement software installation main path is confirmed to exist. The software's version is checked. Also the measurement software's license is checked for validity. All of these read values are compared to the expected values defined at Mapvision office at update setup time. If any of the chosen checks fails, the update package is assumed to be in the wrong computer and the update process is not run.

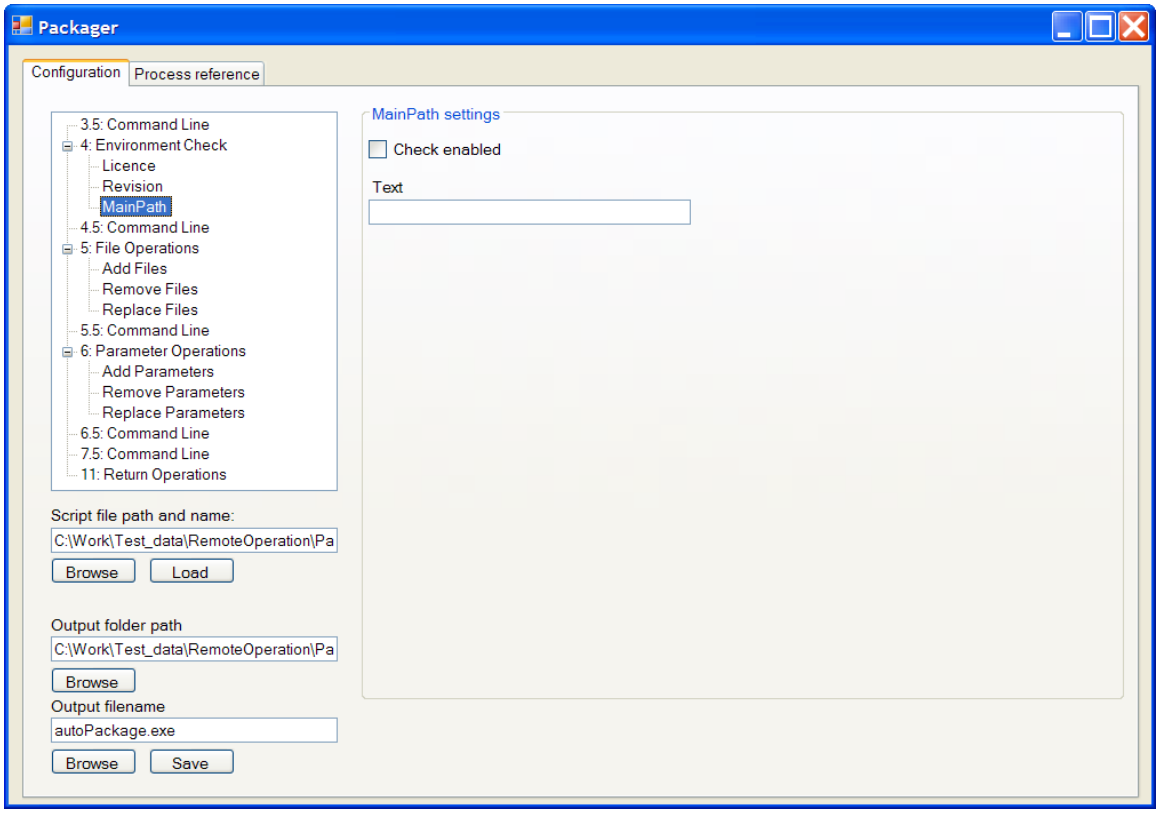

Image 12: Environment check branch of the main settings three at the left side of the window. Main path settings at the right side of the window.

4.1.4 File writing programming

The names and locations of files to be added, deleted and replaced are saved at a specific data file which is included in the final update package. For additions and replacements also the files themselves are included in the package. The client side executable then reads these file instructions and performs corresponding file system manipulations by standard .NET Windows commands. Any unexpected file events are recorded in the update log. For example missing targets of file deletions are reported to the expert user opening the log.

4.1.5 Parameter writing programming

The names of parameters and their location by the containing file and location inside this file are recorded at the data file included in the final update package. This is implemented the same way the file data is. In contrast to file operations, the parameter values are included in the data file instead of the new files in file operations. The client side executable reads these parameter instructions and calls corresponding parameter management operations from included in-house developed software.

4.1.6 Report generation

As the client side part of the update process runs, a text file is composed to report all actions taken by the update program. Also the success or failure of each individual action is recorded for later analysis by Mapvision staff. This is expected to confirm the success of all parts of an update operation. Alternatively reports of any failed parts of an update are expected to help in an investigation to uncovering the cause of a failed update.

4.2 Online update

An online update consists of all the parts of the offline update, performed by Mapvision staff remotely over a VPN connection. A remote update consist of network related tasks followed by a standard offline update over the network connection. The offline part of an online update is essentially the same process as a standalone offline update. The same tasks are involved and they take place over the network connection by using the remote desktop system.

4.2.1 Network settings

Some ports and connection protocols must be set to the allowed mode by using Windows settings in both connecting computers and also by the network component in both the Mapvision end and the customer end. This is a required part of any online update as any denied port or missing protocol at the route would block the connection. These case specific settings are expected to be negotiated with the customer technical staff over the phone or by email well prior to the actual update task. A typical set off ports are planned to be offered to customers who may then choose their own preferences if this is not readily acceptable to them.

4.2.2 Chosen connection program

The chosen connection program is a secure VPN application. This is provided by the close Mapvision associate company Ideal Engineering. Technical expertise is easily accessible and any support is very close when needed. Also the core functionality expected from the connecting program is provided.

4.2.3 Task sequence

The following steps are carried out by an expert user performing an online update remote from Mapvision offices:

1. Connection establishment

- The needed network settings must be negotiated with customer IT staff.

2. Sending an offline updater

- The entire update package file is sent over the remote desktop connection.

3. Running an offline updater

- The update file is executed exactly the same way as in the case of an offline update. It is simply started by using the remote desktop software.

4. Report retrieval

- Once the entire update process is complete – regardless of the success or failure of any of it's parts – a report file is transferred back to Mapvision office over the same remote desktop connection.

5. Connection closing

- Once the report file is securely retrieved, the connection may be closed and the entire update process is complete.

4.3 Potential future improvements

There was user interest to have the packager input automated. Currently typical expert users in Mapvision headquarters make manual changes to a local copy of the customer measuring software. Then the changed file names or parameters are input manually into packager as file operations or parameter operations. The changes are done to correct any problems encountered by the user or to add new measured products for example. In other words in the proposed concept the software would watch the relevant folder structure for manual file changes during all upgrade operations in Mapvision headquarters. When the expert user was done with the manual changes he or she would click the software to list all changes found into lists in packager. These would then be used as default settings for this particular instance of the packager software. The user would then be able to leave all the changes as they were detected by packager or remove or edit any of them.

A potential future upgrade could be a graphical and auditory presentation of the customer side update process progress to make the visualization more approachable. This might be especially compatible with the customer side users as they have little knowledge of the meaning of the displayed text based status information. The productivity of this investment of work time is however debatable as the customer side user is not expected to actually do anything with the progress information he or she received during the patching procedures. This upgrade might however improve the subjective feeling towards the product.

As the product is targeted specifically towards updating a certain industrial production software it may become desirable to implement some method of running the update while the target software remains running. This would reduce the average downtime of the system by eliminating the systematic need to shut down the target software for the duration of the update. One can duplicate an executable target software into two identical programs and modify it so that it runs normally with or without it's redundant copy. Thus one can shut down and restart the copies one at a time for updating purposes. This way the system remains in active service during an update operation. (Ssu, 2000)

System / Property	This thesis	Windows update	The Murarka method
Planned primary update target	Individual PC in a factory	End-user PC s globally	Critical system
File update strategy	entire files	diff files	None found
Parameter update feature	Yes	None known	None
Update strategy of running programs	None	None known	Thread analysis

Image 13: (A). State of the art in software updating compared vs. this thesis work.

Property	System	The Wei method	The Ssu method
Planned primary update target		Client-server applications	E-commerce etc.
File update strategy		None found	None found
Parameter update feature		None	None
Update strategy of running programs		Redundant server	2 replicated programs

Image 14: (B). State of the art in software updating compared vs. this thesis work.

Property	System	The Myrén method	The Yajnik method
Planned primary update target		Telecom system	Cluster systems
File update strategy		None found	None found
Parameter update feature		None	None
Update strategy of running programs		Dynamic C++ classes	Compiler and library

Image 15: (C). State of the art in software updating compared vs. this thesis work.

5 Discussion

The main research questions for this project were reducing the time spent by a customer waiting for a software update, reducing human errors in the environment of a noisy and distracting factory floor and reducing the money

spent by Mapvision to perform a software update. In the course of this project these questions have received functional answers that will be discussed in this section.

The time spent by the customer has been reduced by removing Mapvision personnel from the customer work site. With the new software it is no longer necessary to wait for the arrival of a human updater to the customer location. Earlier the arrival often took several days by plane, car or other means of human transportation. With the new system an update file package is sent by email or other means practically instantaneously as compared to the previous solution.

Human errors occurring as a consequence of the loud and distracting factory environment have in all realistic likelihood been mostly eliminated, as the tasks remaining to be accomplished at the factory are reduced to extreme simplicity being composed of running an executable file and copying the result file back.

The money spent by Mapvision in the course of a single software update targeted to customer production facilities has been traditionally mainly the sum of travel costs and a few day's salary for the work by an individual Mapvision employee including the foreign work additions to the traveling employees salary. Now with the new system the travel costs are expected to be in most cases eliminated as Mapvision employees are no longer sent to most software update target sites. In some special cases the local presence of a Mapvision employee may be required if any surprising conditions arise from. Also the salaries should be lower than before, as only work in Mapvision offices is required without the time spent for traveling.

The developed system performed to expectations of Mapvision staff and is expected to enter installation and update service in the near future. With the need to save time in update operations emerging constantly the system is considered a welcomed solution to the problem.

Image List

6 References

[1] Nielsen, Jakob (1994). *Usability Engineering*. San Diego: Academic Press. pp. 115–148. ISBN 0-12-518406-9.

[2] Jokela, Timo (2005). Methods for quantitative usability requirements: a case study on the development of the user interface of a mobile phone. Jussi Koivumaa, Jani Pirkola, Petri Salminen, Niina Kantola

[3] Ssemugabi, Samuel (2007). A Comparative Study of Two Usability Evaluation Methods Using a Web-Based E-Learning Application. School of Information Technology Walter Sisulu University. Ruth de Villiers, School of Computing University of South Africa.

[4] Bak, Jakob Otkjær (2008). Obstacles to Usability Evaluation in Practice: A Survey of Software Development Organizations. TARGIT A/S, Aalborgvej 94, DK-9800 Hjørring Denmark, Kim Nguyen, Logica, Fredrik Bajers Vej 1, DK-9220 Aalborg East, Denmark, Peter Risgaard, EUCNORD, Hånbækvej 50, DK-9900 Frederikshavn, Denmark, Jan Stage, Aalborg University, Department of Computer Science DK-9220 Aalborg East, Denmark

[5] Ivory, Melody Y. (2001). The State of the Art in Automating Usability Evaluation of User Interfaces, University of California, Berkeley. Marti A. Hearst

[6] Hornbæk, Kasper (2005). Comparing Usability Problems and Redesign Proposals as Input to Practical Systems Development. Department of Computing, University of Copenhagen. Copenhagen. Denmark. Erik Frøkjær

[7] Frandsen-Thorlacius, Olaf (2009). Non-Universal Usability? A Survey of How Usability is Understood by Chinese and Danish Users. Department of Computer Science. University of Copenhagen. Copenhagen. Denmark. Morten Hertzum. Computer Science. Roskilde University. Roskilde. Denmark. Torkil Clemmensen. Copenhagen Business School. Department of Informatics. Denmark

[8] Hollingsed, Tasha (2007). Usability inspection methods after 15 years of research and practice. ACM New York, NY, USA. Novick, David G.

[9] Nielsen, Jacob (1995). Usability Inspection Methods. CHI Companion 95, Denver, Colorado, USA.

[10] Law, Lai-Chong (2002). Complementarity and Convergence of Heuristic Evaluation and Usability Test: A Case Study of UNIVERSAL Brokerage Platform. NordiChi. Denmark. Ebba Thora Hvannberg.

[11] Frøkjær, Erik. and Hornbæk, Kasper. (2008). Metaphors of Human Thinking for Usability Inspection and Design. ACM Trans. Comput.-Hum. Interact.

[12] Wei, Wen-Kang. (2003). Implementation of Nonstop Software Update for

Client-Server Applications. Proceedings of the 27th Annual International Computer Software and Applications Conference (COMPSAC'03).

[13] Ssu, Kuo-Feng. (2000). Online Non-stop Software Update Using Replicated Execution Blocks. IEEE Xplore.

[14] Murarka, Yogesh. (2008). Correctness of Request Executions in Online Updates of Concurrent Object Oriented Programs. 2008 15th Asia-Pacific Software Engineering Conference

[15] Yajnik, Shalini. (1997). STL: A Tool for On-line Software Update and Rejuvenation. Eighth International Symposium on Software Reliability Engineering (ISSRE '97).

[16] Myrén, Henrik. (2001). Run-Time Upgradable Software in a Large Real-Time Telecommunication System. IEEE Xplore.

[17] automation. (n.d.). *The American Heritage® Dictionary of the English Language, Fourth Edition*. Retrieved September 20, 2009, from Dictionary.com website: http://dictionary.reference.com/browse/Automation

[18] usability. (n.d.). *The Free On-line Dictionary of Computing*. Retrieved September 20, 2009, from Dictionary.com website: http://dictionary.reference.com/browse/usability

[19] update. (n.d.). Dictionary.com Unabridged (v 1.1). Retrieved September 21, 2009, from Dictionary.com website: http://dictionary.reference.com/browse/update

[20] benchmarking. (n.d.). The American Heritage® Dictionary of the English

Language, Fourth Edition. Retrieved September 21, 2009, from Dictionary.com website: http://dictionary.reference.com/browse/benchmarking

[21] state of the art. (n.d.). The American Heritage® Dictionary of the English Language, Fourth Edition. Retrieved September 22, 2009, from Dictionary.com website: http://dictionary.reference.com/browse/state of the art

[22] user interface. (n.d.). The Free On-line Dictionary of Computing. Retrieved September 22, 2009, from Dictionary.com website: http://dictionary.reference.com/browse/user interface

[23] interface. (n.d.). The American Heritage® Dictionary of the English Language, Fourth Edition. Retrieved September 22, 2009, from Dictionary.com website: http://dictionary.reference.com/browse/interface

[24] remote control. (n.d.). The American Heritage® Dictionary of the English Language, Fourth Edition. Retrieved September 22, 2009, from Dictionary.com website: http://dictionary.reference.com/browse/remote control

[25] usability testing. (n.d.) Wikipedia. Retrieved February 12, 2010. http://en.wikipedia.org/wiki/Usability_testing

[26] task analysis. (n.d.) Wikipedia. Retrieved February 12, 2010. http://en.wikipedia.org/wiki/Task_analysis

[27] Usability Professionals' Association. (n.d.) Usability Professionals' Association. Resources: Usability in the Real World. 140 N. Bloomingdale Road Bloomingdale, IL 60108-1017. Tel: +1.630.980.4997. Fax: +1.630.351.8490