Helsinki University of Technology

Department of Computer Science and Engineering

Antti Kangas

Usability and user centered methods for frequency converter software tools development

This thesis has been submitted for the partial fulfillment of the requirements of the degree of Master of Science in Computer Engineering.

Helsinki, on May 17th, 2005

Supervisor: Marko Nieminen, D.Sc.(Tech.)

Instructor: Kari Ranta, M.Sc.(Tech.), R&D Manager, ABB Oy

Abstract

HELSINKI UNIVERSITY OF TECHNOLOGY ABSTRACT OF MASTER'S THESIS

Author: Antti Kangas

Title of the thesis: Usability and user centered methods for frequency converter software tools development

Date: May 17th, 2005 Number of pages: 106, Appendices 25

Department: Department of Computer Science and Engineering

Professorship: T-111 Interactive Digital Media

Supervisor: Prof. Marko Nieminen, D.Sc.(Tech.)

Instructor: Kari Ranta, M.Sc.(Tech.)

There are many methods to build and ensure the usability of a product which is an important aspect of quality. The key methods are finding out and documenting the users' tasks, capabilities, and operating environment in addition to iterative usability evaluation. In most cases, these usability engineering practices can be added to traditional product development processes.

This research was done to find user centered methods that ABB Oy Drives tools team could utilize in developing software tools for frequency converter products. Also usability problems in the current software products and requirements emanating from the operating environment were desired to be found.

Different methods and techniques available were mapped out with a literature study. Some of them were tried out in practice to find the problems and requirements and to bring the users closer to the development process. The most significant methods were interviews, user observations, questionnaires, heuristics, and use cases.

Many methods for different situations were found in the literature. The ones tested in the field were shown to be useful by providing results that can be used to make better products. Information on users and their work was unveiled. Numerous problems in the current software tools were found, and requirements for new features were discovered. Introducing user centeredness to development activities was begun.

Keywords: usability, user centered, development, design, software, user interface, frequency converter, drive, end-user

Tiivistelmä

TEKNILLINEN KORKEAKOULU DIPLOMITYÖN TIIVISTELMÄ

Tekijä: Antti Kangas

Työn nimi: Usability and user centered methods for frequency converter software tools development

Päivämäärä: 17.05.2005 Sivumäärä: 106, liitteet 25

Osasto: Tietotekniikan osasto

Professuuri: T-111 Vuorovaikutteinen digitaalinen media

Työn valvoja: Prof. TkT Marko Nieminen

Työn ohjaaja: DI Kari Ranta

Käytettävyys on tärkeä laatuominaisuus tuotteessa. On monia tapoja todeta ja rakentaa tuote käytettäväksi. Tärkeimmät keinot ovat käyttäjien tehtävien, kykyjen ja toimintaympäristön selvittäminen ja dokumentointi sekä iteratiivinen käytettävyyden arviointi. Useimmiten nämä käytettävyystyön käytännöt voidaan liittää perinteisiin tuotekehitysprosesseihin.

Tällä tutkimuksella etsittiin käyttäjäkeskeisiä menetelmiä, joita ABB Oy Drivesin tools team voisi hyödyntää taajuusmuuttajatuotteiden työkaluohjelmien kehityksessä. Nykyisten ohjelmatuotteiden käytettävyysongelmia haluttiin selvittää sekä niiden toimintaympäristölähtöisiä vaatimuksia.

Eri menetelmiä kartoitettiin kirjallisuuskatsauksella. Joitakin niistä kokeiltiin käytännössä ongelmien ja vaatimusten selvittämiseksi sekä tuomaan käyttäjiä lähemmäs tuotekehitysprosessia. Tärkeimmät menetelmät olivat haastattelut, käyttäjien havainnointi, kyselyt, heuristiikat ja käyttötapauskuvaukset.

Lukuisia menetelmiä eri tilanteisiin löytyi. Kentällä kokeillut osoittautuivat hyödyllisiksi antamalla tuloksia, joita voidaan käyttää parantamaan tuotteita. Tietoa käyttäjistä ja heidän työstään saatiin selvitettyä. Nykyisistä työkaluohjelmista löytyi lukuisia ongelmia ja uusien toiminnallisuuksien vaatimuksia saatiin päivänvaloon. Käyttäjäkeskeisyyden esittely kehitystoimiin saatiin alkuun.

Avainsanat: käytettävyys, käyttäjäkeskeinen, kehitys, suunnittelu, ohjelmisto, käyttöliittymä, taajuusmuuttaja, käyttö, loppukäyttäjä

Preface

This Master's Thesis is done for ABB Oy Drives in Helsinki, Finland. I would like to thank Kari Ranta, the thesis instructor and tools team R&D manager, for providing the opportunity and resources necessary for completing this research, product manager Ilkka Tunkkari for showing the ropes, other tools team members and ABB employees for their cooperation, and professor Marko Nieminen, the thesis supervisor, for his guidance, feedback, and encouragement.

I would also like to thank my father for his continuous support throughout my studies.

Helsinki, May 17th 2005

Antti Kangas

Table of contents

1 Introduction

The tools people use in their work should be easy and comfortable to operate. They are everyday companions with which users must get along. They are subjects of heated coffee break discussions if they don't conform to users' expectations. Tools with bad usability add stress, reduce productivity and they require a lot of expensive training.

Tools implemented in computer software are amazingly complex and expensive to make. Yet, they are cheap to multiply and distribute all over the world. However, the people who use them are diverse in many ways. Their education, language, cultural and work context are different. How does one design a product to satisfy all needs and situations?

A folk wisdom in the IT industry says that the later in development a problem is caught, the more it costs to fix. Many problems in the delivered systems can be traced to problems in the requirements that were used in designing the systems. Guesses and half-truths about the users' needs have been made because no real user information was available, or methods for gathering it were not known.

No matter how technologically apt the designers and coders are, the software product they develop can be cumbersome and utterly unusable. This can happen, if the developers don't know or don't care what the users want and need for supporting their work. Negative experiences in one area can have ramifications for business in others as well, when a reputation of a bad apple in the bunch spreads.

A product doesn't become easy to use by itself. In global software product development and deployment environment the developers' own intuition is not enough. Products have to be designed with sound engineering methods. A user centered development process. Users and their interests should be included in all phases of product development: from requirements definition to design, testing, and follow-up afterwards.

"Products will succeed only if they facilitate users having successful first experiences, and only if they also allow for growth and learning and variety of patterns of use." [17]

Matthias Rauterberg lists [46] three major topics of problems in traditional non-user-oriented software development. A specification barrier deals with how the software developer can ascertain that the client is able to specify the requirements in a complete and accurate way that will not be modified during the project. The medium used by the client to formulate requirements should be formal and detailed for the developers to have it easier to implement the system. But that requires certain measure of expertise from the clients and they are not likely to be prepared to acquire it beforehand.

A communications barrier exists between customers, users, and the software developers. The developers' jargon is technical and the users speak in an application oriented manner. Non-technical facts can slip through the holes of this conceptual net, which restricts the social character of the technology to be functional and instrumental. Jointly created, preferably visual, perceptually shared contexts have to be created to improve communication.

The third problem is optimally designing a product with interactive properties for supporting the performance of work tasks. Technical aspects of software products have their algorithmic treatments. Non-technical context requires an increasing amount of attention with procedures of different nature. A single group of persons cannot have a complete, pertinent, and comprehensive view of the ideal work system to be set up. The requirements concretize within boundaries through iterative analysis, evaluation, and planning processes conducted together with the involved parties.

1.1 Goals for the thesis

The objective of this study was to provide a stepping stone for ABB Drives tools team to facilitate their moving towards a user centered development process as they develop and produce ABB's DriveWare® software products. The focus of their work is changing away from architecture and implementation issues towards designing and specification of software.

Code writing can be bought from outside, but it poses the challenge of communicating all necessary aspects by proper specifications. Another challenge is the division of work: what should be done internally, and what can be left to the implementers.

Currently, the tools team has very limited knowledge on the actual users of their products. Who they are, what they do with the software, and how, are valid questions that all product developers should be able to answer. In addition, the features of current products that users regard as good should not be forgotten.

The team manager was interested in getting directions and information on methods, best practices, and ideas that could be utilized in the software production and the products. How to recognize which steps are necessary to take when new features are required is a concern. Recommendations on abandoning old, inferior practices were also wanted.

To reach the objective, it was decided that goals for the study would be to find applicable methods to

- improve software usability
- pull users closer to the development process

Some of the methods would be tried out in practice so that their results would act as indicators of the methods' applicability and benefit to the development activities. With concrete results it will be easier for managers to decide on their further use. This thesis was not a part of any particular project of the tools team and the researcher was not employed by ABB, but interjections were made to ongoing development activities whenever an appropriate occasion was available.

Reading this thesis should also raise the development team members' awareness of issues of usability, user interface design, and user centered development. A personal goal for the researcher was to gain experience and develop his professionalism by applying some of the methods in real life situations. Also learning about the automation industry was seen as beneficial.

Ripple effects of this work could accommodate increased end-user productivity, enhanced design quality, promotion of cooperation in the development team, and lower costs for subsequent development and maintenance.

1.2 Structure of the thesis

This thesis is divided into seven sections. This first one outlines the general goals of the research. Second deals with the more specific research questions. Also the scope of the research is defined. Number three describes the context in which the research was carried out including corporate issues, electric drives, and the current ABB DriveWare® software tool products.

Section four dives into the literature for theories and practices of usability, user interfaces, and user centered methods. Number five tells which of these methods were selected to be used in this research, why, and how.

Part six contains the results of the research: what new information was uncovered, found usability problems, and new functionality concepts from user requirements. The last section, seven, draws everything together and assesses the success of this study.

2 Theme of research

2.1 Research questions in detail

Improving software usability and pulling users closer to the development process are very broad goals. An equally broad answer is to utilize a user centered development process. To narrow down the field, these two research questions were formulated.

1. Which user centered methods could ABB Drives tools team incorporate into their software development activities?

To make future products easy to use, gaining true knowledge on users is needed. What expectations do users have on the tools software of drives is an important point when designing new products as well as requirements emanating from the operating environment.

Not all methods or techniques are applicable in all development environments. Limitations rise from the education, background, work habits, and experience of the developers. Corporate environment by its vision and guidance have an impact, not to mention fiscal issues and customer involvement.

2. What problems do users have with current ABB DriveWare® software products?

Some current products have been criticized of being difficult to use. They also have a drastic lack of consistency in their look and feel. It would be beneficial to find out what specific usability problems there are and how they could be fixed or at least avoided in the future. The methods found in question one would be applied to answer question two.

A third research question was originally lingering in the air: what issues must be taken into account when designing user interfaces for multi-cultural professional users in the global market. However, available data and resources did not allow exploring this further than a skim over the surface.

2.2 Scope

The scope of this thesis was agreed to include:

- The latest versions of these DriveWare® software products: DriveWindow, DriveWindow Light, DriveDebug, DriveAP and DriveSize.
- User interface level; what is visible to regular users.
- Primary users (see chapter 4.5) performing selected tasks.
- Main emphasis on the operating environment of drive commissioning.
- New feature concepts and non-functional prototypes as time permits.
- Windows desktop or laptop PC running environment.
- The software production process, as it affects usability.

Outside of the scope would be:

- Hardware issues.

- Software architecture and module level implementation.
- Programming language specifics.
- Drives’ internal software and communication technology.
- Traveling outside Finland.

One narrowing factor could have been the application areas like the field of industry, specific user groups, or plant size, but the lack of existing user information prevented making this choice in the beginning.

3 Development and application environment

3.1 Summary

This section outlines the context of the part of automation industry where this thesis work is conducted. It tells what drives or frequency converters are, what is ABB's take on the area of PC software production, and how product development is carried out currently.

3.2 Corporate context

This thesis was made at tools team of ABB, one of the world's largest automation and power technology corporations with over 130000 employees in around 100 countries and 19 billion $US revenues in 2003 [2]. Head office of ABB is in Zürich, Switzerland. In Finland, there are about 6400 employees in nearly 40 locations with 1.3 billion euros revenue of which 7% goes to research and development.

In power technology, ABB makes and delivers transformers, distribution automation, switchgear, medium and high voltage apparatus, cables, entire power technology systems and services. Automation products include drives, electrical machines and motors, low voltage apparatus and switchgear, robotics and support services. Process automation, drives and electrifications are utilized in both industry and marine settings.

The corporate history begins in 1889 with the founding of Ab Gottfrid Strömberg Oy which evolved by acquisitions and mergers into the current ABB in 1988 when Asea and Brown Boveri (BBC) joined forces. ABB Drives business unit (BU) develops and manufactures frequency converters, also known as drives, that power and control electrical motors, and is the market leader. Alternate current (AC) drive development started at the corporation in the late 60s. The first AC drive, SAMI, was ready for market in 1976. The main drives factory with research and development facilities is in Helsinki, and other factories are located in Switzerland, the USA, China, India, and Germany.

The tools team in Drives is responsible for designing, developing and upholding PC software that is used in commissioning and servicing of drives. Also internal development tools software are made here. The tools team belongs to Drive Care unit under ProductAC division.

3.3 Drives and their uses

Devices known as drives are by a more complete name electric motor drives or electric drives [39, page 13]. They convert energy from the electricity supply network into mechanical movement that is utilized in a process. By definition, a drive consists of an electric motor, a converter, and their controls. Although on a factory floor, the word drive usually refers only to the converter and controls. The same custom is followed in this thesis.

The modern converters are called frequency converters. They use semiconductor components and digital control to supply power to electric motors and adjust their operation. Their job is to convert AC power from the electricity network into a form that is used to power an electric motor. Most new industrial motors use AC power nowadays. The network AC is often converted first into direct current (DC) inside the frequency converter before changing it back to variable amplitude and frequency AC for the motor. DC motor drives exist also.

Frequency converters can be utilized wherever there is a need for a rotating electric motor. In a production line, precise control improves the quality of the end-product [3]. Increasing capacity is possible without further investments. Adjustable speed saves energy, which decreases load on the environment. A soft start of rotating saves equipment from mechanical wear. Variable speed drives can eliminate the need for transmission mechanisms altogether, which reduces hardware investments and maintenance costs.

Application targets for frequency converters can be found in many places, like pumps, fans, compressors, conveyors, crushers, mills, mixers, extruders, lifts, cutters, cranes, and marine propulsion. Even wind power generators have been deployed. ABB drives can operate motors from low voltage 0,12 kW to medium voltage 30000 kW models. Digital control became mainstream in ABB drives around 1985. In 1995 a noteworthy step forward was taken with the introduction of direct torque control (DTC).

Traditionally, drives are operated via discrete I/O controls, a fixed control panel on the drive cabinet or shell, or with an elaborate automation system utilizing field buses. Drives have analog and digital input terminals for control and sensor signals. A parameter called reference value for motor rotation speed is entered, which the drive then pursues taking into account set acceleration and deceleration times.

The parameters are the principal means of configuring a drive. They are name-value pairs that are located in the memory of the drive. Non-modifiable parameters that give out values, like measured temperature, are called signals.

3.4 Current ABB products

Digital control allows more and more enhanced controls and automation to be built into the drives for various application areas. Commissioning, adjusting and monitoring these functionalities practically require computer applications. Interaction with only a limited control panel in each drive would not be efficient.

A single ABB drive connects to a PC via an RS port or with fiber optics to a DDCS bus adapter. This is called point to point communication. DDCS devices can also be chained together to form a network so that one PC can communicate with several drives without removing and connecting cables individually.

There are several drive types available from ABB. The ones with PC tools software connectivity are mainly from standard drive and industrial drive series (see Appendix B). The **standard drives** models ACS550 and ACH550, also known as “housewife drives”, are used in wide range of lighter duties in industries as well as heating, ventilation and air-conditioning (HVAC) setups. They connect to PC’s serial port with an adapter and a cable that’s plugged into the drive’s control panel slot. The control panel can’t be used simultaneously.

The **industrial drives** series has a wide range of ACS800 models in stand-alone single and multidrive configurations reaching up to 5600 kW of power. They are used in more demanding purposes than standard drives and in larger plants they often feature external controlling logics connected via field buses like Profibus. The fiber optic ports for DDCS communication are located inside the drive’s covers. The control panel interface features similar serial port communication capabilities as the standard drives.

ABB PC software tools run mainly on Microsoft platforms. There are some old DOS applications that are still maintained, but all modern software is made for 32-bit Windows NT and derivatives. Also some simple web-based utilities come from the tools team. The applications are “commercial off the shelf” (COTS) or “shrink wrapped” in nature, which means that they are not customized for singular users. All customers get the same product. See Appendix I for screenshots.

3.4.1 DriveWindow

DriveWindow is meant to be an easy-to-use start-up (commissioning) and maintenance PC software tool for end-customers and drive specialists who work with the industrial drives. It provides drive parameter handling, drive control, graphical presentation, logging functions, monitoring, and backup features for maintenance and service personnel. The current version is 2.12. The 2.x series is very different from 1.x due to complete refactoring for 32 bit platforms.

DriveWindow is comprised of two components: DriveOPC is a server that handles communication with the drives via the fast DDCS on fiber optics technology and provides a standard OPC interface which the DriveWindow user interface presents. The OPC Server can reside on a different PC, which is reached through a LAN or other network solution. The number of simultaneously monitored signals is six. They can be all from one drive or selected from many.

3.4.2 DriveWindow Light

DriveWindow Light is aimed mainly for the standard drives, like ACS550. It supports point-to-point serial port communication only. Industrial drives can be connected through their control panel interface. DriveWindow Light, currently at version 2.31, is also meant to be an easy-to-use start-up and maintenance tool, but it comes from an altogether different codebase than DriveWindow.

The application provides similar functionality as DriveWindow does with the exclusion of backup and restore of the drive’s internal software. Monitoring of four simultaneous signals is supported. There is a start-up wizard aimed to aid those not so familiar with drive technology.

3.4.3 DriveDebug

DriveDebug was born as a tool for ABB’s internal use, when DriveWindow 1.x was deemed as unusable and buggy by many people. It wasn’t meant to be for sale to customers, but when they have seen ABB personnel use it, they have wanted it too. DriveDebug provides more advanced functionality than DriveWindow, including macros, and access to the drive at a more rudimentary level. It communicates only with DDCS and fiber optics.

3.4.4 DriveAP

The current flagship of ABB industrial drives, ACS800, contains block programming capabilities that provide simple automation without external logic systems. The block programming can be done via the control panel of the drive by inputting one parameter at a time, or with the DriveAP PC application. DriveAP 1.x supports Adaptive Programs and DriveAP 2.x also Multi Block Programming Applications. Despite the version numbers, 1.x and 2.x are concurrent products.

Adaptive Programs form the automation logic in 15 configurable blocks, selectable from 20 alternatives. In Multi Block mode, the user can place over 200 blocks in three time levels and there is a wider selection of function and I/O blocks to choose from. DriveAP 2.x has two different interfaces for these programming methods.

DriveAP works as stand-alone, or with a drive connected with DDCS and fiber optics, in off-line and on-line modes. In off-line the program has to be transferred to and from the drive with specific commands. In on-line all changes are copied straight into the drive and signals can be monitored in real-time.

3.4.5 DriveSize

DriveSize is a stand-alone dimensioning PC software tool. It helps to choose an optimal configuration of electric motors, frequency converters, and transformers. DriveSize can also be used to compute network harmonics and to create documents about the dimensioning. It is said to be the most valuable salesman of these products. Therefore it was included in the questionnaire part of this research.

3.4.6 Demo cases

Demo cases are hard travel cases containing a frequency converter, an electric motor, and an external control panel for the drive’s analog and digital inputs. They are used widely at ABB and its customers for demonstration, education, and development purposes. These demo cases run on regular mains power, so they are easy to set up anywhere. A demo case containing an ACS601 is pictured on the right.

3.5 The tools team and product development

According to the handbook [1], the DriveWare® software tools are options and add-ons for the frequency converters made by ABB drive units. The tools team is a common task force for ABB Drives to make the tools available cost-effectively with high and consistent quality. The drive units are the key customers for the tools team and also a source for ideas and requests. Additional ideas come from customers, other ABB units, competitors and general IT development.

The mission of the tools team is to develop abilities in the focal competences of tools software with regard to the needs of ABB Drives and the DriveWare® implementations, to maintain relations to suppliers and universities, and to provide equally and cost-effectively high-quality DriveWare® implements to the product organizations.

The tools team is rather small, only six people, of whom two or three write actual code for select products and prototypes. Additional coding is outsourced. None of the team members have educational background in computer science or software engineering, but some have coding experience spanning decades. They are M.Sc.

university graduates in the field of electric engineering. The team works in a cubicle farm environment in a large open office, where meeting rooms are shared by all floor occupants.

The development of the frequency converters takes a long time and there is an established gate model and quality directions for that work. For the tools team software production, there is virtually none. Collaborative design methods are not utilized; people sit mostly alone at their desks. So far they have assumed that the target users are similar to themselves, so no special care has been given to the user issues. Only in DriveWindow Light there are the wizards intended for those less experienced, because concern for the level of user skills was raised during development in 2000-2001.

4 Usability, user interfaces and user centered methods

4.1 Summary

This section contains theories on and definitions of usability, user interfaces, usability engineering and different user centered methods found as a result of a literature study. The methods presented here are not likely nor possible to be applied as-is, but they are adaptable and offer the best potential for improving the development process at ABB Drives.

The user centered methods follow in major parts the ISO 13407 process. Every method is not described fully to avoid excess repetition, only their practical techniques and new contributions to the general framework are explored.

4.2 User interface principles

4.2.1 User interfaces in general

One definition of user interfaces is: the aspects of a computer system or program that can be perceived by the human user, and the commands and mechanisms used to control its operation and input data [53]. A user interface is the surface forming a common boundary between a computer system and a user. It is more than just buttons and menus, also the procedure of interaction and all media that pass information from one entity to the other are included.

Interaction with computers began with flicking switches, moving cables, reading punch cards and watching light bulbs go on and off [36, page 50]. User interfaces developed hand in hand with the generations of hardware technology. Also the user types changed along with the usage purpose of computers.

After machine language programmable batch systems came command languages with question-answer type of interaction via keyboard and full-screen textual menu-driven systems that used specialized function keys a lot. The graphical user interface with WIMP paradigm (Windows, Icons, Menus, and a Pointing device) is already over 30 years old.

Even though many aspects of WIMP build on the earlier techniques, one significant step of advance can be seen in the interaction model which is of Object-Action type instead of Action-Object evident especially in command line systems [50, page 61]. The Object-Action model allowed the evolution of direct manipulation.

4.2.2 Direct manipulation

Interfaces based on written commands burden users with a lot of device dependent syntactic details. The user must choose an action first and then type its command (perhaps multiple) and targets. Graphical user interfaces (GUIs) display and visually represent the users' task objects and actions. The operating system is no longer a strange intermediary, because the user can "physically" interact with the representations. This direct manipulation creates the feeling of being in control. The users' competence in performing tasks is consolidated also by visibility of objects and actions of interest and rapid, reversible, incremental actions [51]. Users approach their goals gradually by manipulating the interface objects until their state shows a desired result on the screen.

A common example of direct manipulation is dragging and dropping a file to the trash can. It mimics a real life action and shows visibly all phases and objects involved. Just like in real life where you pick up a paper and transfer by hand, you choose the file to discard and drag it to the trash can. In the end, the effect of the operation is visible immediately. The file is missing from its previous folder and the trash can is not empty. It can be manipulated further to restore the file.

A context sensitive pop-up menu makes it easier for the user to access the functions he needs when he needs them. It is activated in Windows by clicking and object with the secondary mouse button. There is no burden of remembering which menu contains functions related to the object being manipulated. The contents of the pop-up menu change according to the type of the object. Some standard actions like cut, copy, and paste are usually always present. Using context sensitive pop-up menus improves also mouse movement efficiency, because the user doesn't have to hunt around the screen for the proper function.

Object oriented software architecture and the direct manipulation paradigm complement each other well since the development methods call for analyzing the problem domain and the functions that each object can provide. However, choosing the right objects and actions is not necessarily easy. It might be better to start with simple metaphors, analogies, or models from only one source [50, page 205].

4.2.3 Interaction principles for interfaces and supporting user behavior

Designing a user interface successfully means going beyond [50, page 52] solutions that come to mind intuitively when a design problem emerges. Understanding the human behavior of users with interactive systems is needed. There are high-level theories that facilitate application-independent discussion and middle-level principles that help to create and compare design alternatives. Specific practical guidelines like the Nielsen heuristics act as helpful reminders.

People form automatically **mental models** of the world and everything in it [43, page 130]. A mental model is used to make inferences and predictions on how things will happen and machines function. The mental models are incomplete and constantly modified by our perceptions. Designing user interfaces so that they support existing mental models makes the systems easier to learn and more efficient to operate.

Foley and van Dam developed in the late seventies a four level model that was convenient for designers, because it followed their progression of work and software architecture.

1. The **conceptual level** is the user's mental model of the interactive system.
2. The **semantic level** describes the meanings conveyed by the user's command input and by the computer's output display.
3. The **syntactic level** defines how the units (words) that convey semantics are assembled into a complete sentence that instructs the computer to perform a certain task.

4. The **lexical level** deals with device dependencies and with the precise mechanisms by which a user specifies the syntax.

The descriptions of the latter levels show the model's age and close affiliation to textual command-based interfaces. A revised version could be useful for object-oriented, graphical direct manipulation interfaces.

In the early 80s, the GOMS model (goals, operators, methods and selection rules) was proposed. It claims that users formulate **goals** (like edit a document) and **subgoals** (insert a word). The **operators** are elementary perceptual, motor, or cognitive acts (look at cursor position, press a key, recall a file name) that are necessary to affect the task environment, including any aspect of the user's mental state. The users achieve the goals by using **methods** or **procedures** (like move the cursor to a desired location by pushing the mouse, watching the cursor move followed by clicking a mouse button). **Selection rules** are used to choose a method from several alternatives. GOMS can be used to analyze or predict what a user does in an ideal, error free situation and how long it takes.

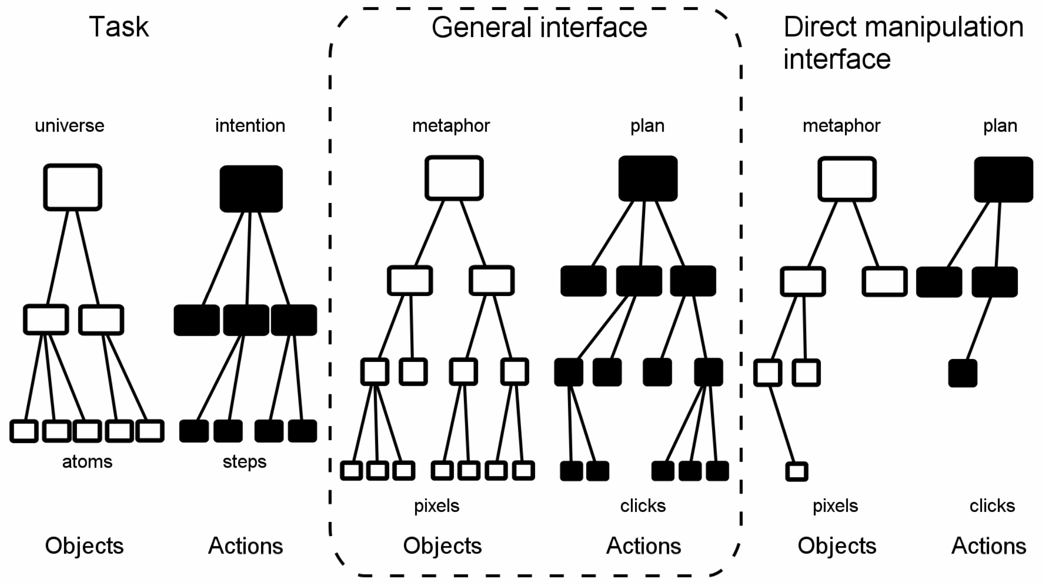
Don Norman offered in 1988 **seven stages of action** as a model of human-computer interaction [40]. In addition to the knowledge in the user's mind, it deals with the dynamic process of cycles of action and evaluation that the user goes through unconsciously. The steps are:

1. Forming the goal.
2. Forming the intention.
3. Specifying the action.
4. Executing the action.
5. Perceiving the system state.
6. Interpreting the system state.
7. Evaluating the outcome.

After the evaluation, new goals are formed and the cycle starts over. Two distinct problems are identified: gulf of execution, which is the mismatch between the user's intentions and the allowable actions, and the gulf of evaluation, which is the mismatch between the system's representation and the user's expectations. Errors can occur at users forming inadequate goals, not finding the correct interface object due to incomprehensible label or icon, not knowing how to specify or execute an action, or receiving inappropriate or misleading feedback. Based on his model, Norman suggests the following four principles for good design:

1. **Visibility**. The user should be able to see the state and action alternatives of the system.
2. **A good conceptual model** the user understands and can relate to. The designer should give the user also a consistent system image.
3. **Good mappings**. The relationships between stages should be revealed, like controls and their effects, or the internal state of the system and its representation.
4. **Continuous feedback**. Results of actions should always be evident.

Schneiderman's **object-action interface model** keeps in with the modern direct manipulation interfaces and their construction. The users' task and interface concepts are separated into hierarchies of real-life objects and actions and their interface representations. The model also describes the design process, which goes top-down and left-to-right in the image below.



The object-action interface model. Direct manipulation systems may require substantial task knowledge, but the complexity of interface elements is reduced.

Decomposition starts from the real world objects of the task universe. For example, you go from stock-market statistics to information on a single stock and into atomic units like share price. Similarly, the actions start from high-level intentions and go to intermediate goals and individual steps. Then the designer can create metaphoric representations of the real world objects to the interface. The user operates the interface with mouse clicks and other actions that stem from a plan and intermediate actions that reflect the task intention and actions.

John Connolly lists [10, page 30] the following design principles that apply especially in the context of computer-mediated communication, but just as well in any applications.

- **Make users feel in command rather than at the mercy of the system.** In particular provide feedback about the actions carried out by the system and about their whereabouts when navigating in complex structures. Make it clear what the users are expected to do next and enable quick and easy escaping from the current process. Offer help facilities for varying levels of knowledge and experience.
- **Spare users as much effort as possible.** Minimize the number of actions required to accomplish a given task. Enable defaults to be set where appropriate. Minimize memory burden. Make error handling as simple and graceful as possible. Enable user actions to be undone.

Connolly also summarizes some screen display guidelines:

- Tread a judicious middle path between a cluttered and an oversparce screen display, and include only what is necessary to the user. Highlight especially important information.
- Organize the display in a helpful manner, by ensuring that the grouping and sequencing of information is well-motivated. Maintain neatness and consistency in the format of displays, and provide clear navigation aids between one display and another.
- Keep abbreviations and codes to a minimum, and keep them consistent.
- Adopt appropriate and consistent formats for dates, and so on.

4.2.4 Intuitive user interfaces

An often used usability requirement for a user interface is that it should be intuitive. Jared Spool explores the concept of intuitiveness in his article [54]. Users find products intuitive when they can intuit, use the interface with intuition. The user interface is intuitable, or familiar, says Jef Raskin [45]. The user has little trouble using a product, if he can use readily transferred, existing skills.

Let’s say we sort all the users by their knowledge of the system and its interface and place them on a line. Those, who know everything about the design, are situated at far right. Those who know barely how to even use a mouse, are on the left. The rest are somewhere in between according to their level of knowledge. Each user’s location gives his Current Knowledge Point. A point on the line, which depicts the knowledge users need to complete their objective with the design, is called a Target Knowledge Point. Distance between these two points is called The Knowledge Gap.

Spool sets two conditions for a design to be intuitive:

1. Both knowledge points are identical. The user knows all necessary aspects of the system to accomplish his tasks.
2. The points are separate, but the design helps the user bridge the gap. The user is unaware that the system is training him, pushing Current Knowledge Point higher, if it feels natural. The designer can reduce complexity to lower the Target Knowledge Point.

One fulfilled condition is enough, but if neither is met, the user might feel that the interface is unintuitive. For sake of scientific accuracy, the first condition should include the case where the Current Knowledge Point can be greater than the Target Knowledge Point.

So, to make a design seem intuitive to users, the designers need to know where these knowledge points are. Current knowledge can be identified by field studies. Observing users working in their own environments gives insight on their knowledge and challenges. Necessary target knowledge emerges from usability testing by analyzing the knowledge gap the user faces when he is working with the design. Just listing all the knowledge the user needed to acquire during the test tells a lot.

If a better interface is wanted, perhaps in terms of learning time, speed of operation, or ease of implementation, it usually has to differ from previous versions. Therefore, it can’t be completely familiar in order to be superior. Other steps have to be taken to bridge The Knowledge Gap.

Sometimes complex tools, such as wizards and data auditors, are needed in the interface. The user has to be asked intelligent questions and be guided through a process with his goals in mind. Clusters of users, who share similar current knowledge, can usually be identified, which helps the designers to get started by focusing on certain personas.

4.2.5 User interface consistency and standards

Making user interfaces consistent and to follow conventions is a prevalent goal for designers. Consistency is a multi-faceted concept, because sometimes it is advantageous to be inconsistent. And you certainly don't want to be consistently wrong.

Platform conventions for interaction and visual appearance should be adhered to. Microsoft provides their official guidelines for Windows user interface developers and designers on their website [32]. Standard functionalities should be kept the same whenever possible, like clicking with the primary mouse button selects an object, or double clicking performs the default action of the clicked object. Common dialogs such as opening and saving files should be kept standard or at most slightly augmented.

The outlook of interface elements that are present in every program, like menu bars and buttons, changes gradually with new operating system versions. Microsoft tends to introduce new looks first in Office products before they are adopted to the next version of Windows. So if the intention is to portray an image of cutting-edge technology, it might be beneficial to be on the forefront also with user interface styles.

Jonathan Grudin argued in 1989 [16] that it would be better to focus on users' work environments than to blindly follow consistency guides as a primary concern. Good design can't be found in the properties of the interface that can be measured and manipulated in isolation.

Grudin defines three kinds of consistency:

- **Internal** consistency of a design with itself, like graphical layout, command naming, objects' colors, shapes, sizes, and other styles. These contribute also to perceived quality of the system.
- **External** with other interface designs, which can compromise internal consistency, or vice versa.
- **External analogical or metaphoric correspondence** to features beyond computer domain. Provides significant aids in recall and initial learning. However, if technology is to provide an advantage, the correspondence to the real world must break down at some point.

A common pitfall for designers is to correspond to the underlying system architecture too rigorously. Interface design is an engineering problem that forces tradeoffs among many factors, often including many possible forms and dimensions of consistency.

User interface design for global applications

Del Galdo [10] contains many examples of design considerations needed for global user interfaces. The globalized industries require software applications that are internationalized so that they can be used by people everywhere. Sometimes specific localization is needed, which means that the interface is tailored to suit a certain culture, country, or language area.

Some cultures write family names first, others last. The number of words and their lengths differ between languages. Different character sets and writing paradigms are used, and sometimes dual keyboards need special support. The direction of text and justification can vary in both horizontal and vertical directions. Segmenting text with delimiters varies as well as the use of vowels, abbreviations, and character linking. Dates and times are marked differently and various calendar types are in use. The graphical layouts of screens have to be able to conform to these differences.

The modern operating systems provide many localized versions and special services for the applications. The applications should be tested on many different platforms for compliance and consistency.

4.3 What is usability

Most people do not use computers and software just because they exist. Users have goals and tasks to do that lead to those goals. They just want to get their work done without too much inconvenience. “User friendly” was a popular term in the past, but it is not sufficient anymore. What is friendly to one can be downright hostile to another due to differences in their context of use.

Official definition of usability by ISO (The International Organization for Standardization) in standard ISO 9241-11 [21] is: “Extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use.”

Further, **effectiveness** is defined as accuracy and completeness with which users achieve specified goals. **Efficiency** means resources expended in relation to the accuracy and completeness with which users achieve goals. **Satisfaction** is freedom from discomfort, and positive attitudes towards the use of the product.

You might say that usability is simply the ability of a person to use something. It can be a product, a toy, a software application, a website, a physical tool or just about anything.

Jakob Nielsen lists five attributes [36, page 26] that affect the usability of a product: learnability, efficiency, memorability, errors, and satisfaction.

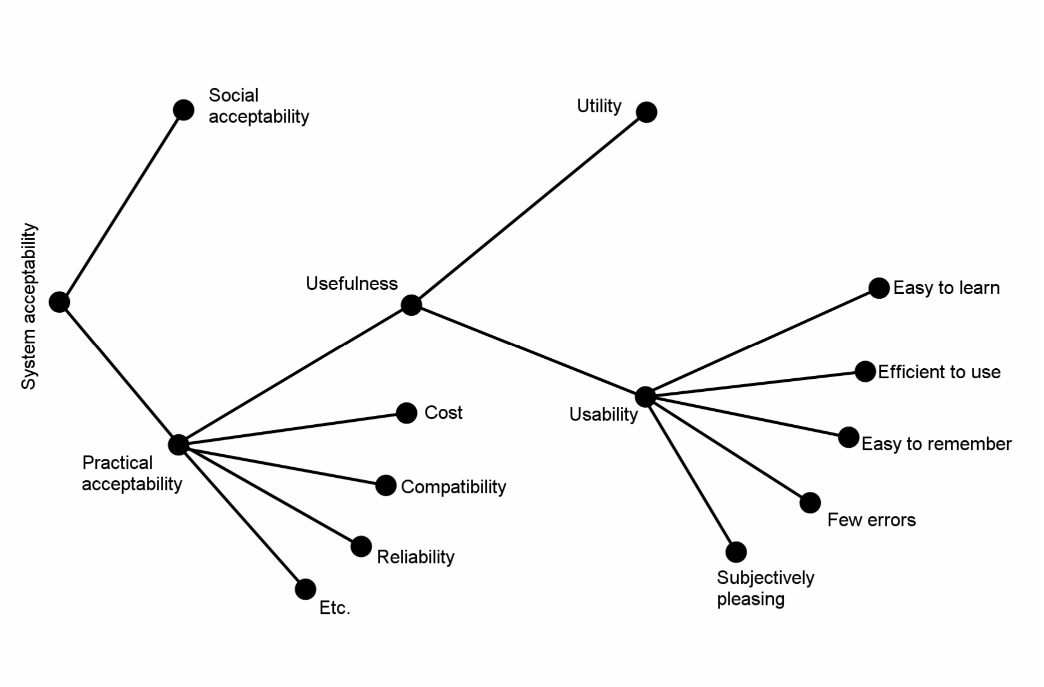
- **Learnability** tells how easy it is to learn to use a new system. For users it is very important that a system is easy to learn so they can start working with it quickly. Usually only a small set of a system’s functionality is used. But when the need for more arises, also the new, perhaps more advanced functions should be easy to learn.

- **Efficiency** of use is desired. After users have learned to use the system, the level of performance can be measured by studying how long experienced users require performing typical tasks. Expertise can be defined by the users them self subjectively, or by following and testing users to find out their learning curve.

- **Memorability** is needed, when the use of a system is occasional. It would be disastrous, if users have to learn everything all over again, when they are in a great hurry to accomplish some task with the system.

- **Error rate** should be low. Beside the system being error-free inside, users shouldn't have many opportunities of making mistakes, especially catastrophic ones that impact on the users themselves or their work. If they do, the system should help then recover easily back to normal operation.
- **Satisfaction** refers to how pleasant it is subjectively to use a system. A system might be discarded, even if other criteria are fulfilled, if the users just don't like using it. This problem is more prominent in a nonwork environment, but it may well have an impact on future business decisions, if the users' opinions get through to decision makers.

Nielsen displays a model of system acceptability as a tree where usability is one node among many other attributes. Under practical acceptability lies usefulness which tells if the system can be used to achieve some desired goal. Under usefulness, utility answers whether the system contains all necessary functionality and usability how well users can use that functionality.



System acceptability model by Jakob Nielsen.

Traditionally, when engineers do designing and testing, they focus mostly on the utility part. Testing functionality is pretty straightforward and easy to define. Usability is an attribute of quality. It can't be rated by a single grade or compared with a totally different kind of product. Also, a product with an explicit level of usability cannot be made by design alone. User centered development methods are needed.

Usability evaluation is employed to find and remove usability defects. Testing the usability of a user interface with real users is similar to inspecting a piece of code for errors and just as important. Another purpose is to test against usability requirements, or summative evaluation.

The professional field working on usability-related issues has many names, like CHI (Computer-Human Interaction). Running the acronym through a blender, you get several combinations and permutations of human/user/man/operator – computer/machine – interface/interaction. Also UCD (User Centered Design), HF (Human Factors), UID (User Interface Design), ergonomics, and the latest buzzword UXP (User Experience) are closely related.

From scientific perspective, usability owes a lot to ethnography, cognitive and experimental psychology, linguistics, human factors and software engineering. Usability started to evolve into a real science in the early 80s. Roots of usability lie in the beginning of the 20th century, when assembly lines emerged in industrial production and methods of efficiency measurements were created [6, page 208]. But more than a hard science, usability is an engineering discipline benefiting entire product lifecycles.

4.3.1 Misconceptions

So, making the user interface as simple as possible yields good usability? Not necessarily [42]. Wrong kind of simplification limits the possibilities of users to operate the system effectively, which violates one of Nielsen's attributes of usability. The user interface should provide the functions of the system when the user needs them. The functions themselves should be simple and feel natural to use.

Usability is **not**:

- a technical problem that can be solved by simple checklists, style guides or exact procedures. You need to employ a user centered design process to really know and fulfill the users' needs and requirements.
- something you can glue on the user interface at the last minute. In fact, the user interface and entire information architecture both need to be designed properly. Just like the rest of the functional system. Usability is about choosing useful functionalities.
- graphic design and decoration. Pretty or artistic user interface may have horrible usability characteristics. However, they can contribute to user experience in other ways. A good screen layout is essential, an esthetic one better, but designing the interaction and intended behavior of users is vital.
- expensive. Usability engineering has a real impact on real projects in the real world [56]. It has a positive return on investment (ROI) and it lowers the total cost of ownership (TCO) of a product. It can boost sales significantly. Read more on discount usability later in this section.
- just research and laboratories. The romantic ideal of perfection cannot be reached even with an infinite budget and massive testing. Even limited, qualitative information from usability testing can make a huge difference in iterative, user centered development.
- an exact science like mathematics, where you can prove theorems indisputably. One might say that usability is an equation with an infinite number of variables and made even more complex by having the human equation included. Select issues relevant to your current project and work on them.

Few slogans from Nielsen [36, page 10] describe some contradictions and misconceptions that are found in the field of usability.

Your best guess is not good enough. An optimal user interface cannot be designed with sheer imagination. It is better to at least validate a design with users if its design can't be based on real understanding about users and their tasks. Admitting mistakes made and correcting them is a measure of maturity in design.

The user is always right. Users are those who work with the design. If they have problems with the interface, it is not their fault. The design has to be modified.

The user is not always right. It is difficult for users to predict their behavior with potential future systems and opinions are diverse. Designs cannot be made solely by asking users what they would like. Other analysis methods must be applied too.

Users are not designers. They have other jobs and they are not user interface professionals. For a customizable user interface to be effective, it has to have coherent design to build on. Novices don't even dare to touch customizations; they have to be taken care of by the initial design.

Designers are not users. Designers have a deep understanding of the conceptual foundation and structure of the system that cannot be undone in order to take the users' perspective. Their experience, knowledge and enthusiasm about computers is worlds apart from regular users. Especially programmers pollute their minds with implementation details, which can significantly curtail their ability to design a user interface that properly supports user activities.

Vice presidents are not users. Their job is to manage assets and resources and make decisions, not to do design. Executives' knowledge on the everyday life of regular users is limited and biased. Same also applies to sales and marketing [17, page 28]. They are in contact with customers, but usually not the actual end-users.

Less is more. Bloating a system with all imaginable functionality is not a good idea, because it places additional burden on the user who has to decide whether to use an element or not. Fewer options are easier to concentrate on and understand.

Details matter. Even the smallest details can make a difference in the usability of a system. Systematic usability engineering work is necessary to ferret out those details.

Help doesn't. Users often don't find the information they need or they misinterpret it. Existence of online help and other documentation is no excuse for designing an overly complex system. In fact, help features add to the complexity.

Usability engineering is process. Detailed advice on the end product cannot be given in a general setting. A guideline may be beneficial to one set of circumstances, but produce a bad user interface in another context. The usability engineering process is well established and contains fairly constant activities that help to arrive at a good result.

The Finnish word "käytettävyys" used to have two meanings. In addition to "usability", it also meant "availability", but the current translation for the latter is "palvelevuus".

4.3.2 Advanced = difficult to use?

So, if a system is designed to be easy for novice users, is it any good for experts? Do they need a system of their own? If a system is designed with the focus on expert users, it might have steep learning curve, but it is not necessary. Certain trade-offs have to be made sometimes with learnability for novices and efficiency of use for experts, but best-of-both-worlds effect is achievable.

The user interface can be constructed with multiple interaction methods to perform the same tasks. A novice can first learn the more general, but slower way, such as selecting an object and activating a suitable function from drop-down menus. Those, who perform the same task more frequently, can take the advantage of accelerators like keyboard shortcuts, automatic fill-in, drag-and-drop, double-clicking, mouse gestures or context-sensitive menus when right-clicking on objects.

All functions should be accessible by the simplest means. Novices shouldn't be forced to use the expert methods when not needed. They can adopt using them in time. Accelerators exist to hasten frequent actions. The key in designing the accelerators is in knowing the users' needs and habits.

Some accelerators serve the novices also, such as default values in dialogs. On the other hand, experts won't mind many concessions to accommodate novices, like labels being more descriptive. If a win-win solution is not possible, the project's usability goals should give the direction and priorities for resolution.

4.4 Evaluating usability

Testing the usability of a system belongs to usability engineering as an integral part of an iterative product development process. There are several methods that are divided into those performed by usability experts alone (expert methods) and those where users are directly involved.

The purpose of testing depends on in which phase of the product development they are done [52, page 297]. Development tests help to find an optimum user interface solution. Acceptance tests make sure that the product fulfills its usability requirements. At the same time the user interface is checked for defects that need fixing or enhanced support and education. Strong sales arguments can also be found.

Usability tests (user tests) are the only objective way of measuring the usability of a product. Feedback from the tests helps to increase the professional user interface designing competence of the developers. It also has an effect on most designers' attitude towards users and the importance of usability. The company learns to make better products and the individuals become better designers.

4.4.1 Heuristic evaluation

One of the most popular and cost-effective usability inspection expert methods is heuristic evaluation [36, page 155]. It is a systematic process which is done by examining a user interface and judging its compliance with a set of recognized usability principles. The rules [37] listed below can also be used when designing the system.

1. **Visibility of system status.** The system should always keep users informed about what is going on, through appropriate feedback within reasonable time.
2. **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 natural and logical order.
3. **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.
4. **Consistency and standards.** Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions.
5. **Error prevention.** Even better than good error messages is a careful design which prevents a problem from occurring in the first place.
6. **Recognition rather than recall.** Make 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.
7. **Flexibility and efficiency of use.** Accelerators – unseen by a 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.
8. **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.
9. **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.
10. **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 tasks, list concrete steps to be carried out, and not be too large.

Nielsen suggests to have at least three usability experts perform the heuristic evaluation to find an adequate proportion of all usability problems. Five evaluators would give a more optimal result. Each evaluator inspects the interface alone so that the aggregated results are independent and unbiased.

An evaluation session for an average system takes one or two hours per evaluator. Larger systems should be split up into smaller parts. The evaluator can get a feel of the system and then go through every screen and dialogue of the system and compare it against the list of rules or check the whole system several times, one rule at a time. Other rules than those listed can be utilized too.

Heuristic evaluation can be used even in an early phase of a development project, where user interfaces exist only on paper, since no real work tasks are performed.

However, if the evaluator is not familiar with the domain of the system, a typical usage scenario from task analysis could be provided.

The output from this method is a list of the found usability problems with references to the principles they violate. The problems can be classified by their severity as shown in chapter 4.4.5. The problems could be communicated in a joint debriefing with the evaluators and developers. Also the positive aspects of the evaluated interface could be addressed in the debriefing.

4.4.2 Cognitive walkthrough

Although cognitive walkthrough of a system is an expert method, also other people than usability engineers, like designers and marketers, can conduct it alone or collaboratively [47]. A cognitive walkthrough focuses on finding properties of the system that hinder learnability which is an important part of a product's usability especially for novice users. Since many people don't care to read instructions, systems should be designed to help users to just start using the interface and figure out how the system works. Reviewing a system with a few cognitive walkthroughs can assist this approach.

In a cognitive walkthrough, the reviewer imagines different people using the system and role-plays their usage scenarios within the limits of their learnability and knowledge. In every phase of the usage situation, the following four questions should be answered:

- *Will the user try to achieve the right effect?* This tells whether the user understands the actions the designer has assumed and that the phase belongs to the task.
- *Will the user notice that the correct action is available?* Is the manipulable part of the object marked clearly and easy to find?
- *Will the user associate the correct action with the desired effect?* Are the labels, terminology, and icons understandable to the user? Is the desired action visible at all?
- *If the correct action is performed, will the user see that progress is made?* Does the user get sufficient and correct feedback to the action?

The main focus of a cognitive walkthrough is not on first-time users, but rather on typical high-frequency tasks and those of critical nature. The method guides the analysts to consider users' mental processes in detail instead of evaluating the characteristics of the interface. The correct sequence of actions is followed; found problems are just recorded, but the walkthrough continues with the correct path.

4.4.3 Usability testing with users

Whether a user interface is OK or not, cannot be solved in any steering group meeting with a demo or a review, it has to be tested. The most fundamental usability method is user testing with real users and real tasks. Many major software companies have their own usability staff with laboratories. Testing services are also readily available from specialized usability firms.

The elementary purpose of user tests [52, page 296] is to improve the quality of use of a product by measuring its usability in a situation that resembles the users' actual working situation with realistic tasks. The results predict how well a product will work in practice when deployed and show potential problems in its use.

Concrete issues are sought, not just abstract principles, or pure user opinions that can diverge in many directions as the number of users grow. Opinions tell the users' level of satisfaction, which is a part of usability, but the problematic areas of using the product are found in user test and they converge as more tests are conducted. Therefore, an extensive number of tests is not needed, nor is statistical significance. The most valuable type of information is usually qualitative.

A typical usability laboratory consists of a room that resembles an office with a desk, chairs and a computer. Video cameras and microphones capture the user's expressions and undertakings with the test equipment. What happens on the computer screen is captured also. Behind a two-way mirror is an observation room with the recording and monitoring equipment.

The test user is accompanied by the experimenter, who directs the user. Often the experimenter has a silent assistant who helps to administer the whole test procedure, move the cameras, and take notes. One more person is needed in the control room to handle the recording equipment and other technical matters. He can also take notes.

Planning and conducting the tests

A usability test process starts with organizing the test activities and laying out a testing plan. After the tests are conducted, their results are analyzed and reported and usually an expert evaluation of the interface is included. The subject of testing can be an entire product, a prototype, some central part of it, or a portion that is believed to be difficult to use. Different usage situations can be chosen to be tested, like commissioning or error recovery.

The objective of the testing should be defined along the relevant aspects of the usability of the product. These should be clear to the testers and those who order the test. Some usability goals and meters for testing are listed in Appendix A. Different user groups use a system for different tasks in different situations. This affects also the importance and ranking of the usability characteristics.

A representative set of test users should be decided next. The size depends on how homogenous the user base is, and the nature of testing, but usually three to six users or pairs is enough. The test users are picked from current or future or potential users of the product, who are not involved in developing it. The users must be assured that they are not judged or doing anything wrong; the system they are testing is incomplete and their input is valuable for its development. Before testing with the actual users, a pilot test should be conducted to make sure that everything works, to measure how long the tasks take, whether their wording is correct, and that nothing is forgotten.

After listing the functions to be tested, they are crafted into test tasks. The test tasks should also be representative of the users' regular operation. It's useful to include both easy central functions as well as hard and complex ones. The tasks are set in a short framing story that is presented to the user as the test progresses. Terms visible in the product should never be used in the test task instructions. Instead of telling the test user to "do a call forwarding", it should be "You want to work in the office across the hall and you want to receive your phone calls there." The experimenter should not help the user explicitly with the tasks unless he is totally stuck.

The user is asked to think aloud while he is carrying out the test tasks. He should tell what he is doing and why. This helps to expose and observe the user's mental models (see chapter 4.2.3). Another way to find out the mental models is to ask the user to draw the way a product works and what components it has. The testing situation and environment can affect the mental models especially, when the product is new to the user. In paired-user testing the users' conversation might contain arguments to convince the other user. If the user is asked to describe the product before the test tasks, his mental models could be affected.

If a user is of the silent type and doesn't make a good progress, the experimenter can take a more active role, and the test situation becomes more like a pluralistic walkthrough. Asking questions actively might interfere with the user's concentration with the test tasks. This requires a lot of skill and sensitivity from the experimenter. True pluralistic walkthrough fits best to an early phase of development, when much of the functionality is not yet implemented.

After the test tasks are completed or the user wants to stop, he is interviewed and/or asked to fill out a questionnaire about the properties of the product. Individual issues and problematic situations can be reviewed and the user's questions answered. System designers could interview the user too, as long as they don't start explaining things and leading on the user. An unbiased person should interview first. The entire test session should not be longer than one hour.

In more quantitative type of testing the user does the test tasks all by himself and afterwards the problem areas can be reviewed with him on tape. The user can't remember everything after the fact or he might censor his thoughts, so his comments don't correspond to his mental models anymore.

A fairly finished product can be subjected to a free exploration. The user finds exactly the functions the system offers him or which he knows to look for. The user has to have knowledge on similar systems beforehand and the test director has to know it thoroughly too.

Afterwards the protocol and usage of the product is analyzed from the tapes and notes and measurements are made. If the user got stuck, the records should be examined to find the place where things started to go wrong and how the mental models of the user and designers differ. A report on the testing should contain also good aspects found in the user interface and not only the problems and their remedies.

Nielsen has done a lot of research on the benefits, reliability, and validity of usability tests. He has found that the pay-off ratio of benefits to costs peaks with three to four test users at 70 [36, page 174]. The ratio drops below 50 when more than nine users are tested. In reliability, five test users when evaluating expert-user performance would give only a 70% probability of getting within +-15% of the true mean width of confidence interval [36, page 168]. 90% confidence interval would be +-24%, which is enough for most projects. Whereas reliability can be measured with statistical tests, a high level of validity requires methodological understanding of the chosen testing method as well as some common sense to assign the right tasks to the right users.

4.4.4 Remote usability testing

Software that has users all over the world should be tested internationally. Conducting usability tests only with users from the country where the software is made is better than not testing at all, but it might create biased results and leave region-specific problems undiscovered. The ways work is done or split between people may have great variations.

The easiest method to improve usability is called **international inspection**, where people from multiple countries are asked to look over the user interface design and analyze whether they think it would cause any problems in their country [10, page 3]. If there are no real tasks to be performed by real users, the results will be partly guesswork, but at least educated guesses – and certainly more educated than the designer's alone. Preferably, the inspectors should be local usability specialists, but local sales office personnel could suffice. Usability specialist can visualize the system from a few screenshot designs and a user interface specification. Others need at least a somewhat functional prototype.

The optimal solution would be to cover all countries in which there are nonnegligible sales or plans of expansion. A normal way is to do evaluation in one country of each main area of the world: Asia, Europe, and North America, even though Asia and Europe are quite heterogeneous.

Doing **international user testing** with real users is the ultimate international usability engineering method. Four main approaches to this are: going abroad to conduct the tests yourself, running the tests remotely, hiring a local usability consultant to run the tests, and having the local branch staff run the tests. In large corporations a fifth choice is to build additional usability groups in major markets, but even then not all markets can be covered.

The preferred choice is to travel and set up tests at the customer's premises since observing the users live gives much better insights than any written report can. An interpreter is usually needed, and he has to be briefed about the specifics of usability testing, like not helping the test user himself and how to translate local idioms and expressions. Recruiting only users that speak the tester's language might result in an unrepresentative profile. The tests could be recorded with a portable usability laboratory that contains all necessary equipment in one travel case. Several products are commercially available. Local laboratory facilities could also be rented.

The tests can be run remotely with minimal facilitation from local people. What happens on the user's screen can be observed over the internet with special software, but it has to be installed and tested successfully first. Communicating with the user can be done with an audio link that runs over the Internet or a phone call. Video conferencing equipment allows to see the user too.

Foreign usability consultants can be trusted to know the basic principles of usability testing, so the goals of the test and scope of the tasks can be concentrated on when discussing with them. They can then run the test in the manner they are most comfortable with and develop the task details to match the environment of their own country.

It might be cheaper to have the local branch office staff run the tests. It also creates them positive feelings about having an influence over product development and learning more themselves. Strict instructions on test user selection have to be given to them to make sure that a representative set is used. Also the whole process of usability testing, including reporting the results, has to be instructed in detail, as it assumably will be a first time for the local staff.

Since not everybody get to travel, the test results have to be communicated to other team members, designers, and management. They can get a good impression of the foreign user situations by showing them highlight reels of recorded video and still pictures. Video and audio recordings may have to be translated and subtitled. There are also transcoding issues, if different video formats were used with e.g. rental equipment.

4.4.5 Error classification

Errors found in testing usability should be ranked by their severity. This gives project management information that helps in deciding priorities for implementing the fixes. Nielsen [36, page 103] suggests a zero-to-four scale:

0 = This is not a usability problem at all.

1 = Cosmetic problem only: fix if extra time is available.

2 = Minor usability problem: fixing this should be given low priority.

3 = Major usability problem: important to fix, should be given high priority.

4 = Usability catastrophe: imperative to fix before product can be released.

Due to the subjective nature, it would be beneficial to have more than one usability specialist do this rating and then calculate the mean values. Solving all problems isn't always feasible depending on the current phase of the project. Another dimension comes from judging weather a problem will be a problem only the first time it is encountered. With a large number of measured test users, the impact of a problem and the proportion of users experiencing it could be factored in.

Sinkkonen [52, page 317] has another rating to use, when product design has progressed to a late stage and there is little time for repairs:

a. A local error, easy to fix.

b. A consistent error, easy to fix with routine operation.

c. Fixing an error requires redesign.

d. Fixing an error requires rediscovering the work processes.

When catastrophic errors are found and they require extensive operations to remedy, it tells that the usability tests have begun far too late. Fixing small and local errors should not be put off, because ignoring them reinforces a negative user experience. They can significantly affect the system as a whole and navigation between and within windows.

4.4.6 Discount usability

Lack of money, time, and other resources is a plight in which one often finds oneself during a development process. Usability testing is still imperative, but the best and scientifically most accurate methods must be then displaced by a more basic approach. Nielsen denotes this as Discount Usability [36, page 17].

The simpler a technique is, the better chance it has of being used in practical situations. More careful methodologies require more usability engineering expertise and may intimidate other system developers. Doing always a perfect job would be nice, but limited resources curtail such ambitions. Taking a discount approach helps to avoid catastrophes, even though some minor problems may go undetected.

User and task observation by simple visits to customer locations is a good way of achieving the basic principle of early focus on users. Without expensive video equipment the “discount task analysis” can be done by keeping quiet, observing the users do their normal work without interference, and taking notes. Only expenses come from traveling to the location and pen and paper.

A cheap kind of prototyping is to use **scenarios**. The level of functionality and the number of features are reduced so much that the simulation works only as long as a test user follows a predetermined path that doesn’t fork much. This is suitable when using paper mock-ups or simple prototyping environments. A small scenario can be changed frequently, which allows testing of many alternatives quickly and frequently. It does take more effort in designing the test tasks so that the users don’t stray off the path, but the tasks must not also be so simple and straightforward that usability problems don’t have a chance to surface.

One of the fastest techniques is **heuristic evaluation**, which uses 10 rules (see 4.4.1) to root out most of the fundamental problems. It does take some experience to apply the principles correctly. Several developers could do the heuristic evaluation concurrently, because different people find different problems. When the lists of perceived problems are combined, the most severe problems are usually found on several lists.

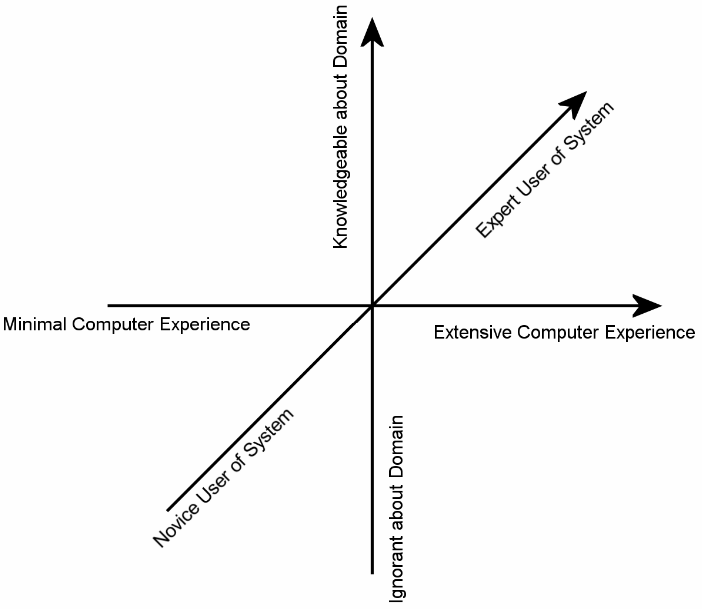
Simplified thinking aloud reveals what a test user is doing and also *why* by verbalizing his thoughts. One user at a time is asked to do a given set of tasks with the system and to “think out loud”. This helps to pinpoint elements of the user interface that cause misunderstandings. Watching the users and taking notes suffices in discount usability when done by ordinary software developers. Videotaping and detailed protocol analysis used by psychologists and user interface experts in a more scientific research might be too intimidating and time-consuming to do.

4.5 Differences and classification of users

People who use software or any product are not a uniform bunch, except in very special occasions. This is clearly evident in the global, mass-produced consumer market. Differentiated groups and individual characteristics exist also in professional business environment. Knowing what these groups are and what they need is valuable capital to developers wishing to sell their products.

Even if the users’ tasks lead to similar ends, their differences affect how the user interface of a system should be designed. In addition to skill sets, people from various cultures have different approaches to work tasks, problems, accumulating new information, and communication, just to mention a few.

Nielsen [36, page 43] models the experience of users (of computer systems) into a three dimensional cube displayed in the image below. The axes are Novice User of System vs. Expert, Minimal Computer Experience vs. Extensive, and Ignorant about Domain vs. Knowledgeable. Naturally, these dimensions are generalizations that could be divided into more specific components, but their relevance depends on the users’ tasks.


Knowledgeable about Domain
Expert User of System
Minimal Computer Experience
Extensive Computer Experience
Novice User of System
Ignorant about Domain

Model for experience of users by Jakob Nielsen.

Hackos [17, page 28] identifies **primary users** as those, who themselves use interfaces to perform tasks. **Secondary users** are affected by the system through their relationships to the primary users. A primary user might, for example, deliver an output document, like a technical report, from the system to a secondary user who needs it in his own tasks. The formulation of the report is significant to the secondary user. The different users usually belong to **communities** of diverse nature that have to be taken into account.

The novice-expert dimension can be branched into four distinct stages of use [17, page 78]: novice, advanced beginner, competent performer and expert. Some users progress through all these stages, but many stay at the lower levels. The same user can be in a different stage with different systems. Facilitating movement between these stages should be a design goal.

Novices have a fear of failure and of the unknown. They do not know the system or what to do to use the interface or information they see. Their focus is on accomplishing real work; therefore they are impatient with learning concepts rather than performing tasks. They might have theoretical understanding only, but no practical experience. The novice experience can last only minutes or hours, but it will take longer if the user has to learn a new subject-matter or technology at the same time with the product. Some users will never progress past the novice stage, especially with infrequent use.

Like novices, also **advanced beginners** focus on accomplishing real work, but without the fear. They are reluctant to spend time learning concepts instead of performing tasks to get their job done as painlessly and quickly as possible. They are not comfortable trying to diagnose and correct problems and are often unsuccessful at it. They can randomly access different tasks and by adding new and progressively more complicated tasks, begin to develop an empirically based mental model. When they have learned to perform the tasks they need, they are content to ignore the rest of the interface.

Those users, who are subject-matter experts, or want to be ones, move on to become **competent performers**, which is measured by the number of tasks and the frequency and breadth of use. Their complex tasks require many coordinated actions. They have formed a sound mental model of the subject-matter and the product. The competence comes through experience which gives the ability to plan how to perform a complex series of tasks to achieve a goal. They can handle unexpected results of their actions and trace back to correct them. Although they are likely to use documentation to learn how things are supposed to work, they prefer straightforward problem-solution tables to more conceptual discussions.

Expert performers use the product frequently as an integral part of their jobs, they have considerable subject matter knowledge, and are able to understand and solve their own and others' problems. They focus on developing a comprehensive and consistent mental model of the functionality and the interface. They are interested in learning about concepts and theories behind a product's design and use, and seek to interact with other expert users. In an organization, they are recognized as more skilled in a particular area than anyone else, and they continually increase their understanding by experimenting and studying to gain new insights.

4.5.1 Characteristic differences

Attributes that are intrinsic to different people affect how they interact with user interface designs [17, page 43]. These individual differences include personal characteristics, physical abilities, cultural differences, expertise, education, and training, and they all have influence on behavior and motivation in the workplace. In product development, it should be taken into consideration, how many users share these differences and that the design is suitable for as many as possible. Still, stereotyping has to be avoided; no pink colors for female users, unless research data shows otherwise.

Learning style belongs to personal characteristics just as reasoning and spatial memory abilities. Some prefer to learn by reading, some ask others how to do the tasks and watch them. Some explore on their own making a great number of mistakes, but learn from them. Formal training classes or self-paced tutorials are also popular. Some need concrete examples while others understand abstract descriptions better [36, page 46]. Learning differences has also a cultural connection [10, page 88]. Motivation and attitudes towards change affect also interface and product design.

Some physical differences like height have little significance when using computer software, but disabilities have to be considered. Movement restrictions and motor coordination difficulties make using a keyboard and pointing accurately with a mouse difficult. Color blindness make certain objects disappear or indistinguishable. Visual disabilities and age inhibit reading texts with small type fonts and detecting small objects.

Cultural differences need to be attended to when selling products globally. Left-to-right reading order is not used everywhere, and icons, metaphors, and other symbols may have even obscene meanings in some cultures. See Appendix C for a table of color associations in different countries. Literacy and language skills set people apart through education and their professional ways of conduct. Even when a foreign user can use an English speaking system, it doesn't mean that he understands what it says; interface usage sequences can be learned by heart. Each industry has its own specialized terms and they vary between experts and non-experts even within the same profession.

There is a lot of literature from ethnographic studies that depict different cultures and highlight the aspects that have to be especially taken into account when conducting business with their representatives. Lewis [28, page 36] classifies the world's cultures in three rough categories. **Linear-actives** plan, schedule, organize, pursue action chains, and do one thing at a time (Germans, Swiss). **Multi-actives** do many things at once and plan their priorities according to the relative thrill or importance that each activity brings with it (Italians, Latin Americans, Arabs). **Reactives** prioritize courtesy, respect, listening quietly and calmly and reacting carefully to the other party (Chinese, Japanese, Finns).

Geert Hofstede has conducted some of the world's most comprehensive studies on values in the workplace in different cultures [20]. The resulting model is known as Hofstede's Dimensions of Culture. The model rates countries on Power Distance Index (PDI, equality in the society), Individualism (IDV, collectivity, achievement, and interpersonal relationships), Masculinity (MAS, gender differentiation, control, and power), Uncertainty Avoidance Index (UAI, tolerance for unstructured situations, rules of conduct), and Long-Term Orientation (LTO, devotion to traditions and long-term commitments).

Recognizing these characteristics can help when (re)designing workflows or analyzing user behavior in usability tests, just to mention a couple.

4.5.2 Ethnography for system design

Ethnography stems from descriptive research by which social anthropologist investigated little-known cultures that differ from their own society [13]. In addition to these differences, they try to understand how the cultures work from the point of view of the people who live in them. Asking direct questions does not reveal the unspoken or tacit knowledge about how to behave. The researcher must observe what people do, how they use tools and objects to accomplish goals and how they communicate with each other, like subculture jargons.

Ethnographers recognize that any or all of their initial assumptions may be incorrect. As many as possible of these assumptions are made explicit and then questioned and systematically examined during the research. Only a minimal structure for the research is planned beforehand because the most relevant issue to study will emerge while conducting the study in the field.

This kind of research usually produces volumes of written notes, audio and videotapes and artifacts collected from the targets. Analysis of this data produces patterns and guides the next iteration with a new set of questions. A thorough ethnographic study takes a long time and it produces unstructured descriptions rather than tightly scientific quantitative data and conclusions are drawn from fewer sources. This might deter those not familiar with this type of research.

In fact, ethnography has a lot in common with proper requirements definition of system designers. They need to understand an unfamiliar culture of tasks and priorities, preserve its essence and translate it into a new system that is understandable and usable. Otherwise, the design won't be accepted by users into their work environment. The designers have to speak the users' own language when gathering information instead of software engineering jargon. Design ideas have to be revised according to the information and the designers need to return to their investigation with new questions.

Traditional system-oriented beta testing tends to be oriented towards debugging. User questionnaires and report forms for debriefing beta users usually produce information of low quality because the users are burdened with twin roles of a software user and an observer of behavior. The information comes in too late in the software development cycle to be able to influence the design in a significant way.

So, it is better to utilize more user centered methods in system design. They are explored in the following chapters.

4.6 ISO 13407: Human-centered design processes for interactive systems

4.6.1 Principles of the standard

The International Organization for Standardization (ISO) has developed many standards that cover also usability issues and user centered product development. ISO 13407 standard [22] describes human-centered design processes for computer-based interactive systems. It is just an overview of human-centered design activities; it doesn't contain detailed coverage of specific methods or techniques. The standard is intended mainly for providing perspective to project managers. It aids in planning and leading the actual product development process in a human-centered way, thus achieving a better level of usability in the developed system throughout its life-cycle.

The following rationale is given for adopting this process. A more usable system is easier to understand and use, thus reducing training and support costs. It improves user satisfaction, productivity of users, the operational efficiency of organizations, product quality, and appeal to users. It reduces discomfort and stress and can provide a competitive advantage. Also the development organization benefits by lower total life-cycle costs. Operational objectives of the system, such as usability requirements, identify the need for a human-centered design approach.

Four key principles of human-centered design apply to any specific design process.

1. The active involvement of users and a clear understanding of user and task requirements.
2. An appropriate allocation of function between users and technology.
3. The iteration of design solutions.
4. Multi-disciplinary design.

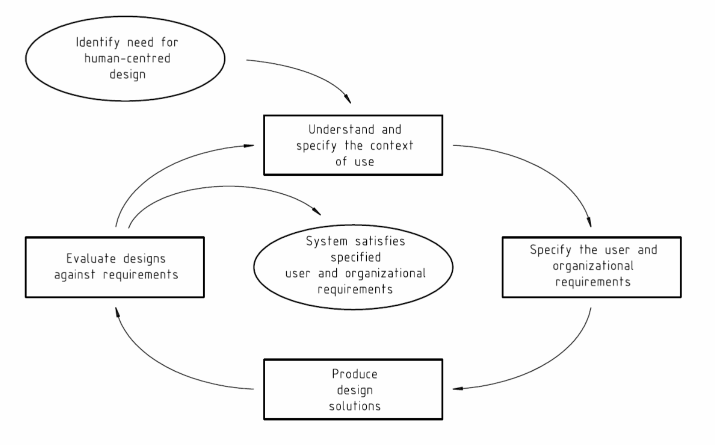
True knowledge about the context of use and tasks is gained only from the actual users. There is a positive correlation between the effectiveness of user involvement and their magnitude of interaction with the developers. Information from and about the users is needed to allocate a meaningful set of tasks for them to perform with the designed system and let the rest be handled by the system with technology.

Iterating design solutions with user feedback helps to refine the solutions from preliminary options to a complete system with minimal risk of the system not meeting user and organizational requirements, including those that are hidden or difficult to specify explicitly. A sufficiently diverse design team has the combined skills necessary for making appropriate design trade-off decisions. Depending on the relationship between the technical development organization and the customer, the team can include a set of the following roles: end-user, purchaser, manager of user, application domain specialist, business analyst, systems analyst, systems engineer, programmer, marketer, salesperson, user interface designer, visual designer, human factors and ergonomics expert, human-computer interaction specialist, technical author, trainer, and support personnel.

The human-centered activities should be integrated into an overall system development project plan. A quality system needs also be augmented to cover the human-centered design process and its quality control measures.

4.6.2 Design activities

ISO 13407 defines four human-centered design activities that are needed in a system development project. They should be started with the project itself, at the earliest initial concept stage. Allocation of focus in the activities depends on the size and type of the future designed product. New products and large market aims benefit from a full multi-disciplinary team whereas in legacy products or niche markets individual team members can occupy multiple roles and use fewer methods.


Identify need for human-centred design
Understand and specify the context of use
Specify the user and organizational requirements
Produce design solutions
Evaluate designs against requirements
System satisfies specified user and organizational requirements

Human-centered design activities of ISO 13407.

Understand and specify the context of use

The context of use means the **characteristics of the users** (e.g. knowledge, skill, experience, education, training, physical attributes, habits, preferences, capabilities), **tasks** (goals, frequency, duration, health and safety implications, not just functions or features) and the **organizational and physical environment** (hardware, software, materials, products, workplace, noise, practices, structure, attitudes) that are associated with the system to be developed. This information is needed to guide design decisions and act as a basis for evaluation. It also contributes to prioritizing user requirements. Some of this information may already exist in the developing organization, but its validity needs to be checked.

This information is usually collected incrementally and the documents are refined as the level of knowledge increases. In the end you should have a description of different user types and their roles and environments that are confirmed by the users and in forms that support design activities

Specify the user and organizational requirements

To identify and state explicitly the relevant requirements in relation to the context of use, the following aspects should be considered:

- Required performance of the new system against operational and financial objectives
- Relevant statutory or legislative requirements, including safety and health
- Cooperation and communication between users and other relevant parties
- The users' jobs (including the allocation of tasks, users' well-being, and motivation)
- Task performance
- Work design and organization
- Management of change, including training and personnel to be involved
- Feasibility of operation and maintenance
- The human-computer interface and workstation design

Important issues are also adequate documentation, trade-offs between conflicting requirements, allocation of function between humans and technology, measurable criteria for testing the design, confirmation by users and updating the requirements during the project, stating human-centered design goals clearly and prioritizing the requirements with the range of relevant users and other personnel in mind.

Produce design solutions

Use existing knowledge to develop and design proposals with a multi-disciplinary input is the first of five components in the activity of producing design solutions. A lot of scientific knowledge and theory from ergonomics, psychology, cognitive science, product design and other disciplines such as standards, internal product knowledge and marketing information can contribute to potential solutions.

Make the design solutions more concrete using simulations, models, mock-ups, etc. to communicate more effectively with users and within the design team. Prototypes make design decisions more explicit and allow exploration of several concepts before selecting one. They allow feedback from users early in the process, and reduce the need and cost for rework and revision, which is expensive later in the life-cycle and especially after initial release to real customers. Different kinds of prototypes can be used at most stages of design.

Present the design solution to users and allow them to perform tasks (or simulated tasks) helps to get feedback that drives the design process. Trying out prototypes in realistic context allows assessing different aspects of the design. If confidentiality reasons don't allow the involvement of users at early stages, expert evaluations can be used instead, but they alone are not sufficient to guarantee a successful interactive system.

Alter the design in response to the user feedback and iterate this process until design objectives are met. Starting with simple paper visualizations and iterating towards interactive software yields maximum benefits. Observed difficulties and user comments guide the functional design changes and help to refine the scope and purpose of the system. More formal evaluation in a realistic context is needed to determine if the overall objectives are met.

Manage the iteration of design solutions by recording the results of these components. The documentary records should include the sources of existing knowledge and standards used, with an indication of how they have been incorporated or why not, the steps taken to ensure that the prototype covered key requirements and followed good practice, and the identified problems with the design changes they caused. The records could include design artifacts (prototypes), too.

Evaluate designs against requirements

Evaluation should be done at all stages in the system life-cycle to provide feedback which can be used to improve design, to assess whether user and organizational objectives have been achieved, and to monitor long-term use of the product or system. Big changes to the system are more expensive to do at the later stages of the development project. Therefore evaluation is an essential part of human-centered design.

An evaluation plan is needed to identify the relevant aspects of:

- The human centered design goals
- Who is responsible for the evaluation
- What parts of the system are to be evaluated and how they are to be evaluated, for example, the use of test scenarios, mock-ups or prototypes
- How evaluation is to be performed and the procedures for carrying out the tests
- Resources required for evaluation and analysis of results and access to users (as necessary)
- Scheduling of evaluation activities and their relation to the project timetable
- Feedback and use of results in other design activities

Specific evaluation goals should reflect on one or more of the following objectives:

- To assess how well the system meets its organizational goals
- To diagnose potential problems and identify needs for improvements in the interface, the supporting material, the workstation environment or the training proposals
- To select the design option that best fits the functional and user requirements
- To elicit feedback and further requirements from the users

By evaluation, it can be demonstrated, that a particular design meets the human-centered requirements, and its conformity to international, national, local, corporate or statutory standards. Minimum acceptance and target levels might be necessary to specify. Getting valid results calls for appropriate methods and realistic tasks for a representative sample of users.

Field validation ensures that the system meets requirements of the users, the tasks and the environment. Help desk data, field reports, real user feedback, performance data, reports on health impacts, design improvements and requests for changes are good sources of field information.

Long-term monitoring is another part of evaluation that has to be planned. User input is gathered over a period of time, because all effects of a new interactive system are not recognizable right away.

ISO 13407 provides skeleton examples on **reporting evaluation results** and proving conformance to this ISO standard. Differences in reporting feedback to design, reporting on tests of the design against specific standards, and reporting on user testing are recognized.

4.7 Participatory Design

Participatory Design is a design ideology of Scandinavian origin sprung from the late 60s [35]. It evolved from trade union demands of giving workers democratic control over changes in their work. The goal is not to replace workers by automating their tasks but to provide tools to make them more efficient in doing their jobs. The users are considered experts of their own tasks and designers are technology consultants who help the users to redesign their work tasks for maximum benefit.

Computers and applications are viewed in the context of a workplace. They are processes, not products. Users' feelings and perceptions of technology are at least as important to success as what they can factually do with it.

Instead of having the first contact with a new software system in the alpha test phase, the users are treated as equal partners on a development team. **Co-locating** the software developers with the user community is recommended at least during the design phase. Communication and the iterative nature of the design-feedback development process are thus facilitated.

Ethnographic field methods are used to observe and interview end-users. This creates an understanding of the natural settings or environment where the new application will be used. People are described nonjudgmentally. Their behavior is understood in a larger social context and seen through their own point of view. Re-creation or play-acting work situations give hands-on experience to designers for the design phase and context for the new tool. This is beneficial because the designers often know little about the work settings for which they design.

Participatory design can be implemented through several methods of **cooperative design**, such as future workshops, organizational games, mock-up designs, and cooperative prototyping. Insights from a future workshop are used in an organizational game, which produces an action plan that tells what can be done immediately at the workplace and what requires external resources, such as new technology.

Another process concept for participatory design is **reciprocal evolution** which consists of three activities: study of work practices with technology, design of technologies, and furthering basic research. General patterns of technology use are sought along with relevant variables that shape interaction with technology. It can help to learn how and why users have modified a product to better suite their work.

Contextual Inquiry is a type of ethnography also used in other techniques. It helps to design systems that support similar work in varying business contexts and cultures. Usability is not seen as an attribute of the system, but rather of the users' interaction with the system. Services provided by the system should match how the users want to do, think and talk about, and structure their work. The three principles in the following paragraph guide contextual inquiry towards this goal.

Descriptions of individual processes do not provide enough information on **context** to support designing for actual workflows. Talking to people while they work and use their tools gives concrete data and opinions on specific points. Forming a **partnership** with the user facilitates the necessary dialogue and control sharing for the gathering of information. Standing outside the work process as a problem solver, watching behavior and taking notes, leaves the users' experience undiscovered. **Focus** needs to be expanded to be dynamic and control shared. The designer must ask about things he doesn't know and double-check what he thinks he understands with the users. He probes behind the solutions offered by the user and discloses his interpretations and design ideas. The focus revolves mainly around issues of what is the users' work, what tools are currently used, what works well and why, and what are the problems that the designer can address with new technology.

Depending on the selected focus, the team conducting Contextual Inquiry can come up with understandings of these:

- A description of users' work
- The flow or structure of the work
- A description of problems in their work
- A description of problems with the computer tools
- Design ideas that emerge from the team's understanding of their work
- Questions for subsequent interviews

A Participatory Design outlook in development promises to lead to quality products, job satisfaction and responsiveness to market, which are necessary to maintain long-term competitiveness.

4.8 The Delta Method

4.8.1 Method history and motivation

The Delta Method is a usability activities framework for software development [11]. It was developed in the early 90s in Sweden by Ericsson Infocom Consultants AB and Linköping University and aimed as a supplement to early phases of traditional system development and integrating usability engineering into the activities. It helps to find and elicit the requirements of the future users by interviews, observations, and studies of users with prototypes. Usability requirements are raised to the same level as technical and functional requirements in order to successfully introduce the new concepts in the development activities. Also a distinction is made between customers and users of the designed system.

Ericsson Infocom relies heavily on documents and specifications for strict and formal controlling of projects. The method was initiated to introduce usability and human factors to Ericsson Infocom system developers and technical communicators and improving communication among them and users. The goal was to improve work situation for the users and to make it possible to develop more usable computer systems. The Delta Method was successfully integrated also into the Quality System in Telia NM [44, page 98].

A member of the original Delta team from Linköping University, Pär Carlshamre, evaluated the method in 2001 [8] as a fit and optimized for business use. It constitutes a good pedagogical model and more than 100 projects used it in 1999. Currently, WM-Data owns the official rights to the method [9].

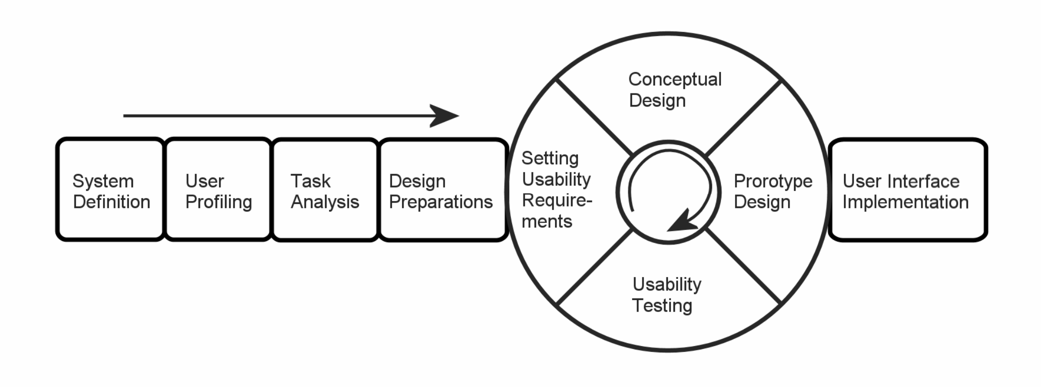
4.8.2 Basics and implementation

A traditional definition of user interface is expanded in the Delta Method from simple graphic layout of a program into three parts:

- The services or functions offered to the user by the system to support his tasks.
- The actual graphic presentation or the services on the screen.
- The enabling information (user documentation) needed by the user to be able to use the system efficiently.

All three aspects are taken into consideration when defining requirements of the system. Significant is the inclusion of documentation from early on. Usually making the documentation tends to be left at the last minute and then rushed together with a minimum effort. Similarly, technical communicators, who make the documentation, should be included in the multi-skilled design group.

In requirements analysis and initial design phase of the development project, the Delta Method involves performing a detailed analysis of the system's planned functions and its users. Also usability requirements have to be defined. This information is the basis of one or several prototypes which are then tested with real users. The complete activities of the Delta Method can be seen in picture below.



The Delta Method development project activities.

In **System Definition**, the design group and customer representatives explore the customer's vision. They analyze the customer requirements, and on a high and abstract level, the preliminary user categories, their rough requirements for the system and their tasks. This phase sets a stage for more user centered work in the future.

An Information Matrix can be used to connect the users and tasks, which converts later into a more textual System Description document that needs to be constantly updated during the entire project. This document serves as guide on what areas the development group should focus on, and as a way to introduce newcomers to the project.

User Profiling and **Task Analysis** describe the users' characteristics, present work situation and tasks. A questionnaire is recommended for initial investigation. Its results are used to distinguish different user groups with different, or similar, requirements for the system. The user profiles are verified and supplemented by interviews of selected representatives from each category.

The purpose of the interviews is to identify the users' present tasks and how they perform them today. Requirements are also formulated with them. The interviews are conducted at the users' workplaces so that the users can be observed performing their tasks. Both the technical communicators and the system designers should take part because their areas of competence and interest and motives for questions are different. Also while one interviews, the other can take notes and construct Action Graphs, which are "played back" to the users for verification and where overlapping, subsequently merged into larger entities.

Design Preparations starts creating the future system. The design group develops fictitious scenarios that describe the future work tasks and user profiles that state the needed skills and characteristics. The scenarios have to be verified with the users and the customer. Needs and requirements not represented in the scenarios are covered by separate design recommendations. The scenarios provide details of the future work that are needed in the user interface design.

Usability Requirements are based on data from the current system, competing systems, or scenarios from Design Preparations. Business and general process goals can also be used as input. The usability goals can be high level ones, such as "The system should be fit for intermittent use", or explicitly measurable, like time to complete a task and number of user errors permitted. A more comprehensive list of different usability requirements from the Delta Method [11, work resource 5] are in Appendix A.

The scenarios and usability goals form test cases for a test specification. The project manager has to negotiate with the customer an agreement on its contents along with the new requirements, just like when deciding the functional requirements.

In **Conceptual Design**, the work on designing the structure of the user interface commences. Initially decisions have to be made on which services the system should offer and how they are presented to the user. Which conceptual model is the interface intended to convey, should a metaphors from the users' background knowledge be used, and which enabling information is needed, are some of the questions that need answers.

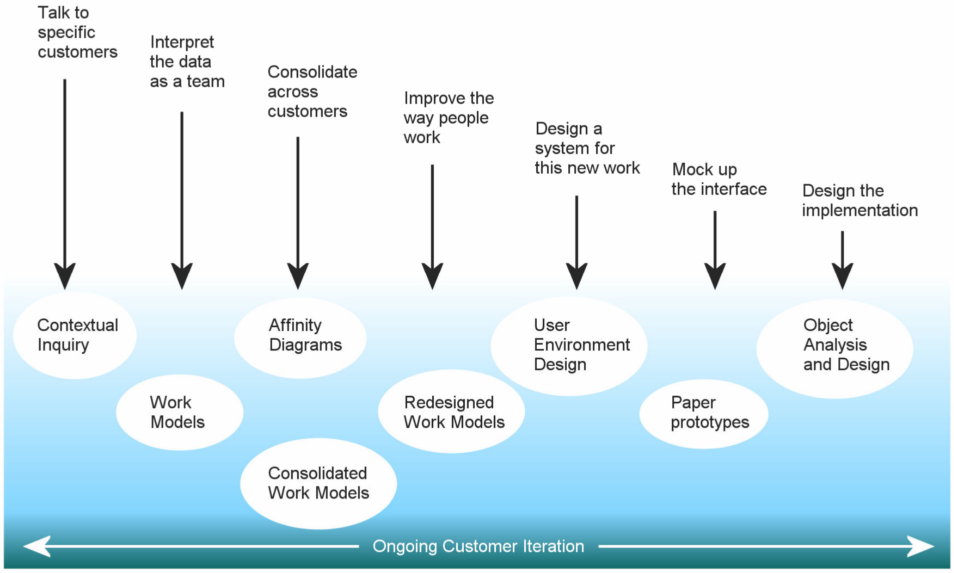
The conceptual model is visualized by rough paper prototypes or neutral computer prototypes and verified with the users and the customer. UED (User Environment Design) notation from Contextual Design [7, page 307] can be used to focus on a general service level rather than having the conversations slip into graphical user interface implementation specifics.

Prototype Design of the user interface stems from the results of the conceptual design. It is performed iteratively with an evaluation against the usability goals at the end of each round. Design decisions are documented so that they can be reevaluated later. Prototypes can evolve from sticky notes on whiteboard to paper sketches to screen window layouts on a computer prototyping tool or interface builder. Deliverable from this phase is a Design Specification.

The structure and contents of enabling information need to be outlined too alongside the system prototypes. Incomplete parts of the user documentation can be subsidized in **Usability Testing** by one of the test managers acting as a “talking book”. The work tasks the users are asked to carry out with a prototype are the same used in formulating the usability requirements. Problems and defects found are addressed in the next iteration along with opinions of the test users. The iterations are continued until their allotted number is full or the usability goals are reached acceptably.

In **User Interface Implementation** the prototype is transformed into the real user interface on the target platform. The Delta team can then verify the success of the transformation by checking if it was complete and correct. They can also review suggested changes to the design. This is done to ensure that design decisions that would jeopardize the usability are not made due to lack of information and understanding. It is always a good idea to test the final product with real users in their own environment.

4.9 Contextual Design


Talk to specific customers
Interpret the data as a team
Consolidate across customers
Improve the way people work
Design a system for this new work
Mock up the interface
Design the implementation
Contextual Inquiry
Work Models
Affinity Diagrams
Consolidated Work Models
Redesigned Work Models
User Environment Design
Paper prototypes
Object Analysis and Design
Ongoing Customer Iteration

Development process according to Contextual Design.

Contextual Design [7] is a popular methodology that had been widely adopted. It’s been called a backbone for organizing the use of a whole range of techniques for customer-centered design [19, page 330]. It defines a framework for a cross-functional design team by providing them a common language for the different stages of the development process. It can be used to support, for example, a Participatory Design project, since they have many activities in common, such as Contextual Inquiry.

Collecting only feedback on existing product or prototype tends to produce variations on the existing theme rather than radical corrections and new approaches to fundamental problems. Users' own summaries are not considered reliable basis for design. Contextual Inquiry was born to tackle the issue of how to better understand how users work. The need to organize, interpret, share and represent the data collected from users brought techniques such as affinity diagrams to Contextual Design.

The different diagram models function as a graphical language by which the development team can communicate. Five work model types make the researched customer's work practice concrete. They uncover and represent the structure of work of individual users, making it possible to consolidate models across all users to form a common structure with internal relations, which forms the basis of design.

- **Context model** represent standards, procedures, policies, directives, expectations, feelings, perceptions, and other emotional and cultural influences (external on internal to the organization) on how people are willing to work.
- **Physical model** represents the physical environment as it impacts the work. It shows the organization of work areas across sites, buildings, and rooms, movement between them, the tools, artifacts, and their relationships to each other. The natural organization of work might be mimicked or supported in the designed system.
- **Flow model** represents people's responsibilities, communication, and coordination, independent of time. It shows how people combine roles in performing their jobs. The roles reveal coherent sets of responsibilities for the system to support.
- **Sequence model** shows the sequence of actions to accomplish a specific task in time including its intent and trigger. It can focus on the coordination of activities across individuals, the thought steps and strategies of a single individual in doing one activity, or the steps taken by a person in using a tool to accomplish an activity. The designed system can support, improve, or replace these steps. They also drive test cases for usability and system tests.
- **Artifact model** reveals the detailed structure, usage, and intent of an artifact created to support the work like a tool, a work order on paper, handwritten additions to manual, a sticky note on the frame of a computer screen, or a notebook entry. The artifacts show how customers break up the work conceptually and the models guide the creation of new system artifacts.

The work models are **consolidated** per type to provide single synthetic representations of work. They reveal structure and strategies that are pervasive and consistent across customers and the market. Without consolidation, there is a risk of creating only isolated fixes for single problems and designing for the specific customers who were interviewed.

Key points, insights, questions, and design ideas are written as separate sticky notes during interpretation sessions of customer interviews and user observations of the contextual inquiry. The notes that have “affinity” with each other are clustered together one at a time and the cluster is named. This bottom-up process allows a structure to bubble up from the details and it created **affinity diagrams**. New design ideas for the work redesign are brainstormed per every part and added to the affinities. The ideas are thus tied to the customer data which originated them.

The size of the diagrams requires a dedicated **design room**, where this shared knowledge of the team can stay undisturbed and referenced whenever needed during the design process and implementation. The advantages of this concept are obvious for collaborative work practices. The room also helps communicating progress and insights to managers and the rest of the organization.

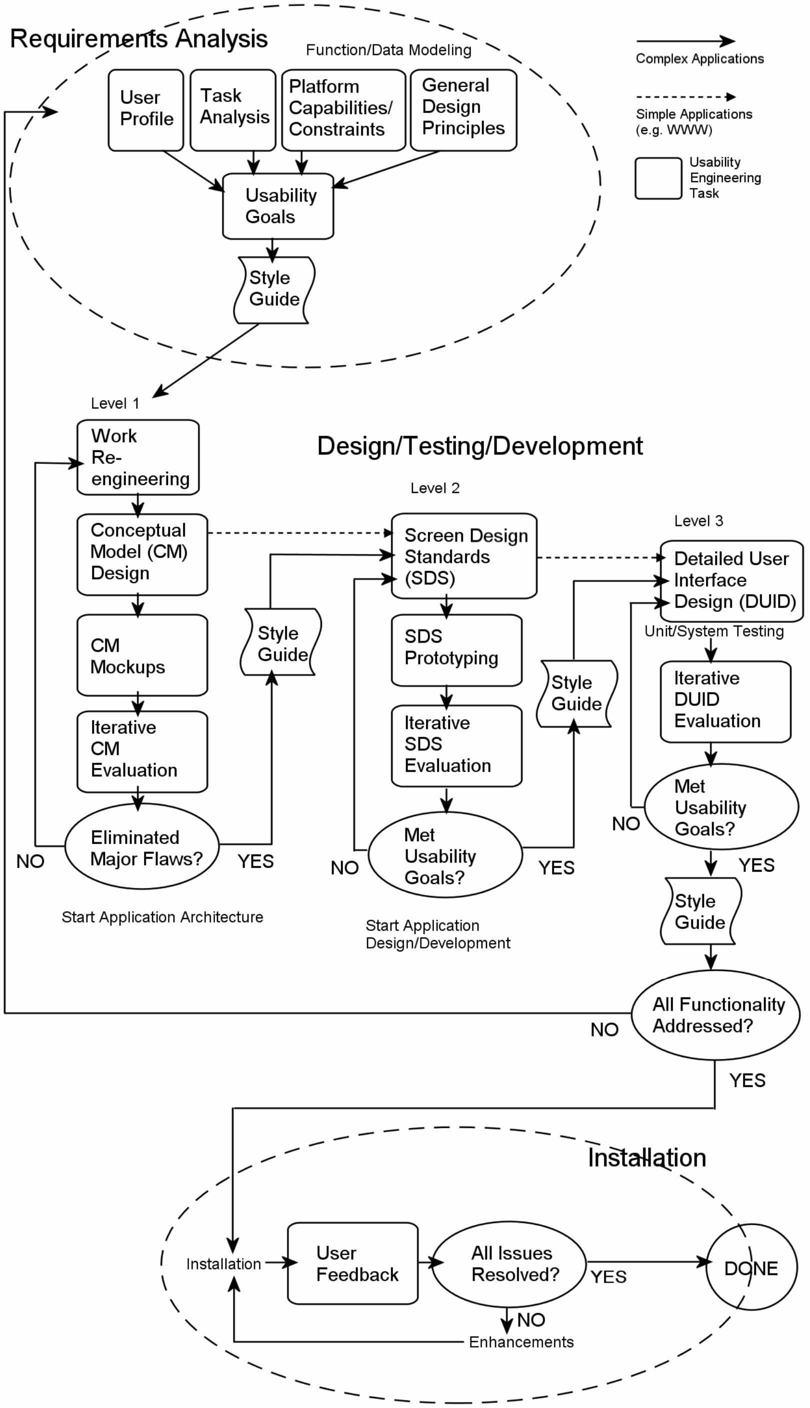
Alternative design ideas from the brainstorming are elaborated for five to fifteen minutes and then compared to identify the good and bad points of each. A Pugh matrix [55] derivative can be used for the evaluation with the current design included as baseline datum. The best parts of the alternatives are merged into a **vision sketch** that focuses on the big picture, still lacking details. The vision is realized through **scripts** which reunite the different perspectives of the work models. A redesigned script shows the role performing the work step, the work step itself, the changes in the physical environment required for this step, and the concepts or artifacts used in this step.

To separate the discussion on system design from the work and implementation issues, **User Environment Design** is used. The UED notation highlights the key concepts for designing a system work model from the redesigned scripts. It shows the parts, or *focus areas*, of the system and their relations from the point of view of the user and his work. Functions or services, links to other focus areas, work objects, constraints, issues, and roles are listed in each as needed. It defines the requirements of the implementation, provides initial objects for the system data model, and defines the structure of the user interface.

The base UED proceeds into rapid iterations of prototyping with users starting with rough paper prototypes. Users simulate doing their tasks and help to develop the incomplete models. Later they are asked to stay within the limitations of the more advanced prototypes. Iterating with the customers keeps the design team creative and not affixed to a specific design; without feedback they might resist discovering that a long-time solution might be wrong [19, page 229].

4.10 Usability Engineering Lifecycle

Deborah Mayhew defines usability engineering as a discipline that provides structured methods for achieving usability in user interface design during product development [30, page 2]. She suggests redesigning the whole development process around Usability Engineering expertise, methods, and techniques. Alternative approaches would be to add cognitive scientists to design teams to provide input on design, or to insert Usability Engineering methods and techniques with specific, written work products into the existing development process. All these approaches aim towards a user centered design lifecycle with varying degrees.



Product development process according to Usability Engineering Lifecycle.

The three-phase process of the Usability Engineering Lifecycle is pictured above. Differences between traditional system analysis and contextual task analysis are described in the following table:

	Traditional System Analysis	Contextual Task Analysis
Goal	Input to the design of: software processes and data structures	Input to the design of: the user interface
Output	Function models and data models	Work Environment Analysis, Task Analysis, Task Scenarios, and Current User Task Organization Model
Impacts	Implementation architecture	Reengineered task organization and task sequence models, Conceptual Model Design, Screen Design Standards, and Detailed User Interface Design
Focus	Technical information processing limitations, data characteristics, and implementation architecture considerations	Human information processing limitations, current work, and current user work models
Objects of Analysis	Data and functions	Users, users' work environment, and users' work goals

Results from the Requirements Analysis phase are documented in the work product called product Style Guide. The product Style Guide is further augmented in the second phase called Design/Testing/Development. The validated Conceptual Model Designs and Screen Designs are captured in the Style Guide at the end of the design/evaluation iterations of design levels 1 and 2. The Style Guide is, in fact, part of the specification according to which the product is then implemented and tested in parallel or overlapping with the Usability Engineering tasks. After installation and being in production for some time, user feedback is gathered for use in enhancement design, design of new releases, and/or design of new but related products.

The Style Guide ensures the documentation of design decisions across time, so they are not forgotten. Information is not lost and tasks don't have to be redone in e.g. staff turnover situations. Is also facilitates communication between dispersed developments teams and it is reusable. Communicating the rationale behind the standards helps to motivate the designers and developers to follow them. The Style Guide potentially contains three kinds of information:

- Principles are general, high-level goals, such as "Support the user's task with visual cues." This helps to organize and focus design efforts, but it is not directly applicable.
- Guidelines are general rules of thumb, like "Use colors to code categories." They are based on research and expertise and focus the design, but they still require interpretation.
- Standards are specific, customized translations, like "Green background for status messages." They can be applied directly and are tailored to users and their tasks. However, they are not optimal to be generalized for all cases.

A skeleton for the contents of the Style Guide can be found in Appendix D.

For successful application of Usability Engineering techniques, Mayhew suggests that the organization assign the roles of a Usability Engineer, a User Interface Designer, and a User Interface Developer. The Engineer is generally skilled in carrying out all the tasks and many of the techniques in the lifecycle. The Designer requires good design skills, but not education and expertise in the other techniques, like contextual task analysis or usability testing. The Developer is responsible for the architecture and actual building of the user interface code with receptivity to the issues and techniques of the Usability Engineering process.

4.11 Miscellaneous techniques

Hackos warns against assuming that the designer's own personal characteristics are common to the users [17, page 50]. The assumptions have to be challenged or verified with real users. When doing usability tests and user and task analysis, it is vital that the visits are not dominated by any one type of user. They need to be profiled in terms of stages of use, and representative sets studied. The more data there is to assist in decision making, the more successful the decisions are likely to be. At least it assists in examining the assumptions and recognizing their influence on the design decisions.

The following subchapters list some additional and noteworthy techniques used in, for example, user and task analysis.

4.11.1 Activity classification

When analyzing data from studying users and their activities, the things they do belong to different contexts of the work process, and should be acknowledged accordingly [17, page 60] The users are goal-oriented, so in the task analysis it's good to start with understanding their general goals. These general goals are more like values; they are always present unconsciously. Companies have similar high-level goals as well. More specific work goals motivate the actual tasks that are performed. Often they can be chosen from several alternatives.

A task has an observable beginning and an end. Its level of granularity can vary between projects, but in one context the level should be agreed upon and held consistent. The order in which users do tasks is called a task sequence or a process. A letter can't be sent before it's written. Task hierarchies are formed, when a large task is made up of subtasks.

A task comprises of actions which means the steps and decisions users take to accomplish the task. Studying the actions is called procedural analysis and they can be modeled with the before mentioned Norman's seven stages cycle. What a single individual does throughout the day or week or month is known as a job. A business process or workflow describes how work gets done when several people and sites are involved, and how information streams between the parties.

4.11.2 Questionnaires and interviews

Questionnaires and interviews are indirect methods for studying subjective issues of usability that cannot be measured objectively [36, page 209]. A questionnaire is useful in determining what features the users particularly like or dislike. In principle, a questionnaire could be sent to the entire user population, but a randomly selected sample is the norm. A set of questions is laid, sent to the population sample (or an address to a web form), and the returned answers are recorded. Conclusions from quantitative analysis can be drawn, if the response rate was sufficiently high and not skewed.

Most of the questions can be alternatively asked during an interview either face to face or via telephone. The problem with an interview is that the respondent has limited time to consider his answers. A lengthy interview takes time from both parties and it is harder than a paper questionnaire to put aside and continue later.

There are some common question types that are used in questionnaires. In closed questions the user has to supply a single fact, go through a checklist or state an opinion on a rating scale. Multiple choice ones are easiest to fill out, but they have to be designed carefully so that the different choices are consistent.

Rating scales use often the Likert scale in which a statement is made and the user selects a value between 1-7 on an agree-disagree axis [36, page 35]. In a semantic differential scale, the user is given a choice of two opposites and a 1-7 scale between them. Also 1-5 or other scales can be used, but one questionnaire should stick to consistent scaling. An even number of choices forces the respondent to have an opinion that is closer to one end of the scale than the other.

In open-ended questions, respondents have to write freeform information in natural language. They tend to take a minimalist approach, writing only what seems needed to answer the most literal interpretation of the question [13, page 276]. Lackluster cryptic statements can be hard to interpret. On the other hand, when using closed-ended questions, you need to know enough about the topic to come up with a sufficiently focused question.

One cannot necessarily trust all the users’ answers. Social acceptance of what kind of replies they ought to give may influence their answering. Goals for the results could include criteria like minimum grade, % of users give at least grade x, no more than % give grade y, etc. A questionnaire should always be subjected to a pilot test before sent out.

Concurrent contextual interviews, where the user is probed for information while he is performing a task, are only one of several ways of talking with users. Depending on how much time there is, the nature of the tasks, and the ability and willingness of the users to talk, other possible interviewing techniques are [17, page 273]:

- Immediate recall interview: record what the user does and talk about it immediately after the task is completed.
- Cued recall interview: record what the user does and talk about it sometime later, perhaps while watching parts of the videotape.
- Process interview: talk with users individually or in groups to understand an entire process or work flow
- Ethnographic interviews: interview one user first as a key informant and then discuss with others later while conducting observations.
- Artifact walkthrough: collect artifacts from the user and construct an interview around them, particularly about differences in the way the user dealt with seemingly similar artifacts.

- Critical incident interview: interview users about specific situations you cannot observe.
- Group interview or focus group: interview users individually or in groups about attitudes, desires, preferences, experience, etc.
- Usability roundtables: interview users away from their work sites with examples of their work as stimuli for discussion.
- Customer partnering: work with group of users over time with interviews as one of the techniques.

Important things to remember when interviewing are to treat the user as a partner, not as a research object, assume he knows a lot about his own work, and listen far more than you talk. Asking “why” is an important question, especially when seeking out requirements and their root causes, but they have to be asked in a neutral manner so that the user doesn’t feel blamed for doing something wrong.

4.11.3 Prototypes

Full-scale implementation efforts should not be started based on early user interface designs. Instead, aspects of the final system can be crafted into prototypes that are developed faster and are cheaper to build. Usability evaluation can start with these concrete but modifiable prototypes and be done iteratively. Alternative solutions can be tested concurrently.

The idea of a prototype is to cut down complexity of implementation by eliminating parts of the full system. Nielsen [36, page 94] calls different kinds of prototypes horizontal, vertical, or scenarios. If a full system is imagined as a 2D box, its width depicts the number of different features and height the complexity of functionality. Vertical prototypes implement only few features, but their functionality in full. Horizontal prototypes contain the user interface surface layer showing all the features of the system, but with shallow functionality. A scenario is a compact intersection of both including their limitations. It has one path of progression that cannot be deviated from.

What kind of prototype to use, depends on what aspects of usability are tested and the users’ tasks. Vertical prototype is suitable for test tasks that need some real data to be used. Paper mock-ups belong to the horizontal class and their functionality comes from the explanations of the human playing the part of the system. Human expert operating behind the scenes is known as the Wizard of Oz technique.

Prototypes can be used as design specifications when communicating with developers. Additional documentation is needed to differentiate, which features of the prototype are intentional and meant to be implemented and which are arbitrary. In a screen design example, some fonts and colors have to be used, but the designer might want them to come from the operating system according to a desktop theme chosen by the user.

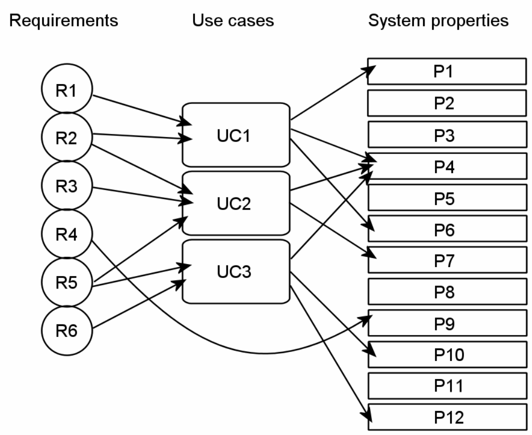
4.11.4 Use cases

Use cases are a widely used method especially in object-oriented software development. They play a key role in, for example, the Rational Unified Process

(RUP). They are defined in the Unified Modeling Language (UML) standard like this [41, page 2-129]:

> Use cases are used to define the behavior of an entity, like a system or a subsystem, without specifying its internal structure. Instances of use cases and instances of actors interact when the services of the entity are used. An actor defines a coherent set of roles that users of an entity can play when interacting with the entity. Each use case specifies a sequence of actions, including variants that can be performed with the entity.

Use cases have a few notations in diagram and textual formats. Their primary purpose is to work as a communication tool for researching customer requirements and mapping them to software requirements [18, page 135]. Use cases cannot function as a basis for system architecture design nor division of coding tasks. They are, however, useful in user interface design as they depict the progression of the user's job or task. It is not necessary for the use cases to cover all system properties, because they don't replace the descriptions of the system in a functional definition document. One use case can include several system functionalities and user requirements, as shown in the following illustration.



Use cases incorporate many requirements and system properties, but not necessarily all of them.

Even though CASE tools (Computer Aided System Engineering), such as Rational Rose, can be used to draw the different use case diagrams, a good textual representation is always needed. The diagrams give a big picture of the modeled behavior and the relations between use cases and roles especially to management, but the details exist only in the text. A good use case should be concrete to be understandable to both the customer and the future users of the system. It should depict the customer requirements and not their specific implementation. Too brief a list of bulleted items of short phrases requires interpretation which can lead to frustration. Full sentences and complete thoughts are preferable.

Use cases should form a basis for testing the system. One use case should not be too large; it should fit on one A4 paper. One template for a textual use case can be found in Appendix E.

4.11.5 User profiles and personas

Communicating the results of user studies might feel like a tall hurdle. Using personas gives a ladder to lower the step and delivers user characteristics and goals to product design and marketing. A user profile can be a simple list of characteristics unique to a group of users [17, page 306]. More narrative forms of description are currently called personas [14]. Personas are user archetypes synthesized from ethnographic studies with real people.

While use cases contain the job descriptions, user profiles or personas include behavior patterns, goals, skills, attitudes, and environment, with a few fictional personal details to bring the persona to life. A good persona description is not a list of tasks or duties, but a narrative that describes the flow of someone’s day. It tells which pieces of information are required at what points in the day, do users focus on one thing at a time, carrying it to completion, or are there a lot of interruptions, and why they are using this product in the first place.

For each product there can be found a set of personas, one of whom is the primary focus for design. For large enterprise applications touching multiple roles, several primary personas can be identified and unique interfaces built to each. The design focuses are not necessarily the same as marketing targets. The number of personas should be kept small; the important distinctions are behavioral, not demographic.

Three or four important goals should be given to the personas. Life goals may not be significant for industrial applications, but experience and end goals help focus the design. Examples of experience goals with using the product are “avoid feeling stupid”, or “confidence of truthful information”. Most important are the end goals that tell what the persona gets out of using a well-designed product, like “solve the customer’s problem”, or an indirect benefit like “be more proactive.”

Personas and user profiles cannot be reused without scrutiny; they are context-specific. Two tools that are very similar or used together on related tasks can share personas.

Hackos notes that one has to proceed slowly when introducing user data to a development team that has no previous experience applying it [17, page 342]. Techniques emphasizing factual information that will be treated as systematic and legitimate are easier to digest than ones depending on narrative accounts. The more the team has had personal contact with the users, the more likely these experiences will have an impact on the design process.

4.11.6 Probing

When a usability engineer cannot go to a user’s site to do contextual environment or task analysis, the user can be equipped to act as a probe. The user could be given a diary to fill out with the daily activities, or a camera to take photos of anything he regards as important [29], like tools and places. He could be asked draw some aspect of his regular working day.

The work products of the probing have to be then subjected to an artifact analysis. Artifact models from Contextual Design could be used to present the ways the users organize and dissect their own work. Questions raised during the study of the work products are addressed in interviews with the probe users. For probing to be successful, the user must not be reluctant to participate.

The resulting information tells more about the personality of the probing user, his environment and the importance of the product to him. Information on a user interface and interaction is gained more by participatory observations. Probing can take a long time with an instrument like a diary.

4.11.7 Card sorting

Sometimes a user interface has to contain large amounts of information. To make it approachable, it has to suit the user's task and the conceptual model the user has formed of it [36, page 127]. The challenge is to organize the information in a way that is useful and meaningful to the users. A good mapping can be achieved with the help of the users.

Each concept is written on a card or a sticky note. The cards are shuffled into one random pile. The user is then asked to sort them into smaller piles of his choosing. Associations in the piles are analyzed, and the user can be asked about his sorting rationale and to name the categories. The user could also add new concepts or rename the existing ones. This method is useful also for designing information architecture, web site structures, or contents of menus and documents. Memorability can be tested too with ready-made categories.

This exercise doesn't produce a finished information design. Identical categorizations are not likely to emerge, and other inputs like business, technical, and usability issues have to be taken into account. Benefits of the method are that it is simple, well understood, quick to apply, and cheap to use.

4.12 Method taxonomy

The following table (based on [36, page 224] with augments) summarizes the methods and techniques presented above and adds some others. Some of these methods and parts of larger frameworks were selected to be tried out in practice (see next section) in order to support current and future development activities.

Method name	**Applicable stages in the development process**	**Users needed**	**Main advantage**	**Main disadvantage**
Heuristic evaluation	Early design, “inner cycle” of iterative design	None, 1-5 usability experts	Finds individual usability problems. Can address expert user issues.	Does not involve real users, so does not find “surprises” relating to their needs.
Performance measures	Competitive analysis, final testing	At least 10	Hard numbers. Results easy to compare.	Does not find individual usability problems.

Thinking aloud (in user tests)	Iterative testing, formative evaluation	3-5	Pinpoints user misconceptions. Cheap test.	Unnatural for users. Hard for expert users to verbalize.
Observation	Task analysis, follow-up studies	3 or more	Ecological validity; reveals users' real tasks. Suggests functions and features.	Appointments hard to set up. No experimenter control.
Questionnaires	Task analysis, follow-up studies	At least 30	Finds subjective user preferences. Easy to repeat.	Pilot work needed (to prevent misunderstandings)
Interviews	Task analysis	5	Flexible, in-depth attitude and experience probing.	Time consuming. Hard to analyze and compare.
Focus groups	Task analysis, user involvement	6-9 per group	Spontaneous reactions and group dynamics.	Hard to analyze. Low validity.
Logging actual use	Final testing, follow-up studies	At least 20	Finds highly used (or unused) features. Can run continuously.	Analysis programs needed for huge mass of data. Violation of users' privacy.
User feedback	Follow-up studies	Up to hundreds	Track changes in user requirements and views.	Special organization needed to handle replies.
Prototypes (in user tests)	Early design, "inner cycle" of iterative design	3-5	Guides design to suit users' needs in their true tasks.	Mustn't be used as only guide for implementation.
Cognitive walkthrough	Early design, "inner cycle" of iterative design	None, 1-3 usability experts	Hones the correct sequence of actions.	Does not involve real users, so does not find "surprises" relating to their needs.
ISO 13407	Entire development lifecycle	Depends on individual methods	Official standard.	General framework only.

Participatory Design	Entire development lifecycle	Depends on individual methods	Users have substantial influence on own work.	Development process renovation. A design philosophy rather than a concrete method.
The Delta Method	Entire development lifecycle	Depends on individual methods	Clear steps and work products.	Development process renovation.
Contextual Design	Entire development lifecycle	Depends on individual methods	Widely used, thorough analysis methods.	Development process renovation, un-engineer-like work products.
Usability Engineering Lifecycle	Entire development lifecycle	Depends on individual methods	Clear steps and work products, customizable per project complexity.	Development process renovation.
Use cases	Task analysis, early design, "inner cycle" of iterative design	2-5 for verification	Communication tool, clarity.	Requires other contextual inquiry methods.
User profiles and personas	Task analysis, early design	2-5 for verification	Communication tool	Narrative skills and other contextual inquiry methods required.
Probing	Task analysis	3 or more	Information from remote environments.	Results depend on users' skills and interests.
Card sorting	Task analysis, early design	3-5	Information architecture coherent with users' mental models.	No conclusive results, expert analysis required.

5 Conducting the research

5.1 Summary

The methods chosen to use in this study are listed in this section. A brief justification is given for each method and then discussed how they were applied in practice.

The work began in October of 2004 by discussions with the tools team on their products and work. The software products and frequency converters were explored. The literature study (see previous section) was also started and it continued in parallel with other activities. Many people of ABB were interviewed to seek out the end-users of the software products, find out who is in contact with them, and what information on them already exists. Other masters' and engineers' thesis from ABB's library and development documents were also examined for this purpose.

Methods of contextual inquiry were selected to get also new user information. By the end of 2004 a questionnaire form was developed and sent to different end-users during the winter. At the same time, observations of 15 end-users were conducted on four different occasions (8 + 3 + 1 + 3) and they took eight days (2 + 1 + 1 + 4) in total. Rest of the spring of 2005 was spent on analyzing the work products of the studies, their results, and developing new design ideas to fulfill new user requirements and to eliminate the found usability problems. The researcher also participated in ongoing development processes and offered user interface insights and user centered techniques to be used where appropriate.

5.2 Interviews

15 people in total from various ABB units were interviewed to find out first what information on users already exists and who is in contact with them. Then, who the users are, what they do with the products, and what problems they have with the products, and how they might be contacted for this research. It was also inquired how requirements gathering and other development activities are carried out currently.

Most of the interviews were done alone. With two interviewees at the same time, it was found that they complemented each other well and could provide more information and details than a single speaker. Four people interviewed at the same time allowed one or two to stay silent for a long time.

A pool of 117 questions Appendix H was formulated of issues to find out. A set of the questions was selected for each interviewee according to their professional area of expertise. Some of the interviews were recorded, especially those that contained specific information on users and their work. Otherwise only written notes were taken. Recording was found beneficial because taking notes and concentrating on the speaker's information content at the same time is hard to do. Because the interviews didn't have a fixed formula other than the questions, the researcher was able to guide the proceeding flexibly according to the content of the answers.

The recordings were done with an iRiver hard disk MP3 player/recorder. Its benefits were time wise a nearly infinite capacity (40 gigabytes), superior sound quality with an external microphone, long-lasting internal battery, and easy transfer to a desktop computer via USB interface. After transferring, the recordings could be accessed at random positions with a media player application. The important points from the interviews were reviewed and written into text files for further processing.

Using a contextual observation/interview data collection template [30, page 106] was considered, but at first attempt it turned out to be too constricting. The interviewees could not describe their work tasks in such a way that this kind of a form could be filled out effectively. They might not be used to speaking about their work in this way or their tasks are very intricate without clear edges and not executed in a straightforward manner.

Support line personnel [5] deal with user problems daily so they were a natural choice as well as the tools team R&D and product managers. Market and product managers from frequency converter development and from domestic and international sales and marketing communication units were also interviewed.

Many of the aforementioned people had field experience of doing commissioning and other work with the frequency converters, which was tapped into. What they could tell about their work in the field was utilized when talking later with current software tools end users from product development and Pulp and Paper units.

5.3 Questionnaires

A questionnaire could be sent to foreign countries and other far away places not able to go visit. It forms a direct feedback channel unfiltered by other ABB units and their agendas. Speculations and guesses about the end-users could be overturned or confirmed. A questionnaire is also rather inexpensive. Issuing questionnaires was chosen as a method for gathering basic information, experiences, and working habits with current software products from tools software end-users. Also their general expectations on frequency converter PC tools software were inquired as well as their interest in participating in further usability or user studies.

The questionnaire form was developed in Finnish with a couple of iterations and translated also into English. The tools team gave input on which issues they would like to find out and the pool of interview questions was also utilized. Work resource 2 of The Delta Method [11] and user profile questionnaire template from the Usability Engineering Lifecycle [30, page 48] had an influence in constructing the questions.

It was desired to get as many answers as possible to be able to form profiles of the end-users and their situations of use, since there are many unknown and unexplored areas for the tools team. However, qualitative information was considered more valuable than quantitative with statistical significance. A representative set of users was desired, but it seemed that the organization had a hard time defining it.

All end-users and their peers encountered during interviews, observations, and customer contacts from sales and product registration were solicited for answers. Also the tools team's personal ABB contacts in Germany, India, the USA, and China were asked to take the questionnaire to about ten of their own customer contacts, and the same request was made to a few product, account, and marketing managers.

Unfortunately, the requests didn't propagate in the organization or at the customers' end. The contact in India filled the questionnaire himself, which was better than no answers at all. From the end-users the researcher contacted, almost all ABB employed people did fill the forms. Only one of the about twenty reached customers returned the questionnaire filled up right away.

The customers were contacted by phone and they all sounded pretty cooperative. They all wanted the questionnaire to be sent to them in electrical format (PDF). If they would have preferred paper forms, it was planned to send along a prepaid return envelope and a small gift as a token of gratitude and an incentive obligating to reply. After a few weeks a free return address was acquired from the post office and a reminder email was sent to the customers with a promise of a small gift. Only two others sent in their replies. The customers were rewarded afterwards.

5.4 User observation

The generalist and sparse nature of the user information gained from interviews prompted the need for and importance of conducting observations of real end-users. Some people were unable to describe their tasks in enough detail or clear chunks for an outsider to get a complete picture. The observations could show how work is actually done with the software tools, and what operating environment related problems and requirements there are that haven't been addressed or even discovered yet.

Both novice and experienced users were desired targets for observation. In the cover letter sent with the questionnaires there was a request for doing a site visit, but alas no external customers agreed. Therefore it was decided to use ABB's internal people and other resources.

5.4.1 Training classes

First contact with eight end-users was made at ACS800 Start-up and Service course at ABB's training center. The course lasted for two days and for many people the exercises are the first time they see and use the software tools, in this case DriveWindow and DriveAP. The exercises turned out to be rather guiding and detached from real-life scenarios. Also the classroom environment with demo cases isn't exactly similar to a factory or even a typical engineers' office. Therefore information only on knowledge and skill levels, general novice problems, initial impressions, and limited learnability could be harvested.

Of the attendants only two came from another company and were novices to ABB software tool products, the rest were from ABB Service unit with substantial experience with at least other products. The students were paired off for doing the exercises and I observed from behind how the pairs conducted the software-related ones. Notes were put on paper with inferences and ideas marked differently from straightforward entries on what the users did or said. The first version of the user questionnaire was also piloted here.

When users faced an error message, they were asked what it means. This and other intermittent questions familiar from usability tests helped to uncover the users' mental models and operating strategies. Asking the users think out loud was not appropriate in the classroom environment as it would have disturbed and influenced other students doing other exercises.

DriveWindow Light usage of three ABB Service people was seen at a one day ACS550 Start-up and Service course. This time the researcher didn't participate in doing the exercises himself as there were not so many new things to learn, which left more time to focus on the other students.

5.4.2 Paper mill fault seeking

An application designer from domestic sales had to do a fault seeking assignment at a paper finishing mill and he was willing to take the researcher along. He had done the drawings for the equipment of the factory, but not visited there before. The concept of explaining the activities as to an apprentice was laid out and accepted. We flew to the site for about an hour in the morning of February 9th and came back the same evening.

As with the training classes, hand written notes was chosen as the best recording medium. A hard cover diary type notebook was easy to write into even when standing up, and to carry to any site where different activities were performed. Activities, environment description, comments, problems, and ideas were written down as they occurred to be analyzed later.

Recording audio was not possible due to the noisy operating environment of the factory. Videography wouldn't have provided enough relevant new information to justify the effort needed for setting it up and post-processing, and for the fact that most work is done on the computer. Flash photography was also out of the question due to detectors of light arcs in the electrics rooms.

5.4.3 Paper mill commissioning

A partial paper machine retrofit was being done at another paper mill and the commissioning of the frequency converters for a new calender machine fell conveniently on late February. It was agreed with ABB Pulp and Paper people and the mill that the researcher could observe the activities there as well using the apprentice concept.

The researcher teamed up with the senior commissioning engineer of this project and, as before, took notes on his and other peoples' actions and other relevant issues. We arrived at the site on Monday, February 14th, and the observations continued for four full working days. The same notebook was used here as during the fault seeking.

A prerequisite for the visit at the paper mill was getting an occupational safety card from a one day safety training class. Also protective shoes, ear plugs, a hard hat, and appropriate clothing were mandatory, and ABB provided all these.

5.5 Applying other methods

The idea of constructing use cases was introduced in a concept design meeting for a new software tool product intended for a new frequency converter. The attendants reacted positively for utilizing them. The benefits of use cases in communicating requirements between a drive unit and tools team and also from tools team to software implementers were recognized.

It was decided that use cases would be tried out since they could also aid in keeping the whole product concept better in the designers' grasps. The contents of the use case template were modified a little in compliance with feedback. Another design meeting was held a few weeks later where three main actors and six use cases were identified along with other requirements and operating environment specific information.

Nielsen's heuristics were constantly displayed by a printout on the researcher's cubicle wall. Often when a problematic situation with any software tool came up, the heuristics were referenced and their violations were marked up in a spreadsheet along with the problem. Screens and dialogs were checked against the heuristics at random times during the research as time permitted. No single session of heuristic review was held where all aspects and screens of a user interface would have been examined.

Cognitional walkthroughs were considered to be done, but the lack of knowledge on the correct paths of execution in the operating environment and information available to end-users prevented them. Instead, the researcher tested intuitive use of the three commissioning software as a novice by attempting to get a demo case motor running without referencing any help material external to the programs. Problems were noted down.

Comparing DriveAP with other programming solutions, such as CoDeSys, was abandoned because no suitable comparison methods were discovered. Also comparing the characteristics of the programming methodologies would have gone outside the scope of this thesis. Comparison with other manufacturer’s commissioning tools turned out to be equally difficult because suitable hardware wasn’t available and one can do very little with just the software.

A formal usability test with test users, thinking aloud, and tasks was also considered, but when there were contacts with suitable test users, no products were in such a phase of development where new functionalities would have benefited from testing, and the user observations supplied enough data on basic usability for this research.

6 Research results

6.1 Summary

This section contains analysis of the findings in this research: what new information was uncovered about the users, found usability problems and user requirements, and a description of some new user interface elements.

6.2 Organizational findings on development and requirements

The interviews and other investigations showed that ABB is a very much technology oriented company. Even though some newsletters from high management emphasize the importance of serving the customers' needs, the effect of these words is not very much visible in the everyday work. Even more so in the tools team.

There is no formal documented information on the users as people. Documents of technical matters are shared in network directories of the projects under which they were conceived. Knowledge on users and their work exists only as tacit information with those employees who have personal experience of said work or active contacts with the users. This information is shared mostly by on-demand chatting at the colleague's desk rather than in a meeting or other established forum of collaborative work. People from various development units expressed displeasure with this practice; they don't have a clear picture of what customers do with the products they make.

Some thesis work on usability-related issues has been done in other ABB units, which contain some utilizable user information. Kirsi Kontio [25] researched the use of Contextual Design with Web Imaging System development in paper and metal industries, Saija Alaharju [4] drives' installation and packaging, Aki Kulmala [26] requirements in food and beverage market, and Elisa Alatalo [5] information flows of product support. Appendix 4 of the Alatalo thesis shows that currently there are no straight information flows between customers and product development. There are deep ravines between different units of the company, which hinders the propagation of information.

When drive development projects gather requirements from customers, they focus on technical matters. Tools team members have not participated in the customer visits nor requested any tools software related issues to be asked. Only if the customers themselves have commented something about the software, it has been written down but not very often delivered to tools team.

Product managers and sales people don't want to inconvenience the customers any more than necessary. Developers are quests of the sales people at customer visits and the visits don't last for more than a couple of hours. They use pre-made question forms that are filled together with the customer representatives.

Generally, customers and sales units view that commissioning software should come free of charge with the frequency converter devices since many competitors utilize this kind of a business model. Also other people voiced similar opinions.

Getting customers involved is a guiding principle that has to be embraced to stay competitive. Customer input and complaints are like gold, to be sought out, and carefully minded to increase customer satisfaction [35, page 297]. ABB has a Customer Complaint Resolution Protocol (CCRP) with online web tools for submitting feedback and viewing reports. So far it is underused and no useful DriveWare®-related submissions were found. The reason might be in the complexity and bad usability of the online tools.

A somewhat parallel system, Statement of Product Need (SOPN), exists too as a mailbox where suggestions can be sent on a pre-formulated Word document. There were only half a dozen several years old DriveWare®-related submissions and most of them were bug reports or about some specific technical matter.

To get people to use systems like CCRP and SOPN to give feedback and suggestions, the tools need to be made very easy to use and supply all necessary documents and information for making choices and decisions.

6.3 Questionnaire results

The questionnaire form is analyzed here. The following subchapters contain results of each question with some commentary from the researcher. The cover page of the questionnaire is in Appendix F. Twelve respondents of the total 28 ticked the box indicating their willingness in answering further questionnaires or participating in usability tests. This is a very encouraging sign for utilizing user centered methods in the future. For numerical results and user commentary answers of some questions see Appendix G.

Ilkka Mellin [31, page 14] directs to use histograms when displaying continuous variables. In this questionnaire the questions where numerical data, like the respondent's age, is collected, the answers are sought in the form of pre-selected groups, which makes the variables discrete [34, page 45]. Therefore bar diagrams are used in visualization.

The N after each question tells how many respondents answered it. Since the sample of users was not representative or large enough to be statistically significant, the reader must draw his own conclusions from the results, their validity, and applicability to his needs with prejudice.

To find out how similar respondents' answers were compared to each other, standard deviation and average deviation, also known as mean absolute deviation, were calculated when possible. Standard deviation is a basis for many other statistics calculations, which has made it more popular, but it is also affected more by aberrant samples in the data than average deviation [15]. Generally, the users were most unanimous about the issues they also felt were the most important ones.

Answer distributions in questions with a single Likert or semantic differential scale are presented with vertical bars. In questions with several scale sub points, horizontal bars depict the average value of answers, and the thin black bars show the average deviation in both directions.

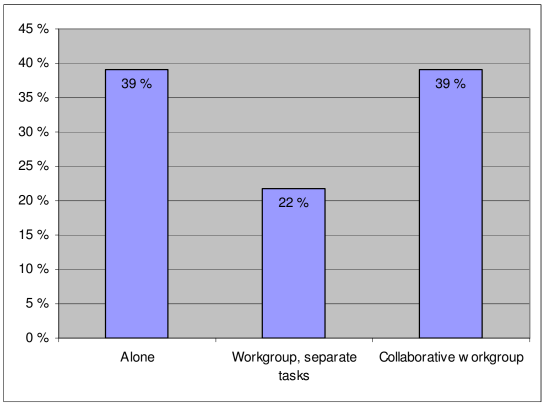
6.3.1 Section A: general information on users and their environment

A1. Profession or title, A2. Main tasks and responsibilities
25 of the 28 respondents worked for ABB. Two of them were engineer students working on the side but the rest were employed full time. Among the respondents were an entrepreneur, engineers, technicians, developers, sales people, and trainers. Three of the ABB engineers worked in type testing, eight in service and support, and 11 in Pulp & Paper division, mostly in commissioning duties.

A3. Role at work
Three respondents categorized themselves as (team) leaders, 19 didn't answer anything. The remaining six were members of a team or just workers.

A4. I do my work tasks mainly

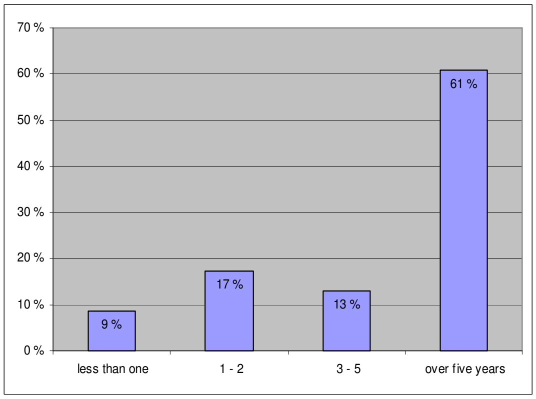
N = 23



A5. I've worked with my present tasks for

N = 23

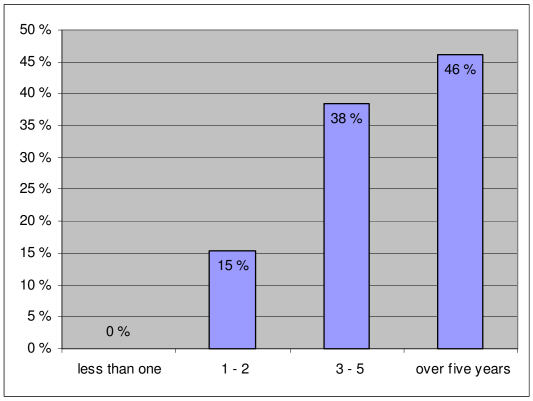
The respondents have considerable experience in their work.



A6. I will continue with these tasks for

N = 13

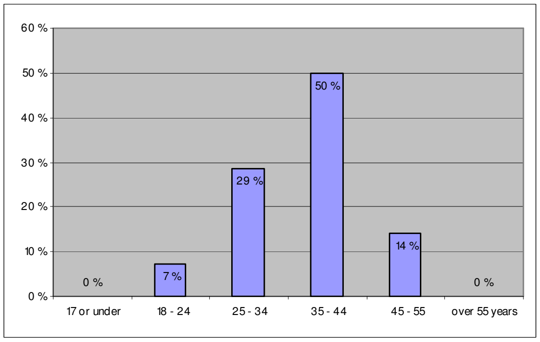
There is not a lot of fluctuation in the employment status.



A7. Age

N = 28

The respondents are in the prime of their working years.



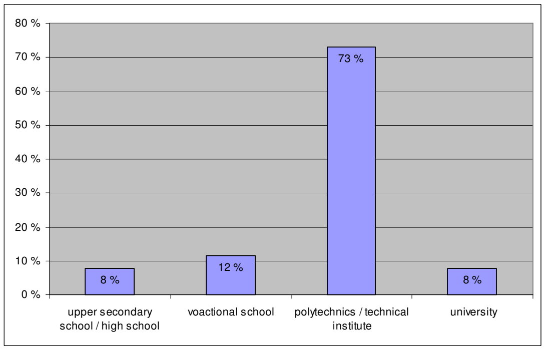
A8. Primary language, N = 27

Finnish: 81 %, Swedish 15%, other 4%

A9. Education

N = 26

The highest level marked was counted. 12 had studied at the older "ammattiopisto", 7 at the current "ammattikorkeakoulu", but these were combined in the diagram.



A10. Highest degree, N = 24

Answers followed above places of education.

A11. Major / field of study

N = 24

Five automation related, one communication technology, one physics, and the rest power electronics, or combinations of these.

A12. English language

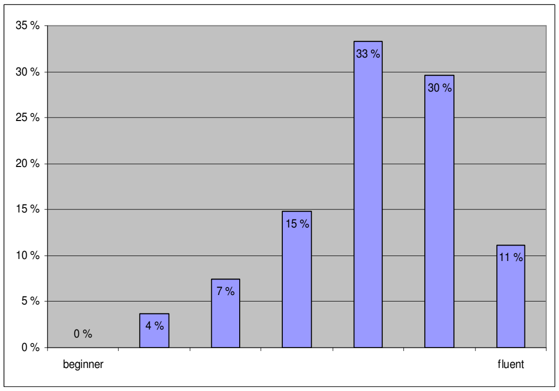
(beginner = 1, fluent = 7)

N = 27

Average = 5,11

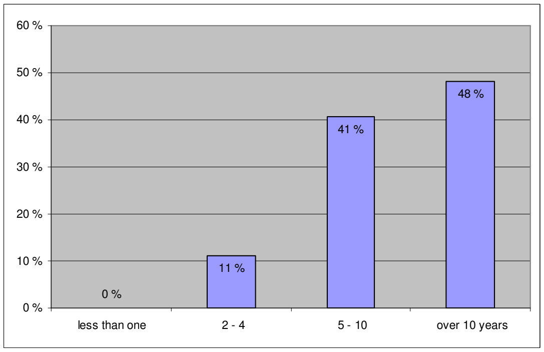
Standard deviation = 1,25

Average deviation = 0,95



A13. Experience on electric drives

N = 27



A14. What drive models (from ABB or others) do you have at your work

N = 23

Service personnel deal with almost every drive model. 600, 700, and 800 series got the most references as well as some older models. Siemens and Vacon were mentioned once.

A15. Drive usage at your work (estimate number of drives, if possible) (pumping liquids, gas transfer, mixing of pulp/bulk, conveyor, rollers, in a paper machine)

N = 15

Inconclusive, but those who answered this question deal with several, even dozens of drives. Pumping, conveyors and paper machines got most references.

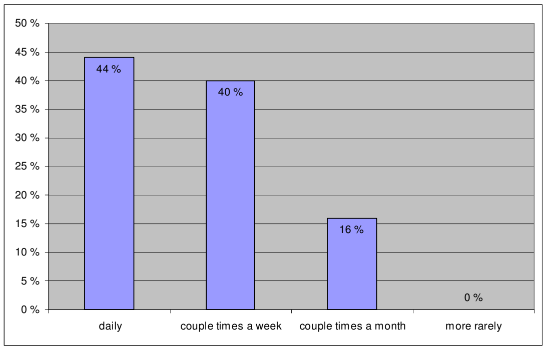
A16. I use a computer at work

N = 26

Same alternatives as below. Everyone answered “daily”.

A17. I use computer at home

N = 25



A18. Computer skills

(beginner = 1, expert = 7)

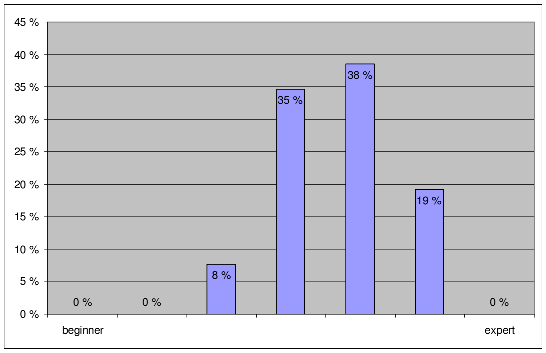
N = 26

Average = 4,69

Standard deviation = 0,88

Average deviation = 0,74

No self-proclaimed experts.



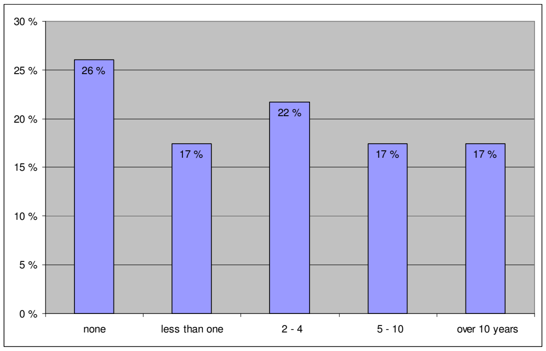
A19. Disabilities that obstruct using computers (such as color blindness, prosthetics)

None reported.

A20. Programming experience

N = 23

The results are slightly balanced towards a short experience (four years or less)



A21. Programming skills

(beginner = 1, expert = 7)

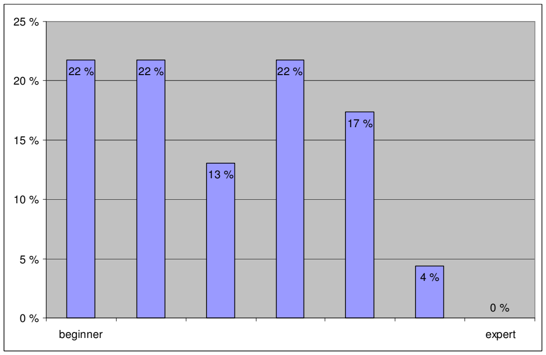
N = 23

Average = 3,04

Standard deviation = 1,58

Average deviation = 1,35

No self-proclaimed experts.



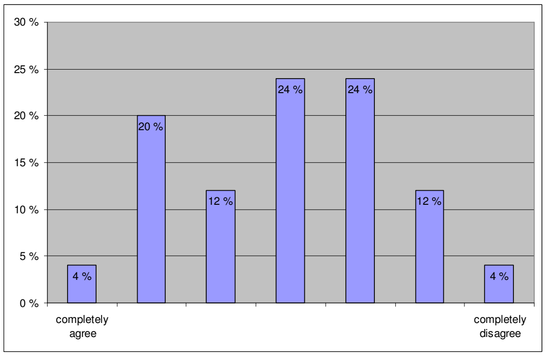
A22. My work is routine (completely agree = 1, completely disagree = 7)

N = 25

Average = 3,96

Standard deviation = 1,57

Average deviation = 1,25



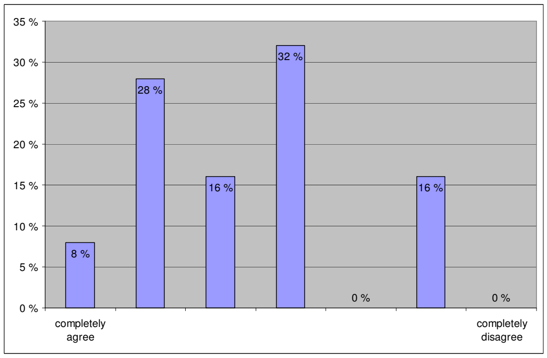
A23. I am proud of my work (completely agree = 1, completely disagree = 7)

N = 25

Average = 3,36

Standard deviation = 1,52

Average deviation = 1,25



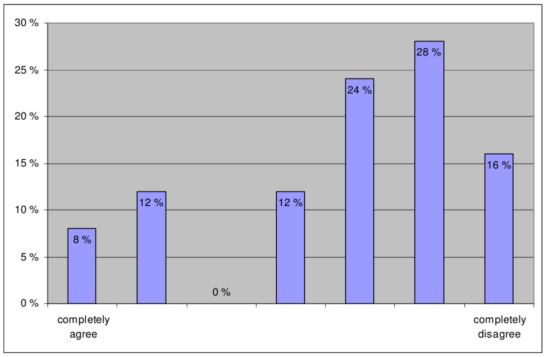
A24. Computers cause extra problems in my work (completely agree = 1, completely disagree = 7)

N = 25

Average = 4,80

Standard deviation = 1,87

Average deviation = 1,47



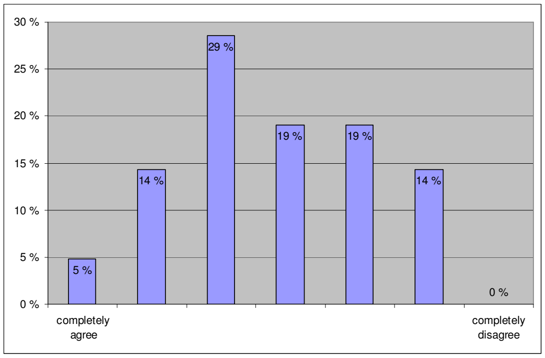
A25. I need information from my colleagues to accomplish my tasks (completely agree = 1, completely disagree = 7)

N = 21

Average = 3,76

Standard deviation = 1,45

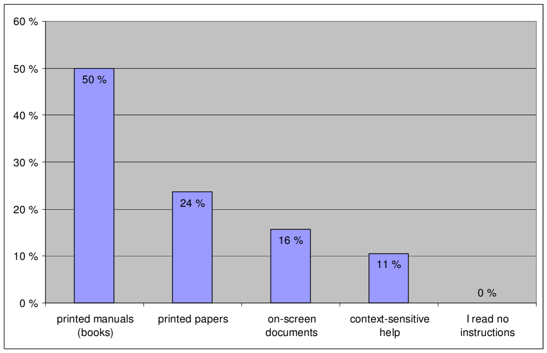
Average deviation = 1,20



A26. I prefer to read instructions from

N = 38

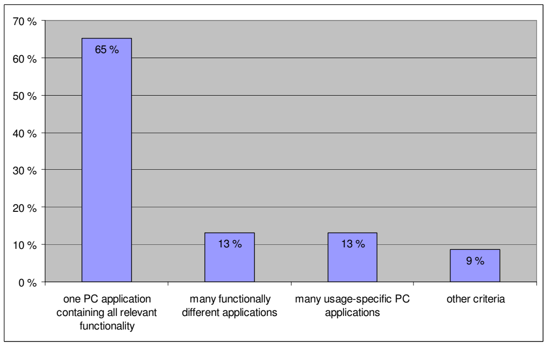
Several alternatives from 26 respondents. Indicates different reading strategies depending on situation of use (in office, on an assignment).



A27. My work with drives would be supported best by

N = 23

20 respondents, some of which suggested different tools for regular customers and professionals.



Question A21 on programming skills can be interpreted to mean different things depending on the computer skills of the answerer. An expert computer user is likely to be aware of advanced computer programming languages like C++ and Java but have little or no experience using them. On the other hand, a drive expert may be competent in block programming with DriveAP but a beginner with computers in general.

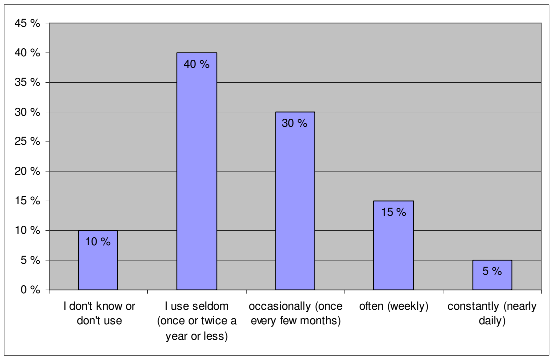
6.3.2 Section B: DriveWindow

DriveWindow 2.x is used a lot, but clearly it takes some getting used to. Not one respondent chose the first box in B2. It is also thought of having a somewhat confusing look.

B1. Usage

DriveWindow 1.x

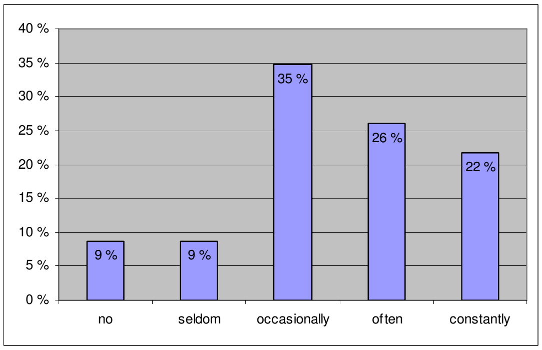
N = 20



DriveWindow 2.x

(same options as above)

N = 23



B2. Using DriveWindow 2.x is

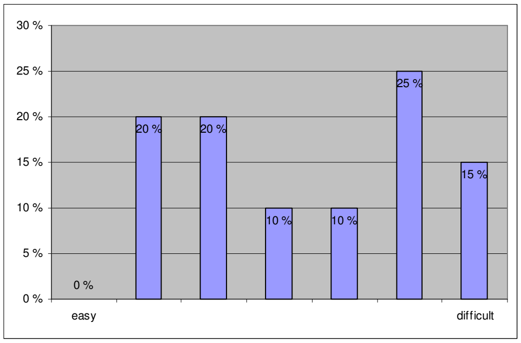
(easy = 1, difficult = 7)

N = 20

Average = 4,45

Standard deviation = 1,85

Average deviation = 1,65



B3. DriveWindow 2.x looks

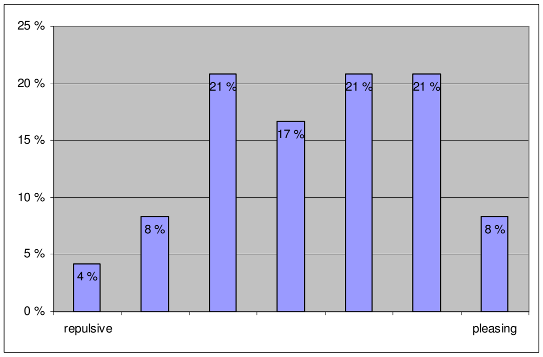
(repulsive = 1, pleasing = 7)

N = 24

Average = 4,38

Standard deviation = 1,64

Average deviation = 1,38



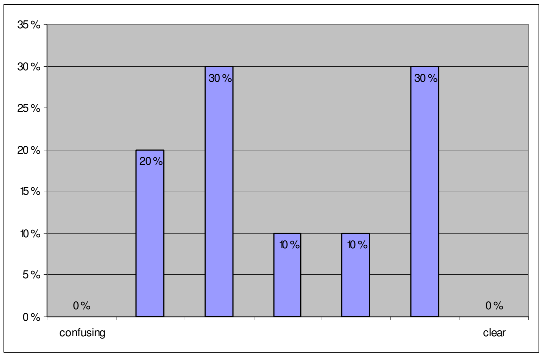
(confusing = 1, clear = 7)

N = 20

Average = 4,00

Standard deviation = 1,59

Average deviation = 1,40



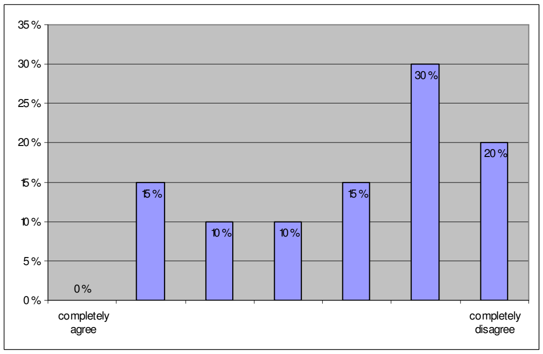
B4. Working with DriveWindow 2.x is efficient (completely agree = 1, completely disagree = 7)

N = 20

Average = 4,95

Standard deviation = 1,76

Average deviation = 1,47



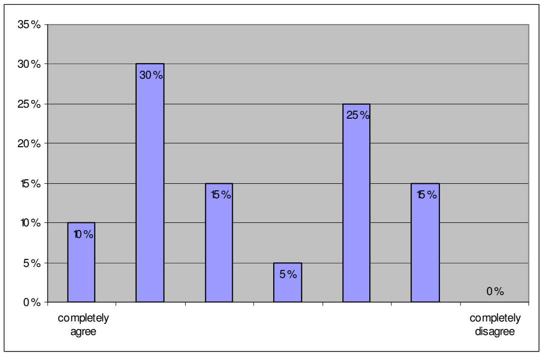
B5. I learned DriveWindow 2.x quickly (completely agree = 1, completely disagree = 7)

N = 20

Average = 3,50

Standard deviation = 1,73

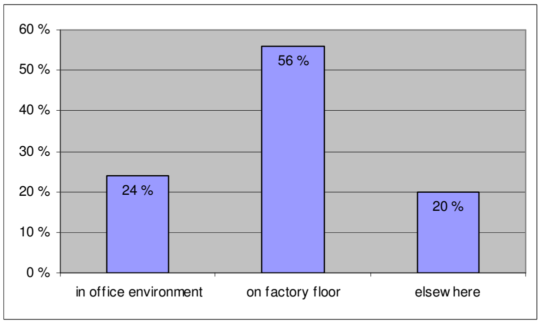
Average deviation = 1,55



B6. I use DriveWindow 2.x

N = 25

Several alternatives from 20 respondents. Electrics rooms or other customer facilities were mentioned five times.

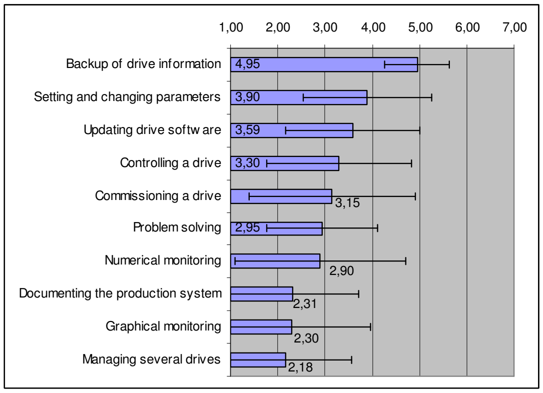


B7. DriveWindow 2.x supports the following work tasks

(very badly = 1, very well = 7)

N = 13 – 20

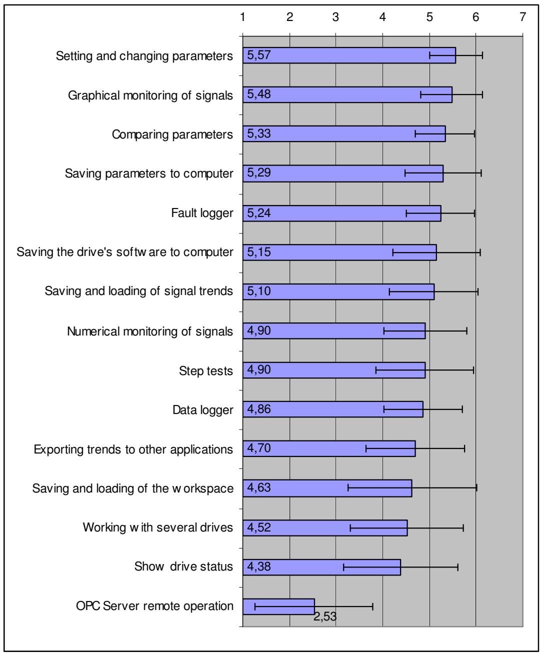
All averages except for the first one are on the negative side of the scale (less than 4). The users are clearly not happy with this software.



B8. Importance of DriveWindow 2.x functionality in my work tasks

(unimportant = 1, vital = 7)

N = 17 – 21



B9. Comments on working with DriveWindow 2.x and/or comparison with other similar software (like DriveWindow 1.x)

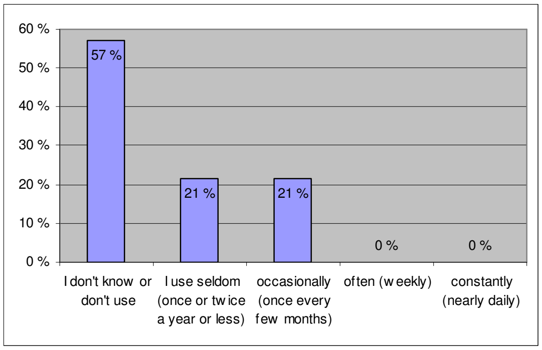
See Appendix G

6.3.3 Section C: DriveWindow Light

C1. DriveWindow Light usage

N = 14

Not a lot of active users within the respondents.



C2. Using DriveWindow Light is (easy = 1, difficult = 7), N = 6

All respondents answered 2.

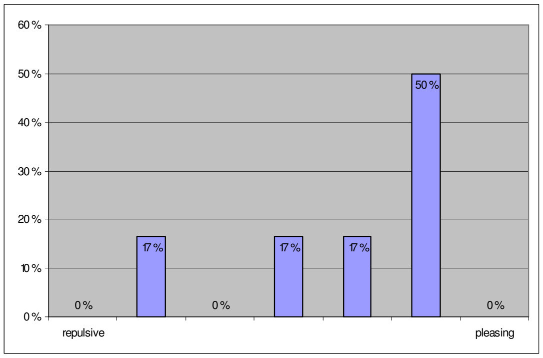
C3. DriveWindow Light looks (repulsive = 1, pleasing = 7)

N = 6

Average = 4,83

Standard deviation = 1,60

Average deviation = 1,22



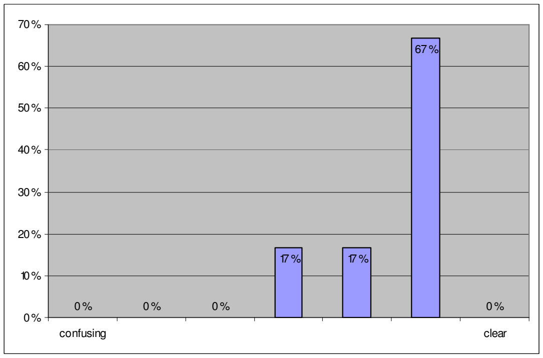
(confusing = 1, clear = 7)

N = 6

Average = 5,50

Standard deviation = 0,84

Average deviation = 0,67



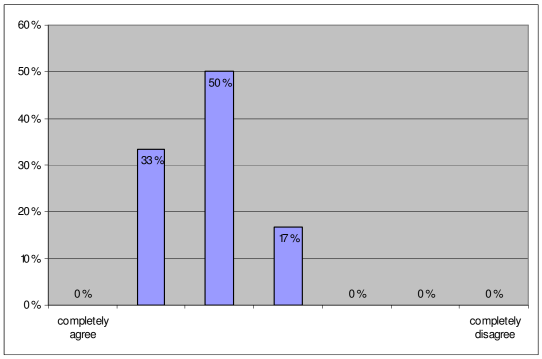
C4. Working with DriveWindow Light is efficient (completely agree = 1, completely disagree = 7)

N = 6

Average = 2,83

Standard deviation = 0,75

Average deviation = 0,56



C5. I learned DriveWindow Light quickly (completely agree = 1, completely disagree = 7), N = 6

Average = 1,83

Standard deviation = 0,41

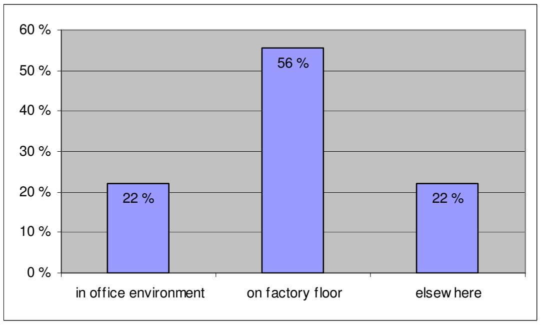
Average deviation = 0,28

One respondent selected box 1, other five selected box 2.

C6. I use DriveWindow Light

N = 9

Several alternatives from six respondents.

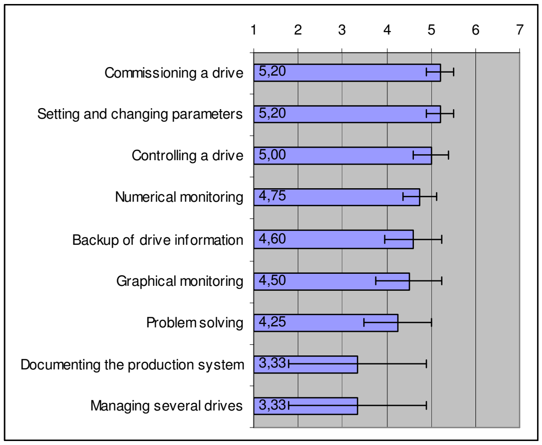


C7. DriveWindow Light supports the following work tasks

(very badly = 1, very well = 7)

N = 3 – 5

Most answers were positive and pretty unanimous.

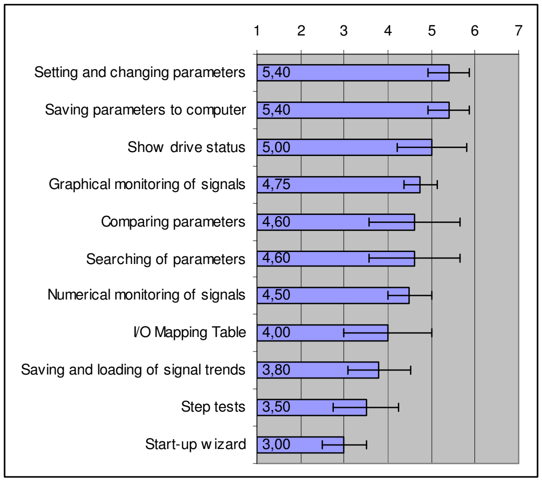


C8. Importance of DriveWindow Light functionality in my work tasks

(unimportant = 1, vital = 7)

N = 4 – 5

These users don't seem to have much of a use for the start-up wizards.



C9. Comments on working with DriveWindow Light and/or comparison with other similar software or using just the panel of the drive

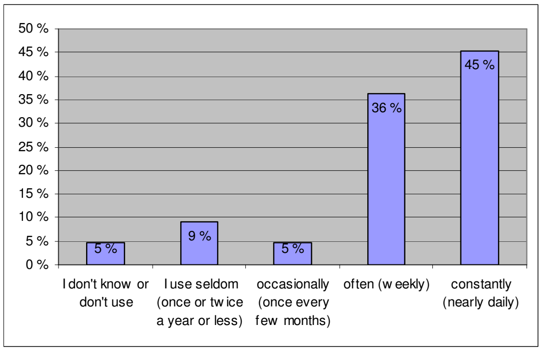
See Appendix G

6.3.4 Section D: DriveDebug

D1. DriveDebug usage

N = 22

This software is in active use, especially among ABB employees.



D2. Using DriveDebug is

(easy = 1, difficult = 7)

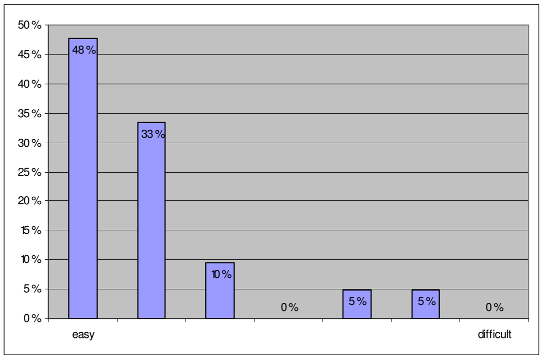
N = 21

Average = 1,95

Standard deviation = 1,36

Average deviation = 0,91

A familiar tool.



D3. DriveDebug looks

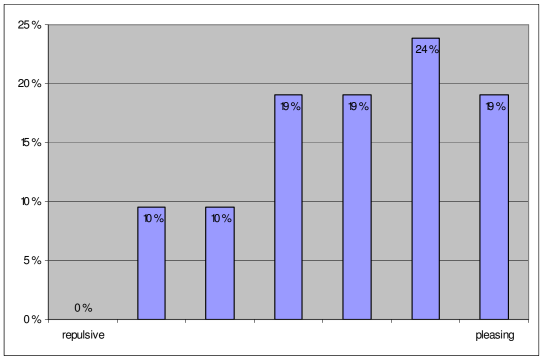
(repulsive = 1, pleasing = 7)

N = 21

Average = 4,95

Standard deviation = 1,60

Average deviation = 1,30



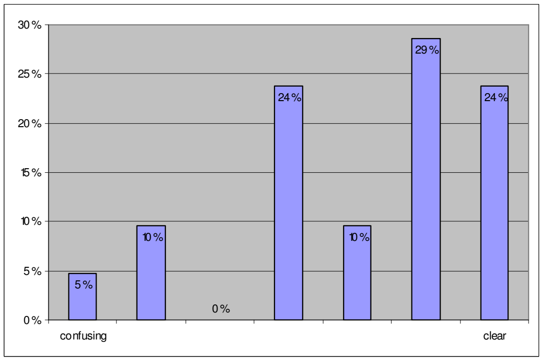
(confusing = 1, clear = 7)

N = 21

Average = 5,05

Standard deviation = 1,80

Average deviation = 1,47



D4. Working with DriveDebug is efficient (completely agree = 1, completely disagree = 7)

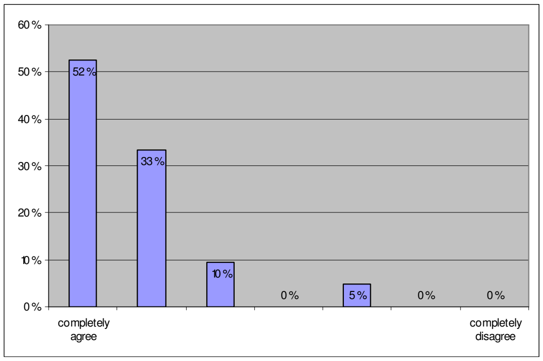
N = 21

Average = 1,71

Standard deviation = 1,01

Average deviation = 0,75

Work gets done well with this tool.



D5. I learned DriveDebug quickly (completely agree = 1, completely disagree = 7)

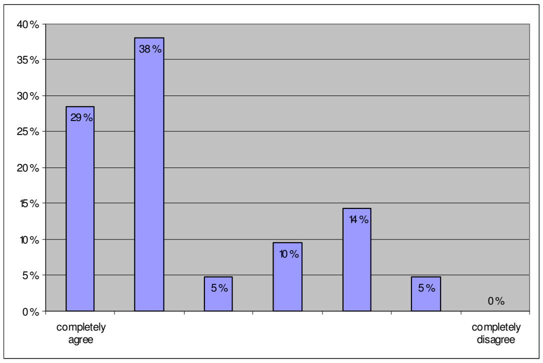
N = 21

Average = 2,75

Standard deviation = 1,60

Average deviation = 1,33

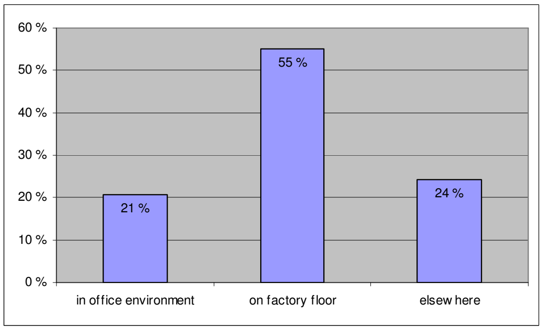
There is a slight learning curve.



D6. I use DriveDebug

N = 29

Several alternatives from 21 respondents. Laboratories and electrics rooms of customers got some mentions.

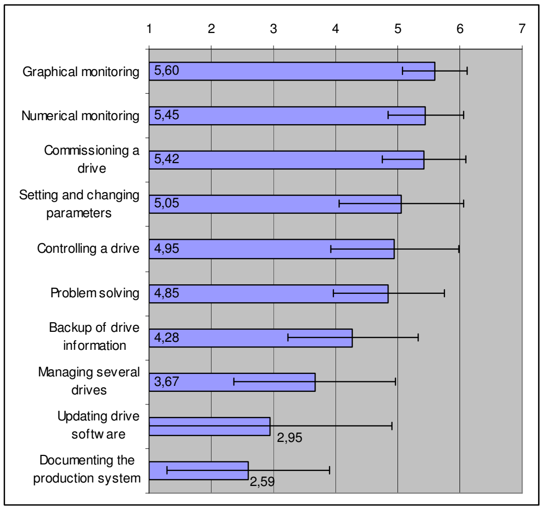


D7. DriveDebug supports the following work tasks

(very badly = 1, very well = 7)

N = 17 – 20

Majority of answers signal positive attitudes. There is substantial disagreement in the negative answers.

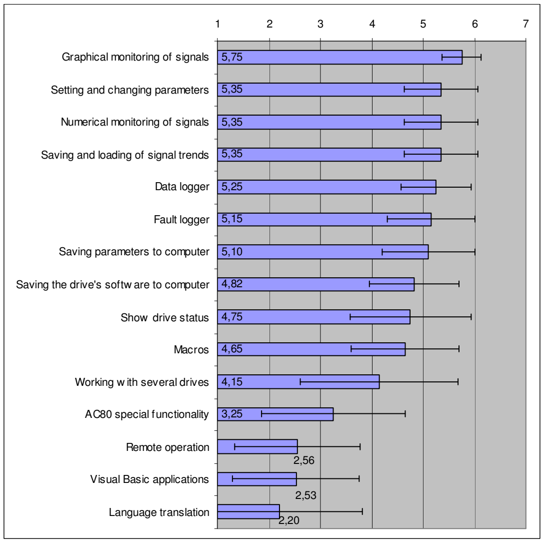


D8. Importance of DriveDebug functionality in my work tasks

(unimportant = 1, vital = 7)

N = 17 – 20

Generic drive tool functionalities rule with this software tool.



D9. Comments on working with DriveDebug and/or comparison with other similar software

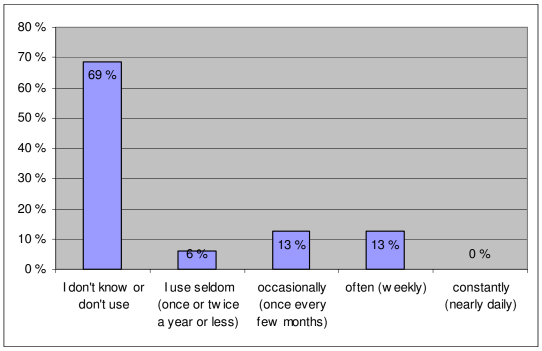
See Appendix G

6.3.5 Section E: DriveAP

E1. Usage

DriveAP 1.x

N = 16

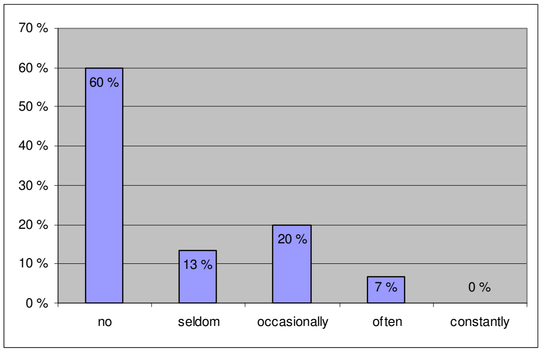


DriveAP 2.x

(same options as above)

N = 15

Neither version of DriveAP is very common among the respondents.



E2. Using DriveAP is

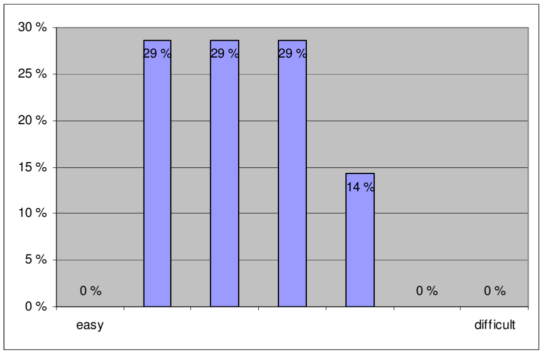
(easy = 1, difficult = 7)

N = 7

Average = 3,29

Standard deviation = 1,11

Average deviation = 0,90



E3. DriveAP looks

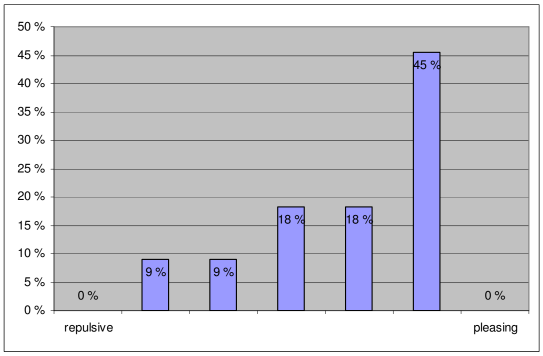
(repulsive = 1, pleasing = 7)

N = 11

Average = 4,82

Standard deviation = 1,40

Average deviation = 1,14



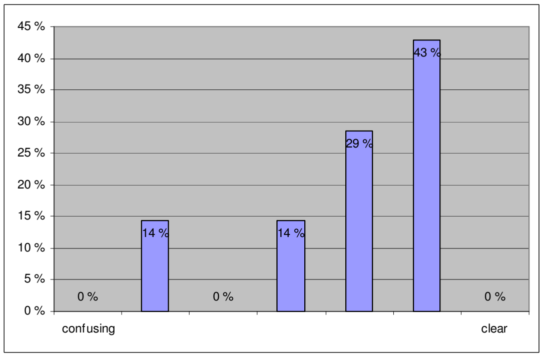
(confusing = 1, clear = 7)

N = 7

Average = 4,86

Standard deviation = 1,46

Average deviation = 1,06



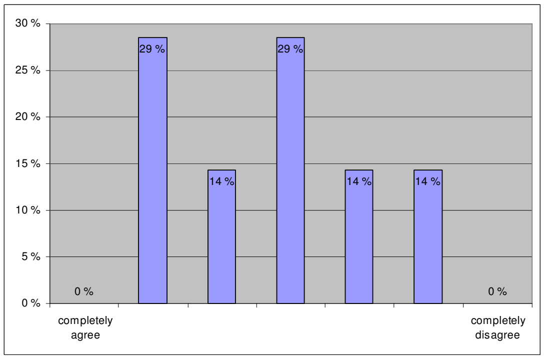
E4. Working with DriveAP is efficient
(completely agree = 1,
completely disagree = 7)

N = 7

Average = 3,71

Standard deviation = 1,50

Average deviation = 1,18



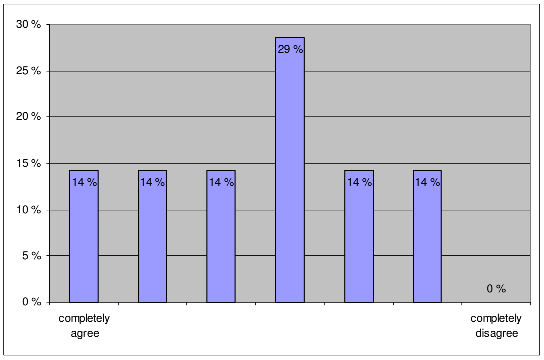
E5. I learned DriveAP quickly (completely agree = 1, completely disagree = 7)

N = 7

Average = 3,57

Standard deviation = 1,72

Average deviation = 1,35



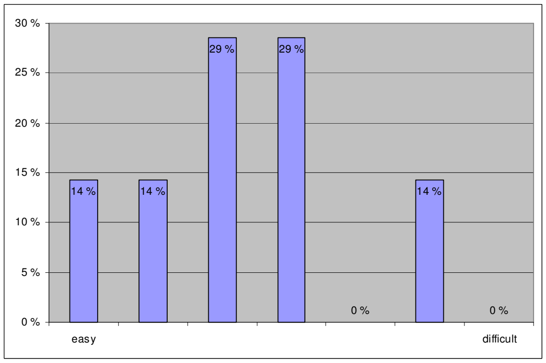
E6. AP block programming is (easy = 1, difficult = 7)

N = 7

Average = 3,29

Standard deviation = 1,60

Average deviation = 1,18



E7. Describe applications you have made with block programming

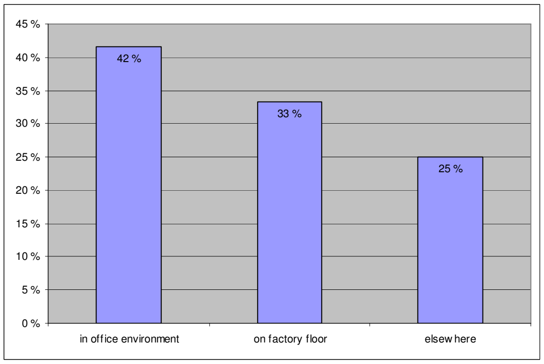
N = 5

"Winder and synchronizing", "crawl speed monitoring", "pump software", "line drives, about 20 blocks", "demonstrations".

E8. I use DriveAP

N = 12

Several alternatives from seven respondents. Electrics rooms were mentioned.

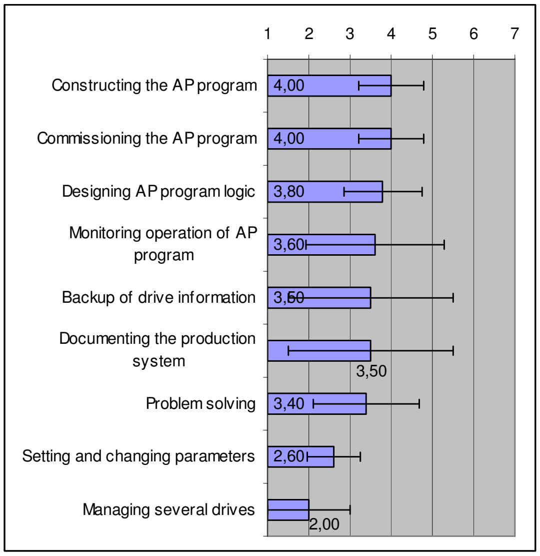


E9. DriveAP supports the following work tasks

(very badly = 1, very well = 7)

N = 4 – 5

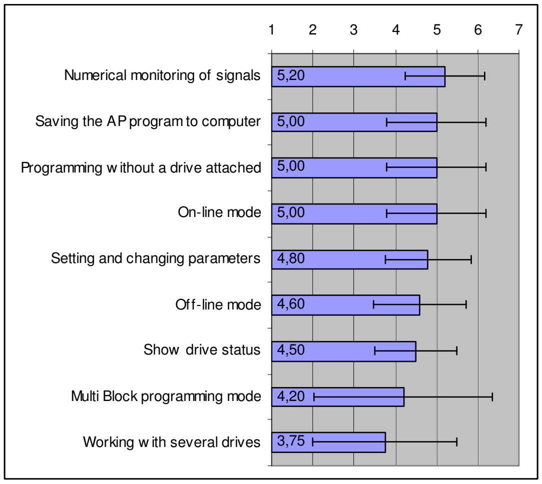
Most averages fall on the negative side although the middle ones show substantial disagreement.



E10. Importance of DriveAP functionality in my work tasks

(unimportant = 1, vital = 7)

N = 4 – 5



E11. Comments on working with DriveAP and/or comparison with others (like IEC 61131-3)

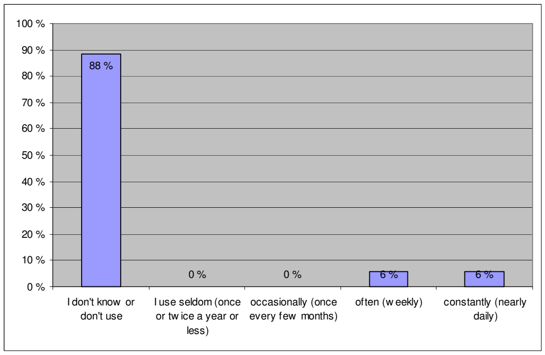
See Appendix G

6.3.6 Section F: DriveSize

F1. DriveSize usage

N = 17

Not widely used or known among the respondents. Those that do, use it frequently.



F2. Using DriveSize is (easy = 1, difficult = 7), N = 2

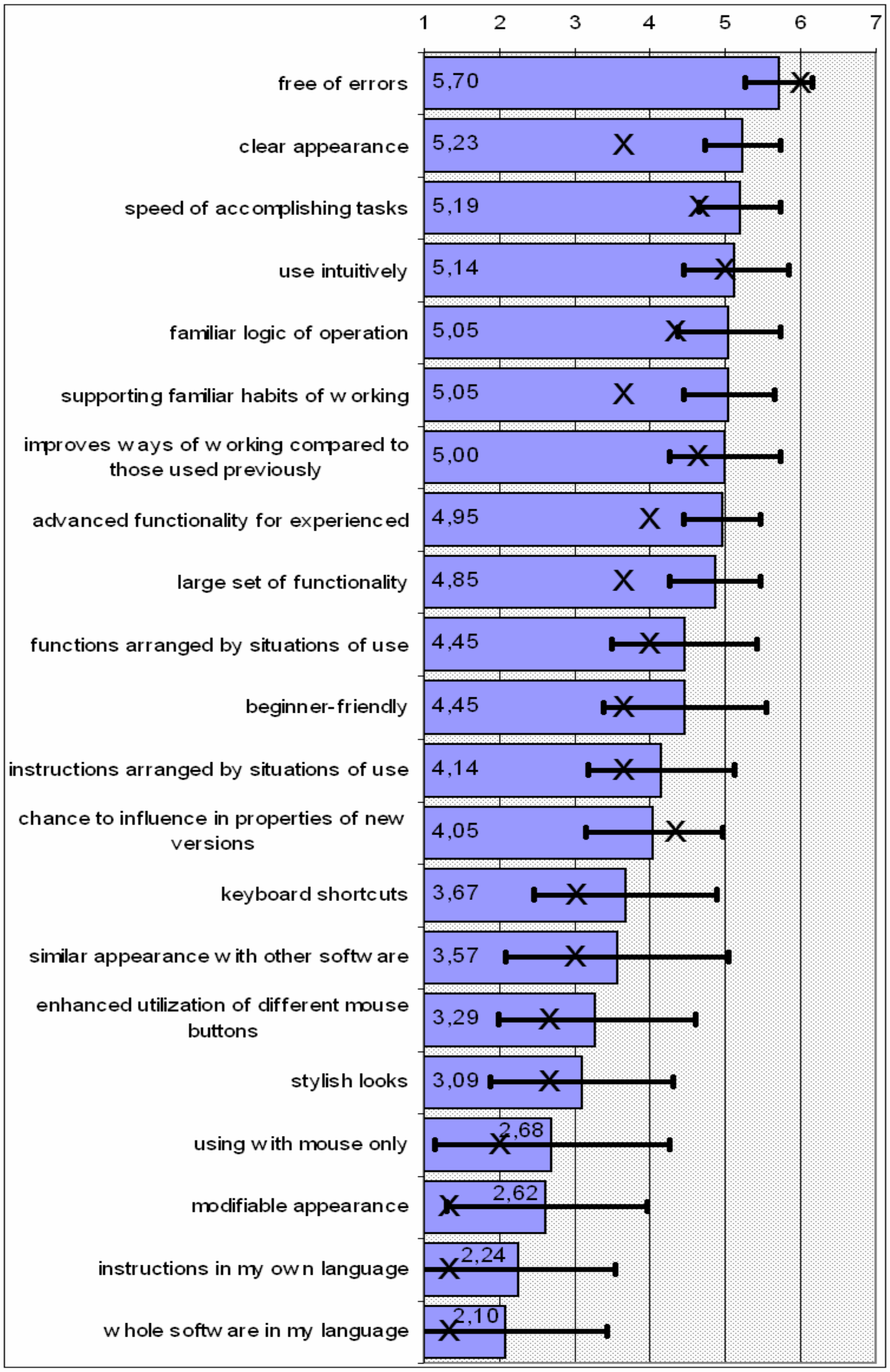
One respondent selected box 2, the other box 3.

F3. DriveSize looks (repulsive = 1, pleasing = 7), N = 2	One respondent selected box 5, the other box 6.
(confusing = 1, clear = 7), N = 2	Same as above.
F4. Working with DriveSize is efficient (completely agree = 1, completely disagree = 7), N = 2	Both respondents selected box 3.
F5. I learned DriveSize quickly (completely agree = 1, completely disagree = 7), N = 2	One respondent selected box 2, the other box 3.
F6. DriveSize calculations are trustworthy (completely agree = 1, completely disagree = 7), N = 2	One respondent selected box 4, the other box 5.
F7. I use DriveSize N = 2	Both respondents selected box “in office environment”.
F8. DriveSize supports the following work tasks (very badly = 1, very well = 7)	“Dimensioning technical alternatives”, “Choosing a motor”, “Choosing a frequency converter”, “Choosing a transformer”: two respondents, all answers boxes 4 and 5. “Designing application target”: one answer: 3. “Making purchase decisions”, “Documenting the production system”: no answers.
F9. Importance of DriveSize functionality in my work tasks (unimportant = 1, vital = 7) N = 2	“Graph display of selected equipment”, “Numerical values of selected equipment”, “Printing reports”, “Saving dimensioning information to a file”: both respondents answered 7. “Calculating electrical properties”, “Free selection of equipment from database”: one respondent selected box 7, the other 6 to all. “Exporting information to other applications”: one respondent selected box 7, the other 4.
F10. Comments on working with DriveSize and/or comparison with other similar software or other ways to do dimensioning.	”Trustworthiness of calculations has improved” ”The software doesn’t tell clearly why it selected the particular equipment”

6.3.7 Section G: General software expectations and experience

G1. Features I expect from drives’ PC software tools (1 = unimportant – 7 = vital)

N = 19 – 23



The tools team were asked the same questions also. Their answers kept pretty much in with the general line of user answers. In the above diagram the average answers of the three tools team respondents are marked with an X. Few notable differences rose up where the tools team underestimated the users' desires. Namely "clear appearance", "supporting familiar habits of working", "advanced functionality for experienced", "large set of functionality", and "modifiable appearance" were outside or just at the border of average deviation.

The users' answers show that they want stable and hassle-free applications that allow them to accomplish their work tasks swiftly in a way they are used to. They are not interested in bells and whistles or other gimmicks; clear appearance comes second only to being free of errors. However, they are also open for improvements if they support or enhance the aforementioned properties, especially the one ranked third, speed of accomplishing tasks. They also prefer having a single application with all needed functionality for the work. This doesn't exclude that the application could also be easy to use for everyone.

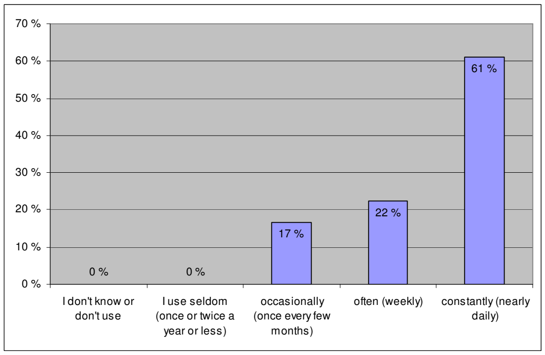
Language isn't an issue to these users. They are competent in English and years of working history has given them helpful routines and insights. They don't feel intimidated by computers which sometimes cause problems. Computer problems are accepted as necessary evils of today. These conclusions do not necessarily apply to other user groups.

G2. Experience with PC software (same options as with DriveWare® software)

Office (word processing, spreadsheets, etc.)

N = 18

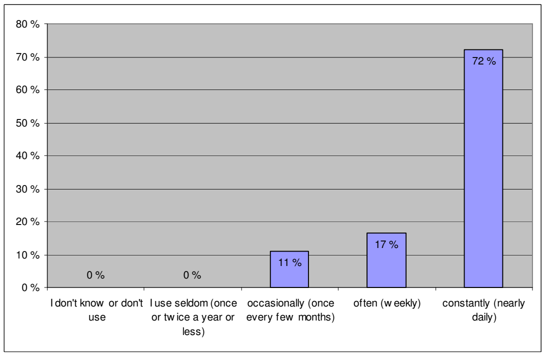
Everyone uses Office software, which makes it a good benchmark when designing interaction for new applications.



Web browser (Internet explorer, etc.)

N = 18

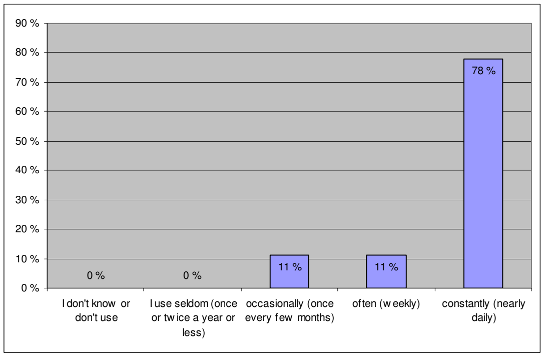
Even more familiar than Office. Presenting information (like online help) could lend design forms heavily from web technology.



Email (Outlook, Notes, etc.)

N = 18

Electric messaging is ubiquitous and should be taken into account when designing interpersonal communication aspects of software.



From competitor software Vacon NCDrive got one "seldom" and one "occasional". Control Techniques CTSoft had one "seldom". Allen-Bradley DriveExplorer, Telemecanique Powersuite, Siemens Starter, and Control Techniques CTSoft didn’t have any users. Others that got mentioned by users were: Omron, Siemens MasterDrives, AutoCAD, Adobe PDF Reader, FCB, and Control Builder.

6.4 User observation

Results of the user observations are reported here by a few different methods. From the training classes only the most interesting notes are given here. A more narrative story of the paper mill fault seeking was chosen. The paper mill commissioning gave enough data to construct a sequence model [7, page 96] of the activities. Specific problems with the software are collected into chapter 6.5.

6.4.1 Training classes

During the training classes some interesting user behaviours were observed. One user constantly scrolled a window by clicking an empty spot in the scroll bar instead of the arrows or dragging the slider. He also checked all modified values in the physical control panel of the drive to see that they were correct. He didn’t trust the program and he had experiences with older drives that went haywire when the connection broke and they were on remote control. Another user didn’t leave any measured values visible; he always closed them after viewing.

A middle-aged student used double click in the drop-down menus, which caused an underlying tool bar button to be depressed also. These kinds of mistakes by occasional computer users in the interface fundamentals do appear from time to time and should be anticipated corrected when encountered, especially in situations of education.

Scrolling by dragging the document with the mouse, like in Adobe Reader, appears to be a familiar technique to most. Several students attempted this unsuccessfully in DriveWare® products too.

6.4.2 Paper mill fault seeking

Paper mill environment

A paper mill has large, open factory buildings and the paper machines are large constructs sometimes over 100 meters long, over 10 meters wide and almost as tall. They are usually located in the middle of the floor with walkways and control consoles lining the sides. Control rooms and crew facilities are also next to the machine. Paper rolls are transported by fork lifts or automatic robots between storage and the machines. Crew comes out periodically from their rooms to exchange paper rolls and do other tasks the process requires.

Electrics rooms where the frequency converters are located don’t have a direct field of view to the paper machine. They are often a floor higher or lower than the machine. All equipment is locked in their own metal cabinets whose rows line the walls and passages. Bunches of cables run on trusses across the ceiling. There is a lot of noise from cooling fans, so ear protection is mandatory and conversation with regular voice impossible. The smell of ozone can be sensed.

The coating machine at this particular factory was smaller, about quarter or less in size compared to a machine that manufactures the paper from pulp to complete rolls.

The factory was lit with somewhat orange lights and there was the distinct smell of melting plastic from the coating process all around.

Proceedings

The symptoms of the problem were over-current faults given by four ACS600 drives that operate a rolling-out machine in front of a paper coating apparatus. The coating process was in full operation the whole time, which imposed some restrictions on the fault seeking activities. New paper rolls were fed into the rolling-out machine about twice in an hour. The machine had two axels for the rolls: while one is being fed into the process, the other can be repopulated so the coating process gets a constant stream of paper coming in. The faults occurred when the machine was changing from the almost emptied roll to the new one.

There was no maintenance history information available, but someone had been nosing around in the drives before, since some password-protected parameters were visible. A suspicion about the origin of the fault was that the control logic of the rolling-out machine instructs the drive to stop and start again immediately and the objective was to catch this behavior and prove that the error is not in any ABB equipment. A data logger of a drive showed a short stop but nothing further. A person from the rolling-out machine manufacturer (who called ABB to the scene) was also present and he mainly stayed by the machine and investigated its control logic with his laptop.

The application designer used DriveWindow 2 in his laptop which he had to connect individually to each drive since there was no optical network installed, only a field bus connection to the drives from the control logic. An ad-hoc optical network was not possible to make because all of the drives had the same ID number and it cannot be changed while running. He set Control word, Status word, Motor torque, and Speed measured to be recorded in one data logger and Maximum torque (from logic, 20.04), Limit word (3.04), Speed measured (2.18), and External reference (1.11) in another. DriveWindow Monitor he set up to draw Speed estimated (2.17), Speed measured (2.18), Speed reference 2 (2.01), Limited torque reference (2.13), Main status word (3.02), and Main control word (3.01).

The Control word is drawn a lot higher in the Monitor window than the speed graphs, which leads to a lot of vertical scrolling. The application designer checks the exact value of the Control word (displayed in base-10) at a visible dip, converts it with a pocket calculator into hex and remembers that it means a stop command. He copies pictures of the graphs (separate picture for speeds and torques, and the Control word) to a Word document and writes some comments. The pictures have to be scaled a few percent smaller to fit the page properly.

During lunch the application designer remembers that a start interlock in Digital input 2 and a mechanical start inhibitor lever might also affect the operation. These are checked during next paper roll change. There are no circuit diagrams in the electrics room so connected cables have to be checked visually.

The software tool of the control logic cannot draw graphs at 10 millisecond level so issuing the stop command cannot be verified there. The engineer from the rolling-out machine manufacturer set up a 100 millisecond delay into the control logic stop command, which eliminated it coming to the drives and the subsequent over-current fault. Normal operation was verified during a few more paper roll changes.

When everything appeared to be in order, graphs of the successful roll changes with comment were added to the Word document. The document is later sent as a report to the rolling-out machine manufacturer. The DriveWindow graph on the screen is also used when the situation is explained to the paper mill host. Finally, parameters of the connected drives are backed up to the laptop.

6.4.3 Paper mill commissioning

Operating environment

The physical factory environment of the mill is quite similar to the one described in the fault seeking. However, since the machine is being retrofitted, there are a lot of people from various companies moving about and doing different things and the normal production process activities are gone. Presence of other companies creates pressure for not delaying their work. No one wants their own company to look bad compared to the others.

In the electrics room, there are two computers built into the drive cabinets One runs the drive tools software, mainly DriveWindow and DriveDebug, and the other is for AC80 tool Application Builder and OPC server. Their keyboards, mice and flat screen displays are connected by Black Box KVM over twisted pair bridge and brought to a small table at one end of the narrow room. Builders are laying bricks to build a new wall and haul cement with a hand barrow. Electricians are laying out and connecting cables, installing the three cooler units in the ceiling and pulling cables under the floor through a small hatch right behind the table.

There are not enough good chairs to sit on. A laser printer box or a stool without a backrest has to suffice when all team members are working in the same room. There are a couple of shelves above the table that contain binders of documentation, but the table is too small to have many of them open at the same time with the keyboards, displays, mice, and an odd laptops of the team members.

The entire paper machine can have as many as 50 to 150 line drives. The calender machine that was being installed had 27 that were driving 33 rolls. There were spelling mistakes in the cabinet door signs so matching them with paper documentation took a little deduction. The situation can be much worse with foreign languages overseas.

All people are supposed to be present at the factory by 8 a.m. Working days of the ABB commissioning team last for 10-12 hours. In the team there are three people one of whom is the team leader. One person usually operates the computers while one, “telajulli”, moves around the paper machine to appropriate locations to see they are clear for operating. They communicate via walkie-talkie radios.

There is a morning meeting every day at 9 a.m. in an office next to the paper machine where representatives of all parties involved in the project are supposed to be present. In the meeting the project manager of the factory gets updated on the subcontractors' progress. Communication is conducted mainly in Finnish but questions to foreign people are asked in English. In addition the local factory hosts do a round or two a day in various facilities to survey the situation. Internal commissioning team decisions on next work steps are done informally at lunch or coffee breaks.

In the mornings there is not much that can be done with the motors or rolls, because the mechanics are working on them. The pace gets faster after midday and more complicated things can be done after regular day workers leave (unless something is needed from them...). Information flow goes mainly from the subcontractors to the mill hosts, not laterally. One has to be active to find out exactly what others are doing.

Proceedings

The following is a Contextual Design sequence model [7, page 96] of the observed commissioning activities complemented with information from interviews. The red lightning bolt (⚡) shows a breakdown or problem in doing the steps. Individual steps are described in more detail afterwards.

Intent: Get the mill operational

Trigger: Commissioning schedule, physical installation and connections done

↓

Check electric connections

↓

Check optical network

↓

Set basic parameters

↓

Empty runs

↓

Intent: Have a baseline that can be reverted to.

Take backups

↓

Set speed scales and other advanced parameters

⚡ Incorrect roll radius measurements and gear ratios from the machine manufacturer.

↓

Intermediate trigger: Automation system subcontractor asks

↓

Empty test runs for mechanics checks

↓

Intermediate trigger: Roll manufacturer asks

↓

Test runs one roll at a time

↓

Intermediate trigger: Mechanical connections and lubrication done by mechanics, safety switches and locks are off

↓

Intent: Tuning the speed controllers.

Stabilization runs (repeated to all rolls)

⚡ Forced to stop some rolls because mechanic issues weren't in order

⚡ Connection to a roll was lost temporarily because probably a cable was disconnected

↓

Running with automation software

↓

Running with factory control panels

↓

Intermediate trigger: Machine ready for full process

↓

Process runs

↓

Intent: Save a configuration where everything works.

Take backups

Details of the model steps

After electricians have done the physical installation and connections they have to be checked. That includes:

- Motor power cables in/out, auxiliary power, I/O cables.
- Emergency stop circuits testing with higher level automation software: AC80 Application Builder / Function Chart Builder. One person operates the stop switches at the paper machine, one checks wire terminals at the drive cabinet, and one checks readouts at the computer.
- Safety gate testing: DriveDebug (Fault/Alarm words i.e. parameters 1.10, 9.1-9, 70.15 in a window) and FCB windows tiled, one person looks in a binder what should be tested, one monitors the programs, one operates the gates at the paper machine

The testing is continued in parallel with other activities as time permits until done completely.

Checking the optical network involves:

- Turn on power to one drive, see what shows up in program (DriveDebug).
- Set ID setting of CRMIO board if it is not correct. It has to be individual and according to documentation. Connection is made point to point with DriveWindow, because it is faster when setting just one parameter.

Setting basic parameters means inputting motor values which are found in documentation or physical plates on the motors. The plate values are written into a small notebook and then brought down to the electrics room for inputting.

Empty runs are done with motors mechanically disconnected from their load. One team member, the "telajulli", radios in from next to the motor which is ready (safety switches and locks are off). The following steps ensue:

- Run slowly at reference value 20.
- Check direction of rotation (radioed in) and whether or not marked correctly to documents.
- Do an ID run.
- Mark ID "ok", rotating direction, safety switch "ok", and temperature ohms to notebook.

Setting speed scales, protective values, and other advanced parameters like temperature sensor types is done using ready-made DriveDebug macros for data sets comprising of about 90% of the parameters, about 15% of which get individually modified:

- A set of about 40 parameters to 90, 91, 92, and 93 groups to get the drive to communicate with the outside systems. There is another data set macro for zeroing these.
- One macro sets parameters 13.03, 18.01, 18.02, 23.06, 24.02, 24.03, 24.09, 30.19, 30.23, 30.24, 30.28, 30.29, 50.06, 50.11, 50.13, 50.14, and 70.04. The macro is run a couple of times successively from a keyboard shortcut, because 50.14 didn't change value at first try.
- Value of one parameter is checked from all drives by opening a Monitor Window where it is included and then clicking through all drives in the Target Window.
- One macro sets acceleration and deceleration times, minimum and maximum speeds etc. (parameters 20.7-11, 22.1-4, 16.8).
- The tachometer values of a roll are checked:
    - Excel in a laptop is used to calculate speed scalings with a formula incorporating gear ratio, roll radius and maximum speed. Even small decimals count, because the speed display of a follower roll can show wrong and then paper men claim that the roll is sliding even though a bunch ("nippi") of the rolls rotate at the master's speed because of high pressure
    - Roll radius measurements and gear ratios received from the machine manufacturers cannot be trusted 100%. They are hard to obtain, there are changes, and blueprints might have errors like missing a gearbox.
    - Trends are drawn of signals 2.03, 107, 2.09, 23.10, 1.04, and 1.03.
    - Motors are rotated: 10 rpm ➔ 30 rpm ➔ 500 rpm ➔ stop.
    - 10 rpm (for 10-20 seconds) ➔ 500 rpm ➔ stop ➔ do adjustments ➔ 500 rpm ➔ stop ➔ adjust ➔ 300 rpm ➔ stop ➔ adjust ➔ 500 rpm ➔ stop.

- Some parameters are exported to a .TXT file, it is renamed to .MAC in File Manager and the parameters are inputted to 13 drives by changing the target number in the macro.

An automation system subcontractor asked to do a few empty test runs so mechanics can see that everything is in order; drive belts don't touch anything etc.:

- rotate at 10 rpm ➔ accelerate to 20 rpm ➔ 8 rpm step test a few times ➔ accelerate to 40 rpm ➔ accelerate to 500 rpm ➔ stop.

The roll manufacturer asked to do some test runs one roll at a time that included some step tests. These had to be done with DriveDebug, because the automation system would rotate all rolls at once. The manufacturer's mechanics do some checks of their own on the rolls.

After mechanical connections and lubrication is done by mechanics, safety switches and locks are off, the stabilization runs can be done to tune the speed controllers and see which rolls need the longest acceleration/deceleration times and then set them to everyone. Graphs are drawn at various speeds, and saved as bitmaps with comments:

- Rotate at 15 rpm ➔ step test ➔ 50 rpm ➔ step test ➔ 155 rpm ➔ 450 rpm (for about a minute) ➔ 775 rpm (~minute) ➔ 1160 rpm (~1,5 minutes) ➔ 1550 rpm ➔ step test ➔ save "at 100%" ➔ 1160 rpm ➔ step test ➔ save "at 75%" ➔ 775 rpm ➔ step test ➔ save "at 50%".
- Repeat previous step to all rolls.
- Rotating some rolls were forced to stop ("telajulli" radioed in) because mechanic issues weren't in order after all: oil or water spilled out or hydraulics weren't operational. Another problem came when connection to one roll was lost temporarily because probably a cable was disconnected. DriveDebug was able to continue operation as the link returned, DriveWindow would have had to be restarted and all settings redone.
- Some rolls are stiff and don't want to accelerate even at 100% torque. Maximum torque is set temporarily to 150% (it has to be remembered to return to 100%) ➔ rotate ➔ roll loosens up little by little.
- Rolls are run in groups: "nippi kiinni", 2-4 rolls coupled mechanically together, master/slave configuration.

The speed controllers are optimized with the step tests that are usually done at 1% of maximum speed. The optimization is a merry-go-round between vibration, filtering and gain. If too aggressive, the machine bangs to the gaps in the gear box and causes torsion vibration in the axel. Too much filtering produces a bad step response. The more gain there is, the faster the motor follows the speed reference.

Running with automation software, running with factory control panels, and process runs were done after the researcher left the mill. They also include drawing graphs of paper web tension measurements, which are made when the machine is in production. They require rapid switching from drive to drive.

Backups are taken of parameters in a few of the stages. Complete backups of all drives are done at the end with DriveWindow. DriveDebug is used after the ID run.

6.5 Found usability problems and suggestions for improvement

Usability problems of the commissioning software tools the end-users were observed to face and that were found by the researcher are listed here. The problems are listed per program and their severity is ranked with the Nielsen (0 – 4) and Sinkkonen (a – d) scales (see chapter 4.4.5). Also any violated Nielsen heuristics (see chapter 4.4.1) are shown.

6.5.1 DriveWindow

<table>
<tr><td><b>DW1. Browse tree pane navigation</b></td><td><b>Severity:</b> 3 c</td></tr>
<tr><td><b>Observed at:</b> paper mills</td><td><b>Violated heuristics:</b> 3, 5, 6, 7, 8</td></tr>
<tr><td colspan="2"><b>Description:</b> With several target drives and even within one drive, the users have to jump up and down in the tree and click open and close the different hierarchy levels of parameter groups. Locations of the items of interest create a memory burden. User quote:” Tossa saa olla tarkkana, että menee oikeeseen.”</td></tr>
<tr><td colspan="2"><b>Suggestions:</b> Replace tree with a structure that displays all targets more equally, i.e. not occluded by other hierarchy levels. The approach could resemble the Finder application of Apple Macintosh MacOS X. Also context-sensitive pop-up menus and drag&drop type direct manipulation between separate windows could be utilized.</td></tr>
</table>

<table>
<tr><td><b>DW2. Step Test regarded unusable</b></td><td><b>Severity:</b> 3 c</td></tr>
<tr><td><b>Observed at:</b> paper mills</td><td><b>Violated heuristics:</b> 1, 2, 5, 6, 7, 10</td></tr>
<tr><td colspan="2"><b>Description:</b> Users feel that the Step Test function is unusable because it doesn’t bypass speed ramps. Doing step tests is an important phase in commissioning and the lack of proper functionality means that the entire tool is unusable.</td></tr>
<tr><td colspan="2"><b>Suggestions:</b> From the users’ perspective this should work like in DriveDebug. Implement an automatic setting of speed ramps to zero before the step and restoring after.</td></tr>
</table>

<table>
<tr><td><b>DW3. Invisible backup package</b></td><td><b>Severity:</b> 3 c</td></tr>
<tr><td><b>Observed at:</b> training</td><td><b>Violated heuristics:</b> 1, 2, 4, 5, 6, 7, 9</td></tr>
<tr><td colspan="2"><b>Description:</b> Existence of an opened backup package is not indicated anywhere. Users get confused when they are prompted to save changes to an open package.</td></tr>
<tr><td colspan="2"><b>Suggestions:</b> Show an open package and provide direct access to its contents in the interface (e.g. in the target structure). Alternatively, integrate the opening and closing operations into the actual act of doing or restoring backups.</td></tr>
</table>

<table>
<tr><td><b>DW4. Trend display graph scaling</b></td><td><b>Severity:</b> 3 c</td></tr>
<tr><td><b>Observed at:</b> training, paper mills</td><td><b>Violated heuristics:</b> 1, 2, 4, 5, 7, 8</td></tr>
<tr><td colspan="2"><b>Description:</b> Scaling the monitor graphs of the Trend display pane is done by entering a coefficient and an offset to each signal. These cannot be done intuitively and require a calculator to help. The graph Y axis has single values shared by all signals. This also distorts the mental model of the actual values of the drawn signals.</td></tr>
</table>

Suggestions: A direct entry of maximum and minimum values for each signal, like in DriveDebug, is better. The Y axis should indicate which signal its values apply to.

DW5. Trend display scrolling	**Severity:** 3 c
Observed at:	**Violated heuristics:** 2, 3, 4, 5, 7, 8
Description: The vertical scroll bar's and buttons' location is not standard and they are visible at all times whereas the horizontal appear only when needed. The scroll box doesn't scale with the window contents.	
Suggestions: Redesign the layout and functionality to follow platform conventions.	

DW6. Trend display cursor values	**Severity:** 3 c
Observed at: paper mills	**Violated heuristics:** 1, 4, 5, 6, 7, 8
Description: The cursor function displays the numerical values of signals overlapping when the graphs are close to each other. This fault is actually documented in the manual as if it were acceptable. Selection of whether the displayed values are scaled or actual is dispersed between File and View menus.	
Suggestions: Implement displaying the values so that they don't overlap. Scaling selection could be per signal and accessible by a context-sensitive pop-up menu when clicking the secondary mouse button on the value (direct manipulation). Redesign the menu structure.	

DW7. Browse tree missing expandability marks	**Severity:** 3 b
Observed at: training	**Violated heuristics:** 1, 2, 4, 5, 6, 7, 8
Description: When DriveWindow has first established contact with drive targets, the + signs indicating expandability are missing from the tree. Double-clicking the target does expand the next hierarchy level and after this the + and – signs appear normally.	
Suggestions: Display the + sign from the beginning so the interaction is consistent. The tree could expand from a single click on the target too, since the latest versions of Windows Explorer work like this.	

DW8. Trend display pane resize scales content	**Severity:** 3 b
Observed at:	**Violated heuristics:** 1, 3, 4, 5, 7, 8
Description: When the trend display pane is resized, its contents, including axis values and scroll bars, are all scaled. This makes the window hard to read in small sizes and no extra information fits when it is large, if the axis scaling is not changed in the trend settings pane.	
Suggestions: Make pane contents scaling separate from the size of the pane. Especially widgets like the scroll bars and buttons should not change their size.	

DW9. Measured parameters choice lost	**Severity:** 2 b
Observed at: paper mills	**Violated heuristics:** 1, 3, 4, 5, 6, 7
Description: When the user has set some parameter values to be monitored numerically and jumps to another parameter group and back, the choices disappear and have to be redone.	

Suggestions: The program should remember the state of all parameters at least as long as there is a connection to the same target. A multi-window solution (like DriveDebug) could be more flexible.

DW10. Item list pane user control	**Severity:** 2 b
Observed at: paper mills	**Violated heuristics:** 4, 7
Description: The user can navigate in the parameter list with the arrow keys of the keyboard, but no parameters can be modified without clicking on them with the mouse or selecting a menu item.	
Suggestions: Enable user interaction in all places with the keyboard too. Choose default commands for all interface items and navigation between them. Pressing enter on a parameter could bring up its modification window.	

DW11. Connection to drive unclear	**Severity:** 2 b
Observed at: training	**Violated heuristics:** 1, 2, 3, 5, 6, 7, 8, 9
Description: The OPC server layer between a drive and DriveWindow is incomprehensible to regular users not educated in telecommunication and network technologies. When the connection to a drive is lost, the users restart the whole program and press enter in the connection pop-up window rather than select Network / Disconnect server and Network / Network Servers to bring up the same dialog.	
Suggestions: Connecting to another OPC server than the local one is very rare. Re-establishing contact should be made very simple with defaults, a single tool bar button, and renovated menu structure. Other connection options could be hidden further away.	

DW12. Local / remote mode confusion	**Severity:** 2 b
Observed at: training	**Violated heuristics:** 1, 2, 5, 6, 7, 10
Description: Users were confused why some functions were grayed out at times. They didn't associate the behavior of the program to the choice of local / remote control, which in DriveWindow is actually called Take/Release Control.	
Suggestions: Use consistent vocabulary in all software and physical control panels. Since only one drive at a time can be a target and under the control of the software, the default position of the DrivePanel (Control Panel or Drive Control Panel in other software) could be closer to the target list or integrated into it with a different design than the tree structure.	

DW13. Similar tool bar buttons	**Severity:** 2 b
Observed at: training	**Violated heuristics:** 1, 4, 5, 6, 7, 8
Description: Some buttons in the four dockable tool bars have identical pictures on them so the user has to remember their locations or wait for and read the tool tip. It is easy for novices to select a wrong button by mistake.	
Suggestions: Renovate the tool bars with more appropriate pictures. A show/hide mechanism with visible textual headings could be implemented. In a multi-window interface the buttons could reside only in the associated window.	

DW14. Reference value disappears	**Severity:** 2 a
Observed at: training	**Violated heuristics:** 1, 2, 4, 5, 6, 7
Description: The reference value edit field in the DrivePanel is cleared often and has to be retyped. The user has to jump between the mouse and the keyboard a lot.	
Suggestions: Implement a history drop list to the edit field so previously entered values can be selected quickly with the mouse.	

DW15. Step Settings window uninformative	**Severity:** 2 a
Observed at: training	**Violated heuristics:** 1, 2, 5, 6, 10
Description: When there is no connection to a drive, the reference value edit field is not displayed in the DrivePanel, and Step Settings is the only active item in the Drive menu. Novice users thought this is the place to enter the reference value to get the motor rotating.	
Suggestions: Revise the procedure of connecting to a drive. Rethink naming conventions to be consistent among different products like renaming DrivePanel to Control Panel and Drive menu to Control accordingly. Is it necessary for the Step Settings to be active when there are no target drives? An illustration of the step function's effect could be added to the Step Settings window. Also the layout of the window needs rethinking because now the contents of the Reset group box affect the contents of the Step group box.	

DW16. Editing Trend settings pane values not flexible	**Severity:** 2 a
Observed at:	**Violated heuristics:** 4, 5, 6, 7
Description: Values in the Monitor and Datalogger tabs in the Trend settings pane can be edited by double-clicking their labels in the Settings column or through Monitor menu and its submenus. The Value column produces a rude error beep.	
Suggestions: Make the Value column selectable too (direct manipulation). Enable edit-in-place (see User Interface Design Patterns, [27])	

DW17. Obscure naming conventions	**Severity:** 2 a
Observed at: training	**Violated heuristics:** 2, 4, 5, 6, 8
Description: DriveWindow is plagued by a plethora of proprietary terms and vocabulary, like workspace, desktop, control item set, the panes, and in the tool bars. Their meaning and connections to the interaction elements are not obvious as some terms are visible only in the manual. Some functions are named differently in the tool bar than in the menus (Activate/Deactivate Items vs. Desktop / Put Items Online/Offline).	
Suggestions: Use consistent names within the software and with others. Revise menu structure. Revise help and manual.	

DW18. Signal value format choice not found	**Severity:** 2 a
Observed at: paper mills	**Violated heuristics:** 1, 2, 4, 5, 6, 8
Description: A user used a calculator to convert a signal value to individual bits even though DriveWindow provides this functionality. The display format of the value of an individual signal can be changed by opening a dialog from the View menu.	

Suggestions: The View menu is not usually associated with single and small interface elements. Revise menu structure so that the format choice is near the Change Item Value menu item. Enable direct manipulation by bringing these functions to a context-sensitive pop-up menu that opens when the parameter or signal or its value is clicked with the secondary mouse button in the Item list pane.

DW19. Compare window filtering	**Severity:** 1 a
Observed at: training	**Violated heuristics:** 1, 2, 3, 5, 6
Description: The Compare function displays only the parameters that are different including also those that are normally hidden from the user (group number over 100). Some users wanted to see a list of all parameters with the different ones highlighted.	
Suggestions: Provide filtering options to the parameter lists. Seeing all parameters helps the user to match the information to the real world.	

DW20. Odd default window size	**Severity:** 1 a
Observed at:	**Violated heuristics:** 3, 4, 5, 7, 8
Description: When DriveWindow starts, the window fills the whole screen but it is not maximized. This looks odd since normally non-maximized applications fill only a portion of the screen.	
Suggestions: Make being maximized the default. In a maximized application the vertical scroll bar can be operated faster by slamming the mouse cursor to the screen edge instead of carefully placing it a few pixels inside into the scroll box (see Windows Explorer). Seasoned computer users take advantage of this kind of accelerators almost unknowingly and their absence is irritating.	

6.5.2 DriveWindow Light

DWL1. Wasteful use of screen real estate	**Severity:** 3 c
Observed at:	**Violated heuristics:** 4, 8
Description: The toolbars and panels have few items in them, but they reserve the whole width or height of the window leaving a lot of unusable grey space. Relevant content is crowded in a display with small resolution. Also the wizard screens waste space.	
Suggestions: Make the toolbars and panels dockable. Have a user interface professional design new screen layouts.	

DWL2. Awkward graph zooming	**Severity:** 3 c
Observed at: training	**Violated heuristics:** 2, 3, 4, 5, 6, 7, 8
Description: The Zoom Selected Area button in the Drive Monitor window initiates a zooming mode that can be exited only by pressing the same button again. The scroll bars are hidden and another region cannot be selected without exiting first.	
Suggestions: Remove Zoom Selected Area button, make the scroll bars operational and a region selectable at all times, and zoom into it with the regular Zoom In button. An Undo command could return to a previous zoom factor or the buttons could have a drop-down selection for this.	

DWL3. Confusing dialogs	**Severity:** 3 b
Observed at: training	**Violated heuristics:** 2, 5, 6, 8, 9, 10
Description: After selecting a language in the start-up wizard the users are faced with a dialog asking “You are about the change the language of the drive. Would you like to reload parameter structure of the drive?” and the novices didn’t know what to answer. The term reload doesn’t indicate a direction and the outcomes of the choice aren’t obvious.	
Suggestions: This kind of dialogs, especially in a wizard intended for novices, should be explicit in their information, explain what happens with the different choices, and offer links to further help and directions.	

DWL4. Number of wizard steps unknown	**Severity:** 3 b
Observed at:	**Violated heuristics:** 1, 3, 4, 5, 6, 7, 10
Description: The user has no way of knowing how many wizard steps there are left and where he is in the process.	
Suggestions: Number the wizard pages and display also a total next to the navigation buttons. An index of screen topics could be displayed and used also for navigation if there are steps that can be skipped.	

DWL5. Compare window stays on top of others	**Severity:** 2 c
Observed at:	**Violated heuristics:** 1, 3, 4, 5
Description: The Compare window is displayed as the topmost window even if it doesn’t have focus. It occludes other windows and it doesn’t show up in the Window menu.	
Suggestions: Make the Compare window either modal or equal with the other windows and visible in the Window menu. Equality is preferred especially if a drive’s parameters are to be modified by the results.	

DWL6. Drive Monitor window buttons	**Severity:** 2 b
Observed at: training	**Violated heuristics:** 1, 2, 3, 4, 5, 6, 7, 8
Description: The Numerical Format button opens an unlimited number of docked numerical windows into the Drive Monitor window. They can fill up the window but there is no scroll bar to navigate to the graph section or the ones “under the window frame”. A user tried to close the numerical windows with the Auto Scroll button located next to the opening button.	
Suggestions: The numerical windows could be resizable and their number limited when they don’t fit into the Drive Monitor window. The number could be displayed in their title bars when undocked. The Auto Scroll button could be placed earlier in the tool bar.	

DWL7. Navigation between windows	**Severity:** 2 b
Observed at: training	**Violated heuristics:** 1, 2, 3, 4, 5, 6, 7, 8
Description: When the sub windows are maximized, the user can easily switch from Parameter Browser window to the Drive Monitor window by pressing the Monitor function button in the Status Panel. He can’t go back to the Parameter Browser the same way, which is	

annoying.
Suggestions: Add a Parameters button which doubles as starting a New Online/Offline Drive function. Consider renovating the whole process of connecting to a drive.

DWL8. Robustness in error situations	**Severity:** 2 b
Observed at: training	**Violated heuristics:** 1, 2, 3, 5, 9
Description: A demo case was powered off while the Relay outputs phase of the Wizard was open. This resulted in repeated "Time Out Error while writing register 14xx" error messages.	
Suggestions: Software developers should anticipate communication errors and offer users constructive solutions to the problems in the error messages.	

DWL9. Unnecessary error messages	**Severity:** 2 a
Observed at:	**Violated heuristics:** 1, 2, 3, 4, 5, 9
Description: When Drive Monitor window is active and the user selects File / Open in Monitor and presses Cancel, a "File not found" error message appears. A user might get the impression that he did something wrong.	
Suggestions: Superfluous error messages should be eliminated with careful design and testing. Canceling an action should not result in an error message.	

6.5.3 DriveDebug

DD1. Relation between target and Control Panel	**Severity:** 3 c
Observed at: paper mills	**Violated heuristics:** 1, 2, 4, 5, 6, 7
Description: When a target drive is changed in the Target List, Control Panel stays with the old target until Loc/Rem is pressed even though other windows follow the active target. A user thought that a motor didn't start rotating because the trend graphs didn't change. Starting a wrong motor could result in damages or injury if someone is working on the mechanics.	
Suggestions: The drive associated with the Control Panel could be marked in the Target List. Changing a target while Loc/Rem is in the wrong mode could prompt the user with a message and a choice of changing the mode and which one to follow.	

DD2. Window and item focus	**Severity:** 3 c
Observed at: paper mills	**Violated heuristics:** 1, 3, 4, 5, 7, 8
Description: The target list is an equal window compared to content windows like MonWin and only one window has focus. A user tried blindly to input a value after selecting a target but his keystrokes didn't have an effect.	
Suggestions: A piece of software shouldn't make users look or feel foolish. The target list could be a special window that is "active" together with the content windows. It could be a part of the frame window like a dockable panel.	

DD3. Control Panel unprotected	**Severity:** 3 c
Observed at: paper mills	**Violated heuristics:** 2, 3, 5

<table>
<tr><td><b>Description:</b> Control Panel buttons are vulnerable to accidental activation. The power cord of a laptop pressed the spacebar when focus was on the Stop button. Construction people were working above the computer when it was left alone and DriveDebug was running. Something might have been dropped on the controls and start a motor.</td></tr>
<tr><td><b>Suggestions:</b> Extend factory safety regulations also to computer software. A safety button could be implemented that enables the Start button only for the next five seconds. On the other hand an emergency stop should always be available so that the software doesn’t block access to it with a modal window or an extended dialogue. It could also be “covered” by a safety button. The Control Panel could be a dockable panel or integrated into the target list.</td></tr>
</table>

<table>
<tr><td><b>DD4. Unorthodox layouts and components</b></td><td><b>Severity:</b> 3 c</td></tr>
<tr><td><b>Observed at:</b> paper mills</td><td><b>Violated heuristics:</b> 1, 2, 3, 4, 5, 6, 7, 8, 9</td></tr>
<tr><td colspan="2"><b>Description:</b> Information is presented in many windows on a plain background with space character delimitation. Textual information of varying lengths breaks up these kinds of columns and spaces in the text add to the confusion and cluttered looks. Other programs show the parameter name before its value. Information and component grouping in several windows (e.g. Parameters, Fault Logger) is done oddly.</td></tr>
<tr><td colspan="2"><b>Suggestions:</b> Use standard list controls for displaying information. They can be scaled and the columns reorganized by the user at his will. Follow platform conventions and standards for component layouts as well.</td></tr>
</table>

<table>
<tr><td><b>DD5. Information conveyed by background colors</b></td><td><b>Severity:</b> 2 c</td></tr>
<tr><td><b>Observed at:</b> paper mills</td><td><b>Violated heuristics:</b> 1, 2, 4, 5, 6, 8</td></tr>
<tr><td colspan="2"><b>Description:</b> The Target List shows the drive statuses by different colors for the text and backgrounds. The user has to remember what different colors and combinations mean. Some color combinations are hard to read like grey on red. A user couldn’t explain why non-faulty drives’ background turns from green to yellow. A big window flashing red and yellow feels intimidating and adds stress when working under time constraints.</td></tr>
<tr><td colspan="2"><b>Suggestions:</b> Express fault status more subtly in a column of its own next to the target name. A faulty drive that isn’t visible without scrolling the target window can be indicated with an arrow pointing towards it (perhaps utilize the scroll bar arrow) or by coloring the upper or lower inside edge of the window. Remove the color when there are no more faults in that direction.</td></tr>
</table>

<table>
<tr><td><b>DD6. Saving trend graphics clumsy</b></td><td><b>Severity:</b> 2 c</td></tr>
<tr><td><b>Observed at:</b> paper mills</td><td><b>Violated heuristics:</b> 1, 2, 3, 4, 5, 6, 7</td></tr>
<tr><td colspan="2"><b>Description:</b> The field for entering a comment was too short. Clipboard and Notepad had to be used to produce regular form comments. The saved picture has to be opened in another application to see how it turned out. Mistakes or typos cannot be easily corrected afterwards; the whole process has to be redone.</td></tr>
<tr><td colspan="2"><b>Suggestions:</b> Provide a preview whose fields can be edited in place. Allow different placements of the legend. Suggest a filename based on the name of the drive (perhaps from parameter 99.11). Provide a comment history drop-down. Allow to add other parameter values to the picture.</td></tr>
</table>

DD7. Other windows freeze when trending	**Severity:** 2 c
Observed at: paper mills	**Violated heuristics:** 1, 2, 3, 4, 5, 6, 7, 8
Description: Trend Window monitoring has to be stopped to be able to operate other windows. Target list is not updated so error information is not received in time. When a parameter is changed, its effect cannot be observed immediately in the graphs. TrendWinHelper is a bubblegum patch.	
Suggestions: Take advantage of platform's multitasking capabilities.	

DD8. Antiquated file dialogs	**Severity:** 2 b
Observed at: paper mills	**Violated heuristics:** 1, 2, 3, 4, 5, 6, 7, 8
Description: Window 3.x style file (saving) dialogs are antiquated and unfamiliar to users today. They don't show the file contents of the current folder so File Manager or Windows Explorer has to be used in parallel to seek out a file to be overwritten.	
Suggestions: Use contemporary dialogs provided by standard libraries or extend them when necessary. Follow platform conventions.	

DD9. Comma or point	**Severity:** 2 b
Observed at: paper mills	**Violated heuristics:** 2, 3, 5, 7
Description: A user is used to entering numbers from the number keypad portion of the keyboard, but when modifying a parameter, a point is needed. This causes jumping between different portions of the keyboard.	
Suggestions: Convert a comma from keypad automatically to a point in parameter numbers.	

DD10. File and folder names are shown in 8+3 format	**Severity:** 2 b
Observed at: paper mills	**Violated heuristics:** 2, 4, 5, 6, 7
Description: Longer file and folder names are not shown fully in e.g. window title bars. The user has to interpret the short names to the ones he sees in File Manager or Windows Explorer.	
Suggestions: Support long file and folder names. Make sure all information presented to the user is in a familiar format.	

DD11. Window contents don't scale when resized	**Severity:** 2 b
Observed at:	**Violated heuristics:** 4, 6, 7, 8
Description: Most windows are resizable but their contents don't resize or scale or scroll bars don't appear. Some information or their changing might be lost when the window has been made smaller and they are not visible. In larger screen resolutions the Trend Window is very small and harder to present to others.	
Suggestions: Remove resizability from windows that are not designed to be scaled. Better would be to make the contents conform to the window size. Anchor some components like buttons to selected window edges and resize others like text fields, graphs, or lists. Follow platform conventions.	

DD12. Signal scaling in Trend Window	**Severity:** 2 a
Observed at: paper mills	**Violated heuristics:** 1, 2, 4, 5, 6, 7
Description: Dialogues for setting the signal Trend Y Value at upper and lower limits and the middle have to be clicked open separately but they affect each other. The number field does not produce a clickable affordance since similar looking fields in other parts of the software don't open dialogues.	
Suggestions: Combine the value settings into a common dialog where the affect is seen instantly or make the Trend Window fields look like regular edit boxes that can be modified directly.	

DD13. Unknown alarm lost	**Severity:** 2 a
Observed at: paper mills	**Violated heuristics:** 1, 2, 5, 6, 7, 9
Description: An alarm status is seen in the Control Panel but it is not found in the Fault Logger. The user commented that "some code has come in between".	
Suggestions: A reason should be found to any alarms and faults displayed in the target list or Control Panel. There should be a link to the Fault Logger from the drive status displays.	

DD14. Finding a parameter	**Severity:** 1 b
Observed at: paper mills	**Violated heuristics:** 5, 6, 7
Description: The Parameters window displays all parameters in one long list. Although groups have headlines, they get lost in the text masses. A user has to remember or look up in other documents the number or location of a particular parameter and then scroll up and down the list to find it.	
Suggestions: Implement a search function. Accept keyboard input and jump to the typed number (like already implemented in adding a parameter to a Monitor Window) or name. Make shortcuts, like a drop-down menu, to parameter groups.	

DD15. Too many decimals	**Severity:** 1 b
Observed at: paper mills	**Violated heuristics:** 5, 6, 7
Description: The Monitor Windows show signal values with too many decimals. They are hard to make out fast. Users rather look at graphs and use the cursor function to find out single signal values.	
Suggestions: Make number of displayed decimals configurable. Align signal values by decimal point.	

DD16. Monitor Window contents management	**Severity:** 1 a
Observed at: paper mills	**Violated heuristics:** 1, 2, 3, 4, 5, 6, 7, 8
Description: A new parameter is inserted into a Monitor Window below a highlighted line. Duplicate parameters can be inserted which can lead to cluttered and confusing window contents.	
Suggestions: Don't allow adding duplicates or remove the previously entered and warn the user. Enable changing the order of the parameters e.g. by drag & drop with the mouse. Use a	

double list for adding new parameters (see [27]). Align displayed values by the decimal point or a selectable method.

6.5.4 DriveAP

<table>
<tr><td><b>DAP1. Block appearance not function-specific</b></td><td><b>Severity:</b></td><td>3 c</td></tr>
<tr><td><b>Observed at:</b> training</td><td><b>Violated heuristics:</b></td><td>1, 2, 4, 5, 8, 10</td></tr>
<tr><td colspan="3"><b>Description:</b> All programming blocks look the same: three inputs and one output, even when the function doesn’t need them all. Users get confused and put e.g. unnecessary constant inputs to ADD blocks even though they could be left empty without compromising the logical operation.</td></tr>
<tr><td colspan="3"><b>Suggestions:</b> Redesign the block manipulation dialog and indicate to the user when an input is optional. If the function type of a block takes only two inputs then don’t draw three.</td></tr>
</table>

<table>
<tr><td><b>DAP2. Connecting outputs difficult</b></td><td><b>Severity:</b></td><td>3 c</td></tr>
<tr><td><b>Observed at:</b> training</td><td><b>Violated heuristics:</b></td><td>3, 5, 6, 7, 8</td></tr>
<tr><td colspan="3"><b>Description:</b> Connecting a block output to other drive parameters (displayed on the right side of the screen) has to be done at the destination. Without a connection to a drive, the block output parameter number has to be entered by keyboard.</td></tr>
<tr><td colspan="3"><b>Suggestions:</b> Support a natural order (like program execution) of doing things. Allow making connections by drag & drop with the mouse. List the available block output parameters so they can be selected.</td></tr>
</table>

<table>
<tr><td><b>DAP3. Disconnecting blocks difficult</b></td><td><b>Severity:</b></td><td>3 b</td></tr>
<tr><td><b>Observed at:</b> training</td><td><b>Violated heuristics:</b></td><td>2, 3, 4, 5, 6, 7, 9, 10</td></tr>
<tr><td colspan="3"><b>Description:</b> Severing a connection has to be done at the destination by opening the Connect Pin dialogue by CTRL + clicking the small red ball and selecting the Not Connected radio button. A user tried to empty the Parameter field but got an error message.</td></tr>
<tr><td colspan="3"><b>Suggestions:</b> Implement an undo function. Make connection lines selectable and deletable. Redesign the Connect Pin dialogue.</td></tr>
</table>

<table>
<tr><td><b>DAP4. Beginning a new program confusing</b></td><td><b>Severity:</b></td><td>2 c</td></tr>
<tr><td><b>Observed at:</b> training</td><td><b>Violated heuristics:</b></td><td>4, 5, 6</td></tr>
<tr><td colspan="3"><b>Description:</b> There is no New function in the File menu. Constructing a new program is started by opening an empty template file in Off-Line mode or when not connected to a drive. The template file can be overwritten and thus lost. There is no direct command or access to the appropriate parameter to empty out the blocks when in On-Line mode.</td></tr>
<tr><td colspan="3"><b>Suggestions:</b> Provide a way to empty the program in all modes. People often want to start over with a clean slate. Follow platform conventions and employ the template usage model from Office or similar applications.</td></tr>
</table>

<table>
<tr><td><b>DAP5. Scrolling model antiquated</b></td><td><b>Severity:</b></td><td>2 c</td></tr>
<tr><td><b>Observed at:</b></td><td><b>Violated heuristics:</b></td><td>1, 3, 4, 7</td></tr>
</table>

Description: Vertical and horizontal scroll bars don’t scale to window contents. Window contents aren’t redrawn until the mouse button is released. When the window is resized, the contents jump to the upper left corner. The contents aren’t scalable with resizing.
Suggestions: Follow contemporary platform conventions.

DAP6. Info tab in Multiblock mode	**Severity:** 2 c
Observed at:	**Violated heuristics:** 1, 2, 3, 4, 5, 6, 8, 10
Description: The tab is placed among the time level tabs and its contents cannot be visible at the same time with the constructed program.	
Suggestions: The information content in the Info tab should be in the online help. It could be implemented like the sidebar Task Pane of Office 2003 so it could be viewed simultaneously with the program. The standard mode blocks should also be described like this.	

DAP7. Odd mouse interaction methods	**Severity:** 2 b
Observed at: training	**Violated heuristics:** 4, 7
Description: Blocks and other objects are selected for manipulation by pressing the CTRL key and clicking with the mouse. Users are not accustomed to this and it takes time to learn.	
Suggestions: Follow platform conventions: selection with a single click, default action with a double click and context-sensitive pop-up menu with the secondary button. See [33] for more.	

DAP8. Connect Pin dialogue coarse	**Severity:** 2 b
Observed at: training	**Violated heuristics:** 2, 4, 5, 6, 7, 9, 10
Description: The Parameter and Constant edit boxes offer no assistance to users on what format their input should be. Limits are not told in error messages. A user tried to enter the parameter number of a user constant visible at the left screen edge to the Constant field.	
Suggestions: Inform users on correct formats. Display default values when appropriate. Redesign the whole dialogue.	

DAP9. Block functionality unknown	**Severity:** 2 b
Observed at: training	**Violated heuristics:** 1, 2, 4, 5, 6, 7, 10
Description: The functionality of some blocks, like logical operations, is not clear to all users. One commented: “Can’t remember. These were taught 20 years ago.”	
Suggestions: All blocks should be described in the online help with truth tables and other aides. This information could be available in/from the block modification dialogue as well.	

DAP10. Switching to On-Line loses program	**Severity:** 2 b
Observed at: training	**Violated heuristics:** 1, 2, 3, 4, 5, 6, 8, 9, 10
Description: When switching to On-Line mode (or opening a file) a dialogue asks: “Do you want to save the program on the screen? Yes/No” Some users didn’t understand how a program could be saved “on the screen” and lost their modifications by clicking No or canceling the file saving dialogue after clicking Yes. Transferring the program to the drive by	

the Download function didn’t occur to them.
Suggestions: Rephrase the message and add a Cancel button. Follow platform conventions.

DAP11. Objects modifiable or not	**Severity:** 2 b
Observed at: training	**Violated heuristics:** 2, 4, 5, 6, 7, 8, 10
Description: CTRL + clicking the Time Level field opens a dialogue but the identical-looking Software Version field below it doesn’t. Users, especially novices, get frustrated when they don’t know why some actions have an effect and some don’t.	
Suggestions: Make elements capable of interaction look different from static ones. Follow platform conventions.	

DAP12. Small texts hard to read	**Severity:** 2 b
Observed at: training, paper mills	**Violated heuristics:** 4, 5, 8
Description: Users cannot read the smallest texts like the parameter numbers inside the blocks. Monitors in the training classes were 17” flat screens with a 1280 x 1024 resolution. Service and commissioning people use laptops with smaller 1400 x 1050 screens. Interpolating smaller resolutions in flat screens produce blurring which renders the smallest details undistinguishable. Also printed on A4 size papers (more convenient to take along to assignments) the text is too small.	
Suggestions: Use larger texts. Allow scaling and zooming the graphics.	

DAP13. Bad layout in block management dialog	**Severity:** 2 a
Observed at: training	**Violated heuristics:** 2, 4, 5, 7, 8
Description: The Insert/Change/Remove Block dialogue looks confusing. Several users instinctively moved the mouse cursor to the bottom of the dialogue after selecting a block type, but the Change Block button is located on top of the list.	
Suggestions: Redesign the block modification process or at least this dialogue. Follow platform conventions and natural flow from left to right and top to bottom.	

DAP14. Bad color contrast in empty blocks	**Severity:** 1 a
Observed at: training	**Violated heuristics:** 1, 8
Description: The empty blocks marked with yellow boxes were almost indistinguishable when viewed with a video projector in a bright classroom.	
Suggestions: Use light gray instead. Make colors configurable.	

DAP15. Too many tabs close together	**Severity:** 1 a
Observed at:	**Violated heuristics:** 2, 5, 6, 8
Description: In Multiblock mode there are 15 time level tabs (five per time level) and the Info tab at the bottom of the window. User has to read sequentially through the tabs to find the one he is looking for.	
Suggestions: Leave a small gap between different time levels and delete the Info tab by	

moving its contents to online help. The tabs could also show with a small indicator which ones have blocks inside them.

DAP16. Clumsy Drive menu	**Severity:** 1 a
Observed at:	**Violated heuristics:** 2, 3, 4, 5, 6, 8
Description: The drive menu has a Parameter List submenu with only one item: Upload Values. Disconnect is not selectable when in On-Line mode; user has to click Off-Line first.	
Suggestions: Redesign menu structure. Follow platform conventions.	

6.5.5 Common issues to all programs

C1. Mutual differences	**Severity:** 3 c
Observed at:	**Violated heuristics:** 1, 2, 4, 5, 6, 7, 8, 10
Description: All commissioning software products have many same basic functionalities like parameter comparing and signal graphs. Yet, they are all implemented differently and call same things by varying names. The programs are divided by technology (the drive models they can connect to) and not so much by their users, who are more than likely to come across different drive models in their career, but they have to learn each program separately. Knowledge on operating one program cannot be transferred directly to another.	
Suggestions: Coordinate development between projects. Share user interface design decisions or have one party responsible for designing everything. Build a completely new drive tool platform, with unified user interaction, capable of connecting to any drive, and whose functionality sets can be configured for deployment to divergent user, operation, and application environments.	

C2. Twisted terms	**Severity:** 2 b
Observed at: training	**Violated heuristics:** 1, 2, 4, 5, 6, 7, 8, 10
Description: The terms download and upload are drive-oriented. Users are used to "downloading" files from the Internet *to* their own computer and get confused when the direction changes with the drives. One user commented that in an earlier drive model the terms were actually reversed.	
Suggestions: Replace upload and download with neutral expressions like "transfer to/from drive".	

C3. User has to wait	**Severity:** 2 a
Observed at:	**Violated heuristics:** 1, 2, 4, 5, 6, 7, 8
Description: When transferring parameter names from a drive the user has to wait. In DriveDebug they are read all in one go (long wait), in DriveWindow by groups (several shorter waits).	
Suggestions: Change the progress indicator in DriveDebug from the uninformative animation to one that shows how long the operation takes proportionally or in seconds. Take advantage of multitasking and start reading the parameter names in the background as soon as a connection to a drive is established. If the user wants to do something else that is communication intensive, give precedence to his actions.	

6.6 New feature concepts

Here are some new feature concepts and requirements. Need for them came up during the interviews and user observations.

Vibration/flutter analysis

Vibrations and fluttering in measured signals can reveal e.g. an axel that doesn't fulfill its requirements. To measure vibrations sample frequency has to be short. Over 100 milliseconds in DriveWindow is too much. Commissioning software should have a function for showing frequencies in measured data, especially those under 10 Hz. Fast Fourier Transform could be utilized. Nowadays the numerical data has to be sent in Excel format to someone who has a program capable of doing this kind of calculation.

Communication history

The software should be able to show history of communication in the field buses and other channels. If tied to timeline displays and fault logs, this could help catch erroneous commands and other anomalies.

Bit watcher

The software should have a feature that could show the changing of individual bits in signals. A graphical step display could be integrated (dockable at the user's will) to a trend window and linked to the timeline. It could also be viewable in a list format with timestamps where a bit changes. All bits of a signal could be viewed or individual bits selected from various signals.

More drawable signals

Six drawable signals at the same time is too few. When e.g. measuring torque from several drives in emergency stop testing it would be good to see them all at once and compare differences. Even if it means lower sample frequencies.

Better facilities for analysis after the fact

Ability to leave the computer to save values all night (while sleeping in a hotel) or longer and analyze the results the next day would be beneficial. Finding significant moments in a large pool of data could be supported by automatic markers in the timeline triggered by specified events and limits. Markers should also be able to be placed interactively when something significant is happening in front of the user and he wants to return to the same moment afterwards. Quick jumping from marker to marker should be facilitated.

The software should indicate clearly to the user how long it can be left to record values. The user should not be forced to calculate days from seconds or vice versa. The software should also be able to initiate information gathering by starting data loggers, reading fault logger contents and recording values on its own in alarm and fault situations by user-specified triggering events and value limits.

7 Discussion and conclusion

In retrospect, this research can be called successful. User centered methods and techniques were found that are aimed at supporting software production and that could be utilized in further development activities. Section 4 presented several tools and methods found in literature that can be utilized when designing new software tools. Some of them were tried out in real situations to see their applicability in the ABB environment.

The chosen methods produced results: real end-users were engaged, and problems and good features of the software tools were found. Also requirements emanating from the operating environment were uncovered. The methods were found to be useful and applicable with minor adaptation. Minor disappointments came from the low number of outside customer responses to the questionnaire. Also the ravines between different ABB units produced some frustration.

A paragraph of intent to integrate users to the development process according to user centered design principles was added to the ABB internal handbook [1]. Permanent solutions and decisions were not made yet. There is still a long road to travel before solid user centeredness is achieved. This research gives one view into Finnish frequency converter commissioning. Hopefully it is used in future development activities and found useful, but there are still many issues left to discover, especially international ones.

7.1 Developing the development process

Nielsen [36, page 21] outlines a five-step action plan for a systematic approach to usability from a management perspective:

1. Recognize the need for usability in your organization.
2. Make it clear that usability has management support (this includes promoting a culture where it is seen as positive for developers to change their initial design ideas to accommodate demonstrated user needs).
3. Devote specific resources to usability engineering (you can start out small, but you need a minimal amount of *dedicated* resources for usability to make sure that it does not fall victim to deadline pressures).
4. Integrate systematic usability engineering activities into the various stages of your development lifecycle, including the early ones.
5. Make sure that all user interfaces are subjected to user testing.

The researcher agrees wholeheartedly with these steps. Furthermore, the software development process should be formally defined and then adhered to. Compensating a lack of user knowledge by blind obedience to some standards and conventions is not ultimately a winning strategy [16]. Reactionary satisficing doesn't function as a driver for innovations and market leadership. Neither does the lingering state of software tools being a customary add-on to the actual drive products. Engagement from management is needed to produce a clear vision for the software tools as viable and established marketable products, or some other solution, which is then propagated to every part of the organization.

Forming official development partnerships [48, page 137] with some customer organizations might be a strategy worth exploring. When cooperating with same users for a long time, they will grow better able to anticipate the kind of information system designers are seeking. [13, page 279]. Those more design-wise inclined will notice useful aspects of their own work environment and volunteer this to research. But care must be taken not to become complacent with this kind of service. When data is needed from novice users, then true novices must be gathered. Even if it means additional costs and more organizing.

It should also be noted that some competitors already have a long head start in recognizing the importance of user centered perspective [24] instead of looking at problems through developer eyes. Good user interfaces have to be designed, since they don't just happen. Developing the organization's competency in this regard should be taken seriously. To test and design successfully means that either the current staff has to be educated, new people hired, or some aspects of the process outsourced.

A company the size of ABB might even benefit from having a dedicated usability and user experience unit that could consult other development units globally on HCI issues. As the development process inevitably gets fragmented between several organizations, special tools are required for supporting the collaboration between designers, developers, testers, and implementers. An example of such is Bugzilla, an online bug-tracking tool.

After initial steps have been taken on the road to user centeredness, there are further needs for storing and managing the gathered information as more methods and techniques are utilized. These needs and a Knowledge Storage solution are explored in [38]. The progress of the organization can be evaluated, for example, with the Usability Maturity Model [12]. Also domestic theories on usability capability exist [23].

In conclusion, usability and user centered methods should be seen as an investment for future as well as servicing current development needs. Communication and collaboration are essential tools in modern product development. Information on users is valuable capital; documenting things that seem trivial now might prove to be vital in the next project or the one after that, and transform into actual cash flow.

References

[1]: ABB, Drives internal handbook, Role of tools team, intranet database in Lotus Notes, referenced 15.3.2005

[2]: ABB, 2004, Marketing communications, company general presentation PowerPoint slides

[3]: ABB, 2004, Marketing communications, general drive information PowerPoint slides

[4]: Saija Alaharju, 2002, Frequency converter: a user-centered design study of usability problems, master's thesis, University of Art and Design Helsinki

[5]: Elisa Alatalo, 2004, Tuotetuen tietovirrat teollisuusyrityksessä, master's thesis, Tampere University of Technology

[6]: Katja Battarbee, 2004, Co-experience: Understanding user experiences in social interaction, dissertation, University of Art and Design Helsinki

[7]: Hugh Beyer & Karen Holtzblatt, 1998, Contextual Design: Defining Customer-Centered Systems, Academic Press, ISBN 1-55860-411-1

[8]: Pär Carlshamre & Martin Rantzer, 2001, Dissemination of Usability: Failure of a Success Story (errorneus title in print: A Narrative Approach to User Requirements for Web Design), ACM Interactions, Volume 8, Issue 1, january + february 2001

[9]: Pär Carlshamre, The Delta Method, http://www.ida.liu.se/labs/aslab/groups/um/projects/delta/, referenced 23.2.2005

[10]: Elisa del Galdo et al, International User Interfaces, 1996, John Wiley & Sons, Inc., ISBN 0-471-14965-9

[11]: The Delta Method home page, WM-Data, http://www.deltamethod.net, referenced 29.4.2005

[12]: J Earthy, 1999, Usability Maturity Model: Processes, INTERACT'99, http://www.idemployee.id.tue.nl/g.w.m.rauterberg/lecturenotes/Usability-Maturity-Model[2].PDF, referenced 15.4.2005

[13]: John Ford & Larry Wood, An Overview of Ethnography and System Design in [57]

[14]: Kim Goodwin, Perfecting Your Personas, http://www.uie.com/events/roadshow/know_your_users/articles/perfecting_personas/, referenced 30.3.2005

[15]: Stephen Gorard, 2004, Revisiting a 90-year-old debate: the advantages of the mean deviation, http://www.leeds.ac.uk/educol/documents/00003759.htm, referenced 12.4.2005

[16]: Jonathan Grudin, 1989, The case against user interface consistency, Communications of the ACM, volume 32, issue 10, October 1989, pages 1164-1173

[17]: JoAnn Hackos & Janice Redish, 1998, User and Task Analysis for Interface Design, John Wiley & Sons, Inc., ISBN 0-471-17831-4

[18]: Ilkka Haikala, Jukka Märijärvi, 1998, Ohjelmistotuotanto, Suomen ATK-kustannus, ISBN 951-762-696-7

[19]: Karen Holtzblatt & Hugh Beyer, Contextual Design: Principles and Practice in [57]

[20]: International Business Center, Geert Hofstede Analysis, http://www.cyborlink.com/besite/hofstede.htm, referenced 23.3.2005

[21]: International Organization for Standardization, 1998, ISO 9241-11, Ergonomic requirements for office work with visual display terminals (VDTs) — Part 11: Guidance on usability

[22]: International Organization for Standardization, 1999, ISO 13407, Human-centered design processes for interactive systems

[23]: Timo Jokela, 2001, Assessment of user-centred design process as a basis for improvement action, dissertation, University of Oulu, ISBN 951-42-6550-5

[24]: Mette Kjaersgaard et al, 2003, A Lost Cause: The Ever-Improving Developer's Map, ACM, CHI 2003: New Horizons, Design and Usability in Practice, pages 642 – 643

[25]: Kirsi Kontio, 2003, Viankuvausjärjestelmän käyttökontekstin kartoitus Contextual Design -menetelmää soveltaen, master's thesis, Helsinki University of Technology

[26]: Aki Kulmala, 2003, Requirements for AC-Drives in Food and Beverage Industry, master's thesis, Helsinki University of Technology

[27]: Sari Laakso, User Interface Design Patterns, http://www.cs.helsinki.fi/u/salaakso/patterns/, referenced 20.3.2005

[28]: Richard D. Lewis, 1999, When Cultures Collide, Nicholas Brealey Publishing, ISBN 1-85788-087-0

[29]: Petri Mannonen, 2004, Photography based artefact analysis in user-centered design, Master's thesis, Helsinki University of Technology

[30]: Deborah J. Mayhew, 1999, The Usability Engineering Lifecycle: a practitioner's handbook for user interface design, Academic Press, ISBN 1-55860-561-4

[31]: Ilkka Mellin, 2004, Johdatus tilastotieteeseen: Tilastollisten aineistojen kuvaaminen, http://www.sal.tkk.fi/Opinnot/Mat-2.090/pdf_varasto/TILAK100.pdf, referenced 12.4.2005

[32]: Microsoft Corporation, Windows User Experience Guidelines, http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnanchor/html/anch_uidesigndev.asp, referenced 10.3.2005

[33]: Microsoft Corporation, Windows User Experience Guidelines / Mouse Interface Summary, http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnwue/html/appxa.asp, referenced 10.3.2005

[34]: J.S. Milton & Jesse C. Arnold, 1995, Introduction to probability and statistics, McGraw-Hill Inc., ISBN 0-07-113535-9

[35]: Aki Helen Namioka & Christopher Rao, Introduction to Participatory Design in [57]

[36]: Jacob Nielsen, 1993, Usability Engineering, Academy Press, London, UK. ISBN 0-12-518406-9

[37]: Jacob Nielsen, Ten Usability Heuristics, http://www.useit.com/papers/heuristic/heuristic_list.html, referenced 7.3.2005

[38]: Marko Nieminen, 2004, Information Support for User-Oriented Development Organisation, dissertation, Helsinki University of Technology

[39]: Jouko Niiranen, 1999, Sähkömoottorikäytön digitaalinen ohjaus, Otatieto, ISBN 951-672-270-9

[40]: Donald A. Norman, 1988, The Psychology of Everyday Things, Basic Books, ISBN 0465067093

[41]: Object Management Group Inc., 2003, OMG Unified Modeling Language Specification, version 1.5

[42]: Jarmo Parkkinen, Käytettävyys, mitä se on?, http://www.adage.fi/artikkelit/kaytettavyys_mita_se_on.html, referenced 26.1.2004

[43]: Jenny Preece et al, 1994, Human-Computer Interaction, Addison Wesley, ISBN 0-201-62769-8

[44]: Martin Rantzer, The Delta Method - A Way to Introduce Usability in [57]

[45]: Jef Raskin, Intuitive Equals Familiar, http://www.asktog.com/papers/raskinintuit.html, referenced 17.1.2005, also Communications of the ACM, volume 37, issue 9, September 1994, page 17.

[46]: Matthias Rauterberg et al, Benefits of user-oriented software development based on an iterative cyclic process model for simultaneous engineering, Elsevier, International Journal of Industrial Ergonomics 16 (1995), pages 391-410 (IJoIE_article_(1995)_vol_16.pdf)

[47]: Sirpa Riihiaho, 2000, Experiences with usability evaluation methods, licentiate's thesis, Helsinki University of Technology

[48]: David Rowley, Organizational Considerations in Field-Oriented Product Development: Experiences of a Cross-Functional Team in [57]

[49]: Patricia Russo & Stephen Boor, 1993, How Fluent is Your Interface? Designing for International Users, Proceedings of the InterCHI '93, pages 342-347

[50]: Ben Schneiderman, 1998, Desinging the User Interface, Addison Wesley, ISBN 0-201-69497-2

[51]: Ben Schneiderman, 1983, Direct Manipulation: A Step Beyond Programming Languages, IEEE Computer, 16(8), pages 57-69

[52]: Irmeli Sinkkonen et al, 2002, Käytettävyyden psykologia, Edita Publishing Oy / IT Press, ISBN 951-826-574-7

[53]: Sirsi Corporation, Library Automation/Technology Glossary, http://www.libraryhq.com/glossary.html, referenced 8.3.2005

[54]: Jared Spool, What Makes a Design Seem 'Intuitive'?, http://uie.com/articles/design_intuitive/, referenced 11.1.2005

[55]: University of Massachusetts Amherst, Pugh method or decision-matrix method, http://mielsvr2.ecs.umass.edu/virtual_econ/module2/pugh_method.htm, referenced 21.3.2005

[56]: Usability Professionals' Association, Resources: Usability in the Real World, http://www.upassoc.org/usability_resources/usability_in_the_real_world/, referenced 26.1.2004

[57]: Dennis Wixon & Judith Ramey, 1996, Field Methods Casebook for Software Design, John Wiley & Sons Inc., ISBN 0-471-14967-5

Appendix A: Usability goals, requirements, and meters

The Delta Method Work Resource 5 [11] lists the following usability goals that can be utilized in defining usability requirements:

- **Efficient.** The system should be fast, the services relevant and it should not be possible to make serious errors. (related to usability attribute efficiency)
- **Fast.** The user should be able to carry out his or her tasks swiftly. (efficiency, relevance)
- **"Walk-up-and-use".** "Anyone" should be able to use the system without previous experience and training, and probably also without any users' information. A user may perhaps use this system only once (compare with information kiosks). (relevance, learnability)
- **Fit for continued use.** Even if the users have a choice of continuing using the system or not, they should wish to do so after their initial contact. (learnability, efficiency)
- **Fit for intermittent use.** A person should be able to efficiently use the system even if there are long periods of time between every occasion. Focus should be on re-learnability, consistency, predictability and task-oriented users' information. (learnability, efficiency)
- **Fit for experienced use.** The users can be expected to become experts of the system, and to use it frequently. Focus should be on support for customization, learning over time and a feeling of development (learning new features). (relevance, efficiency, learnability)
- **Maintaining personal integrity.** For systems that are integrated in a social structure, it can be important that personal integrity is maintained. (attitude, relevance)
- **Allowing control.** The users should feel in control of the pace and contents of their work within the system. Undo functions, visible consequences and visible structure of services and information are important. (attitude, relevance)
- **Interesting.** The system should tickle its users' interest, and make them wish to explore the system and to perform new tasks with it (compare with a good web site). (attitude)
- **Challenging.** The system can require a lot from the users, but their rewards are rich and immediate (compare with computer games) (attitude)

Other usability goals can be formulated from general usability, minimized need for support, learnability (learning rate), or tolerance for errors.

Sinkkonen [52, page 304] lays out the following purposes and goals for usability testing

- find problems for inexperienced/experienced/both users
- measure whether there are functions that novices can't figure out in reasonable time without manual

- what experienced users do when they get certain error messages
- is a new version faster than old

Meters for testing (measure during test or after)

- The time it took to do a task or all of them.
- How many tasks were completed in a certain time?
- How many erroneous task executions there were?
- How long it took to recover from errors?
- The number of usability faults the user suffered from.
- How often were help and manuals needed?
- How many times did the user express negative attitudes or frustration?
- How many times was the user able to do a task straight on?
- How many times did the user hesitate?
- How many times did the user get completely lost?
- How many times did the user need a hint from the instructor or substantial help?
- How many tasks did not get done right?
- How many tasks did the user not notice?

Quantifiable usability measurements by Nielsen [36, page 194]:

- The time users take to complete a specific task.
- The number of tasks (or the proportion of a larger task) of various kinds that can be completed within a given time limit.
- The ratio between successful interactions and errors.
- The time spent recovering from errors.
- The number of immediately subsequent erroneous action.
- The number of commands or other features that were utilized by the user (either the absolute number of commands issued or the number of *different* commands and features used).
- The number of commands or other features that were never used by the user.
- The number of system features the user can remember during a debriefing after the test.
- The frequency of use of the manuals and/or the help system, and the time spent using these system elements.
- How frequently the manual and/or help system solved the user’s problem.
- The proportion of user statements during the test that were positive versus critical towards the system.
- The number of times the user expressed clear frustration (or clear joy).

- The proportion of users who say that they would prefer using the system over some specified competitor.
- The number of times the user had to work around an unsolvable problem.
- The proportion of users using efficient working strategies compared to the users who use inefficient strategies (in case there are multiple ways of performing the tasks).
- The amount of “dead” time when the user is not interacting with the system. The system can be instrumented to distinguish between two kinds of dead time: response-time delays where the user is waiting for the system, and thinking-time delays where the system is waiting for the user. These two kinds of dead time should obviously be approached in different ways.
- The number of times the user is sidetracked from focusing on the real task.

ISO standard 9241-11 [21], especially Annex B, is a good source for further usability measures.

Appendix B: Drive type matrix

Type	power (kW)	application	DriveWindow Light "for ABB standard drives"	DriveWindow "for ABB industrial drives"	DriveDebug "for products that use DDCS"	DriveAP "for industrial drives"
standard drives:						
ACS550	0.75 - 355	wide range in industries	x	-	-	-
ACH550	0.75 - 355	HVAC industry (heating, ventilation, air conditioning)	x	-	-	-
industrial drives:						
ACS800 (stand-alone single)	0,55 - 2800	wide range, process industries	x (via panel interface)	x	x	x
ACS800 (multidrive)	1,5 - 5600	wide range, process industries (system integrators, OEMs)	x (via panel interface)	x	x	x
ACS800 modules (single/multi)	45 - 500 / 1,5 - 2000	(system integrators)	x (via panel interface)	x	x	x
ACS600 MultiDrive, MarineDrive, etc.	1,5 - 4300		x (via panel interface)	x	x	-
General machinery drives:						
ACS140	0,12 - 2,2	material handling, packaging	x	-	-	-
decentralized drives:						
ACS160 (integral drive)	0,55 - 2,2	dairies, food processing, automotive, conveyor	x	-	-	-
component drives:						
ACS50	0,18 - 2,2	fans, pumps, gate control, material handling, conveyors (OEMs, installation companies, panel builders)	-	-	-	-
ACS100	0,12 - 2,2		-	-	-	-

Appendix C: Cultural associations of color

Colors have different associations in different countries [49]. In Egypt, a red X may not be associated with forbiddance and prohibition. In designing user interfaces, the colors may have to be translated as well, when localization is planned. The following table shows some colors and their meanings in a few countries.

Culture	Red	Blue	Green	Yellow	White
United States	danger	masculinity	Safety	cowardice	purity
France	aristocracy	freedom, peace	Criminality	temporary	neutrality
Egypt	death	virtue, faith, truth	fertility, strength	happiness, prosperity	joy
India	life, creativity		prosperity, fertility	Success	death, purity
Japan	anger, danger	villainy	future, youth, energy	grace, nobility	death
China	happiness	heavens, clouds	Ming dynasty, heavens, clouds	birth, wealth, power	death, purity

Appendix D: Style Guide skeleton

Mayhew [30, page 321] suggests the following as a template for a Style Guide in the Usability Engineering Lifecycle process.

Preface (authors, time frame, evolving nature of the document)

Part one: Basis for the XXX Application User Interface Design

1. Introduction (considerations that went into design)
    - 1.1. Purpose (benefits of good user interface design)
    - 1.2. Scope (future related applications, subject to parts of this guide)
    - 1.3. Audience (users and maintainers of the document)
    - 1.4. Application of Standards (definition of usage in the development process)
    - 1.5. How to Use This Document (by chapter, when, references)
2. Overview of Functionality (of application XXX)
3. User Profiles
    - 3.1. Summary of Methodology
    - 3.2. User Profile Summaries with Implications
4. Contextual Task Analysis
    - 4.1. Summary of Methodology
    - 4.2. Work Environment Analysis with Implications
    - 4.3. Task Analysis Document with Implications
    - 4.4. Task Scenarios with Implications
    - 4.5. Current User Task Organization Model with Implications
5. Platform Capabilities and Constraints
    - 5.1. Identification of Hardware Platforms
    - 5.2. Identification of Software Platforms and Tools
    - 5.3. Summary of Platform Capabilities and Constraints with Implications
6. Usability Goals
    - 6.1. Qualitative Goals
    - 6.2. Quantitative Goals

Part Two: Designing the XXX Application User Interface

7. Reengineered Work Models
    - 7.1. Reengineered Task Organization Model
    - 7.2. Reengineered Task Sequence Models
8. Conceptual Model Design
    - 8.1. Identification of Products/Processes

8.2. Presentation Rules for Products/Processes

8.3. Rules for Windows

8.4. Major Displays

8.5. Major Navigational Pathways (e.g., Menu Bar)

9. Input and Output Devices (refer to higher level style guides where appropriate)

9.1. Screen (available attributes from Platform Capabilities/Constraints)

9.2. Mouse/Pointing Devices (click, drag, and other actions in different contexts)

9.3. Keyboard

9.4. Sound

10. Screen Design Standards

10.1. Application Windows

10.2. Product/Process Windows

10.3. Dialog Box Layout and Design (outline of rules)

10.3.1. Title

10.3.2. Dialog Box Body

10.4. Message Dialog Box Design

10.4.1. Error Message Box

10.4.2. Warning Message Box

10.4.3. Information Message Box

10.5. Help Windows

10.5.1. Appearance of a Help Window

10.5.2. Invoking the Help System

10.5.3. Help Window Writing Style

11. Feedback

11.1. Response Time (general description and feedback standards)

11.2. Error Handling (general description)

11.2.1. Types of User Errors

11.2.2. Guidelines for Error Prevention

11.2.3. Guidelines for Error Recovery

11.2.4. Error Message Writing Style

11.3. Status Information (guidelines for static and transient status message wording and presentation)

11.4. Visual Cues (general rules organized as usage followed by cue)

Appendix E: Use case template

This template is a compilation from several sources: HUT courses, online tutorials, and personal experience. Other possible fields in the use case could be: Domain, Parameters (from the initiating actor), Trigger, Priority, Performance (criteria), and others as seen necessary.

Name	*Name and a possible identifier of the use case.*
Summary *(optional)*	*A short description for an exceptionally long use case.*
Performers	*The actors performing this use case.*
Preconditions	*Must be met before this use case can be invoked.*
Description	*The basic sequence or course of action in this use case and possible exceptions:* 1. First step 2. Second step [Exception A] 3. Third step, etc.
Exceptions	*Description of the exceptions to the basic sequence:* Exception A: *reason for deviation and a walkthrough for the different course of action*
Postconditions	*Will be true after a successful finish.*
Other requirements *(optional)*	*Issues and constraints affecting this use case. Can be of technical or human nature.*
Frequency *(optional)*	*How often this use case is invoked by the actors.*
Extended use case *(optional)*	*Which use case this one extends.*
Included use cases *(optional)*	*Which use cases are parts of this one.*
Assumptions *(optional)*	*About the domain. When verified, can evolve into decisions or parts of the course of action.*
Alternate courses of action *(optional)*	*A correct sequence of action that can be substituted with the basic sequence.*
Change history *(optional)*	*When, why, by whom.*
Decisions *(optional)*	*A list of critical decisions pertaining to the content of this use case*
Status *(optional)*	*For project management:* Work in progress / Ready for review / Passed review / Failed review

Appendix F: Questionnaire cover page

User questionnaire of DriveWare® software for master's thesis, 2004-2005
Helsinki University of Technology, Antti Kangas, antti.kangas@fi.abb.com

ABB Oy Drives / Antti Kangas
P.O.Box 184
00381 Helsinki
Finland

These questions are made for research purposes. Individual answers will not be shown to outsiders and identities of respondents will not be revealed. If you cannot answer a question, you can leave it blank. Your answers are appreciated highly and they help to develop the DriveWare® software family to better fulfill user expectations.

First part (A) of these questions inquires your general information and your work environment. Sections B, C, D, E, and F contain specific questions about DriveWindow, DriveWindow Light, DriveDebug, DriveAP, and DriveSize respectively. If you don't know or don't use some of these applications, just mark the first box of the first question in that section and move on to next page. The last section (G) is about your expectations regarding drive related software and your experience with other software titles.

There are many questions that have seven boxes between two statements. Mark the box that you feel most closely matches your current situation and opinions.

Example:

A18. Computer skills: **beginner** □ □ □ □ ☒ □ □ **expert**

You might answer like this, if you feel that you have slightly better than average computer skills.

Date of answering: ____

Part A, personal information:

Name: ____

Employer: ____

Address: ____

____

____

Country: ____

Gender: □ **male**

□ **female**

□ **I am interested in developing ABB's PC software tools in the future by answering questionnaires or participating in a usability test. This is a preliminary inquiry, <u>not</u> a binding commitment. For further details, I can be reached by:**

Phone number: ____

Email: ____

Appendix G: Expanded results of select user questionnaire questions

B7. DriveWindow 2.x supports the following work tasks
(very badly = 1, very well = 7)

	N	Average	Standard deviation	Average deviation
Backup of drive information	20	4,95	1,05	0,68
Setting and changing parameters	20	3,90	1,77	1,35
Updating drive software	17	3,59	1,70	1,43
Controlling a drive	20	3,30	1,81	1,53
Commissioning a drive	20	3,15	2,01	1,75
Problem solving	20	2,95	1,57	1,17
Numerical monitoring	20	2,90	1,94	1,80
Documenting the production system	13	2,31	1,75	1,41
Graphical monitoring	20	2,30	1,87	1,66
Managing several drives	17	2,18	1,67	1,38

B8. Importance of DriveWindow 2.x functionality in my work tasks
(unimportant = 1, vital = 7)

	N	Average	Standard deviation	Average deviation
Setting and changing parameters	21	5,57	0,68	0,57
Graphical monitoring of signals	21	5,48	0,81	0,65
Comparing parameters	21	5,33	0,73	0,63
Saving parameters to computer	21	5,29	1,06	0,82
Fault logger	21	5,24	0,89	0,73
Saving the drive's software to computer	20	5,15	1,42	0,94
Saving and loading of signal trends	21	5,10	1,18	0,95
Numerical monitoring of signals	21	4,90	1,09	0,89
Step tests	20	4,90	1,37	1,04
Data logger	21	4,86	1,01	0,84
Exporting trends to other applications	20	4,70	1,26	1,06
Saving and loading of the workspace	19	4,63	1,77	1,37
Working with several drives	21	4,52	1,54	1,21
Show drive status	21	4,38	1,66	1,22
OPC Server remote operation	17	2,53	1,62	1,27

B9. Comments on working with DriveWindow 2.x and/or comparison with other similar software (like DriveWindow 1.x)

- Printerfunktionen saknas. Borde finnas valmöjligheter att printa kompl. parameterlista eller de som avviker från macrot.
- Tosi hyvä ohjelma. Esim. Siemenssin Drive Monitor trendien piirto hidasta (sarjaliikenne). Tosin Drive Monitor parempi parametrien tallennus. "Tallenna vain muutetut parametrit" tämä on hyvä. Siemenssin ohjelma ilmainen tulee käytön mukana.
- ihan jees
- ihan ok käyttää, mutta tulee käytettyä DriveDebugia enemmän (monessa asiassa Dwin on parempi kuin DB)
- DW1 parempi ja selkeämpi kuin DW2. Molemmissa käytöstä toiseen siirtyminen hankalaa (hidasta)

- Käytän Drive Debuggeria enemmän
- Buggeri on paljon parempi k.otto työkalu
- Vertaus Drive Buggeriin. Macrotoiminta on ehdoton. Buggerissa aina startattaessa edellisen istunnon ikkunat ruudulla. Drive Buggerissa paljon hyviä ominaisuuksia. Siitä puuttuu Backup toiminnot. DW käyttäminen hankalaa, ei vaikeaa. ”Parametrien asetus ja muuttaminen” etenkin 100:n ryhmään tukee erittäin huonosti
- Ohjelmasta pitäisi pystyä tulostamaan parametrit paperille. Jos joku valikkotoiminto ei ole mahdollista niin se on harmaana. Olisi hyvä jos näkyville tulisi ikkuna jossa kerrottaisiin miksi toiminto ei ole sallittu.
- Ensimmäinen DW2 oli suorastaan surkea ohjelma, mutta tällä hetkellä se paranee jokaisen päivityksen yhteydessä. Monitorointi sekä sen skaalaus on vielä erittäin huono.
- Overall window size was better in 1.4. Comparing should be possible in local mode. Numerical upload should be possible in 2.1. DriveWindow & DDC tool should have same look & feel. Time stamping & date stamping should be possible on all backups & also on data / fault loggers

	2.x	1.4
Numerical upload of data logger	ok	very good
zooming graph	same	same
selecting graph signals	ok	very good
look & feel	ok	very good
par compare	ok	very good
many drives	v.good	ok
cursor pointing	ok	v.good

- Kaikki toiminnot englannin kielellä, mistään ei tule niin huono olo kuin käyttää suomenkielisiä ohjelmia englannin kielisessä käyttöjärjestelmässä
- DW soveltuu mielestäni huonosti "vaativaan" käyttönottoon. Verrattuna Drive Debuggiin käyttäminen on hidasta. DW:n miinukset verrattuna: (käyttönottoa ajatellen, aktiivinen tarve käyttöönotossa ACS6000 noin 4-6 viikkoa)
  - monitorointi-ikkunoiden teko ehdottomasti liian hidasta
  - mon. ikkunan skaalaus työlästä ja hidasta
  - valmiiksi tehtyjen mon. ikkunoiden avaaminen hidasta
  - act signaalien monitorointi (valmiiksi valitut parametrit) hankalampaa kuin DD
  - Toisaalta etuna DW:ssä DD:n verrattuna parametrien nopea "latautuminen" (DD:ssä parametrir täytyy ladata aina uudelleen kun valitsee eri käytön (multidrive), kestää noin 1-2 min)
  - Ja DW:ssä ulkoasu ja käyttöliittymä kieltämättä parempi ja edustavampi.
  - Tämän hetkisistä DW:tä en kutenkaan käytä muuhun kuin dataloggereihin ja asiakaskoulutukseen. -> Parempaa odotellessa.

C7. DriveWindow Light supports the following work tasks
(very badly = 1, very well = 7)

	N	Average	Standard deviation	Average deviation
Commissioning a drive	5	5,20	0,45	0,32
Setting and changing parameters	5	5,20	0,45	0,32
Controlling a drive	5	5,00	0,71	0,40
Numerical monitoring	4	4,75	0,50	0,38
Backup of drive information	5	4,60	0,89	0,64
Graphical monitoring	4	4,50	1,00	0,75
Problem solving	4	4,25	0,96	0,75
Documenting the production system	3	3,33	2,08	1,56
Managing several drives	3	3,33	2,08	1,56

C8. Importance of DriveWindow Light functionality in my work tasks
(unimportant = 1, vital = 7)

	N	Average	Standard deviation	Average deviation
Setting and changing parameters	5	5,40	0,55	0,48
Saving parameters to computer	5	5,40	0,55	0,48
Show drive status	5	5,00	1,00	0,80
Graphical monitoring of signals	4	4,75	0,50	0,38
Comparing parameters	5	4,60	1,52	1,04
Searching of parameters	5	4,60	1,52	1,04
Numerical monitoring of signals	4	4,50	0,58	0,50
I/O Mapping Table	4	4,00	1,41	1,00
Saving and loading of signal trends	5	3,80	1,10	0,72
Step tests	4	3,50	1,00	0,75
Start-up wizard	4	3,00	0,82	0,50

C9. Comments on working with DriveWindow Light and/or comparison with other similar software or using just the panel of the drive

- Borde finnas printfunktion för parameter listor.
- komea on joo
- nopeempi käyttää koneelta jos paljon asetteluja parametreihin, mutta monesti paneelilta kerkeää tehdä pienemmät muutokset ennen kun on avannut PC:n ja ohjelman. Vianhaussa kätevämpi.
- Ohjelman huono puoli on, että PC:ssä pitää olla ko. käytön ohjelmaversion luettelo. Jos tulee uusi versio, niin ohjelma pitää päivittää.
- Par backup should be possible in Loc mode also. Par compare is very good. Wizard is excellent. But overall this is v.good compared to old version.

D7. DriveDebug supports the following work tasks
(very badly = 1, very well = 7)

	N	Average	Standard deviation	Average deviation
Graphical monitoring	20	5,60	0,60	0,52
Numerical monitoring	20	5,45	0,69	0,61
Commissioning a drive	19	5,42	0,84	0,67
Setting and changing parameters	19	5,05	1,27	1,00
Controlling a drive	19	4,95	1,31	1,02
Problem solving	20	4,85	1,14	0,90
Backup of drive information	18	4,28	1,32	1,06
Managing several drives	18	3,67	1,64	1,30

Updating drive software	19	2,95	2,20	1,95
Documenting the production system	17	2,59	1,58	1,31

D8. Importance of DriveDebug functionality in my work tasks
(unimportant = 1, vital = 7)

	N	Average	Standard deviation	Average deviation
Graphical monitoring of signals	20	5,75	0,44	0,38
Setting and changing parameters	20	5,35	0,93	0,72
Numerical monitoring of signals	20	5,35	0,88	0,72
Saving and loading of signal trends	20	5,35	0,88	0,72
Data logger	20	5,25	0,79	0,68
Fault logger	20	5,15	1,04	0,85
Saving parameters to computer	20	5,10	1,17	0,90
Saving the drive's software to computer	17	4,82	1,33	0,88
Show drive status	20	4,75	1,59	1,18
Macros	20	4,65	1,35	1,06
Working with several drives	20	4,15	1,90	1,54
AC80 special functionality	20	3,25	1,77	1,40
Remote operation	18	2,56	1,50	1,22
Visual Basic applications	19	2,53	1,50	1,24
Language translation	20	2,20	1,91	1,62

D9. Comments on working with DriveDebug and/or comparison with other similar software

- Hieman sekava
- Ainoa ohjelma, jolla laitetta pystyy oikeasti käyttämään / testaamaan. Laitteen ohjaus ja monitorointi yhtä aikaa ei onnistu (ellei käytä 2 PC:tä). Näytteenottotarkkuus ei aina riitä. Hienoa että asetukset voi räätälöidä laitteen/mittauksen tarpeiden mukaan ja palauttaa tarvittaessa.
- todettu parhaaksi
- Käytän eniten ko. ohjelmaa (ohjelmalla hankala tehdä raportteja asiakkaalle)
- Graafisen ohjelman näytön käyttämä ala hirmu pieni!!! Esim. 19" näytöllä vain 1/3 käytössä!!!
- Helppokäyttöinen ja selkeä verrattuna DW 1.4/2.12. Parametrien (parametri-ikkuna) huonompi kuin DW 1.4;ssä muutoin selkeästi parempi työkaluohjelma kuin DW:t
- Paras k.otto työkalu
- Parempi kuin muut, joitain parannuksia kaipaa
- Periaatteessa erittäin hyvä ohjelma. Soveltuu käytt.ottoihin, joissa 1-100 käyttöä
- paras käyttöönottoon
- DriveDebug on tehokas työkalu, DriveWin 2.12 backupin tekemiseen sopiva, muutoin huono.
- Ulkoasu vähän sekava. Hyvää: kun ohjelman käynnistää niin siinä on samat ikkunat kuin sammutettaessa, voi olla useita käyrän piirtoikkunoita, luotettava

- Erittäin helppokäyttöinen. Sopii hyvin k-ottoon jouhean trendi/numeerinen esityksen vuoksi. Parannettavaa: Numeerinen esitys voisi toimia yhtä aikaa trendi(graafisen)esityksen kanssa. (esim. Numeerinen taustalla, hitaammalla päivityksellä.)
- Parametrien ylöslatautuminen käytöltä on liian hidasta ja etukäteen on mahdotonta tietää kuinka kauan tämä tulee kestämään koska nopeus vaihtelee.

E9. DriveAP supports the following work tasks
(very badly = 1, very well = 7)

	N	Average	Standard deviation	Average deviation
Constructing the AP program	5	4,00	1,00	0,80
Commissioning the AP program	5	4,00	1,00	0,80
Designing AP program logic	5	3,80	1,10	0,96
Monitoring operation of AP program	5	3,60	2,19	1,68
Backup of drive information	4	3,50	2,65	2,00
Documenting the production system	4	3,50	2,65	2,00
Problem solving	5	3,40	1,67	1,28
Setting and changing parameters	5	2,60	0,89	0,64
Managing several drives	4	2,00	1,41	1,00

E10. Importance of DriveAP functionality in my work tasks
(unimportant = 1, vital = 7)

	N	Average	Standard deviation	Average deviation
Numerical monitoring of signals	5	5,20	1,30	0,96
Saving the AP program to computer	5	5,00	1,41	1,20
Programming without a drive attached	5	5,00	1,41	1,20
On-line mode	5	5,00	1,41	1,20
Setting and changing parameters	5	4,80	1,30	1,04
Off-line mode	5	4,60	1,34	1,12
Show drive status	4	4,50	1,29	1,00
Multi Block programming mode	5	4,20	2,68	2,16
Working with several drives	4	3,75	2,22	1,75

E11. Comments on working with DriveAP and/or comparison with others (like IEC 61131-3)

- Ohjelmasta ei näy tilatietoja. Vaikea simuloida (sovellusta). Vältän on-line ohjelmointia. Lohkoja voisi kehitellä ja tehdä uusia. Ohjelmissa virheitä esim. AO lähdöt jos muutat skaalausta muuttuu 4.mA myös.
- It should be like FCB, click&drag type. It should be possible to monitor on line values at the input & o/p of each block. Blocks should not have the same position, it can be part of DW 2.x & not a separate tool.

G1. Features I expect from drives' PC software tools (1 = unimportant – 7 = vital)

	N	Tools team average	Average	Standard deviation	Average deviation
free of errors	20	6,00	5,70	0,81	0,61
clear appearance	22	3,67	5,23	0,76	0,53
speed of accomplishing tasks	21	4,67	5,19	0,87	0,58
use intuitively	22	5,00	5,14	0,98	0,75
familiar logic of operation	22	4,33	5,05	0,98	0,76

supporting familiar habits of working	22	3,67	5,05	0,88	0,67
improves ways of working compared to those used previously	19	4,67	5,00	1,45	0,87
advanced functionality for experienced	22	4,00	4,95	0,81	0,62
large set of functionality	20	3,67	4,85	0,89	0,70
functions arranged by situations of use	22	4,00	4,45	1,34	1,00
beginner-friendly	22	3,67	4,45	1,23	1,09
instructions arranged by situations of use	22	3,67	4,14	1,20	0,98
chance to influence in properties of new versions	19	4,33	4,05	1,34	0,90
keyboard shortcuts	21	3,00	3,67	1,50	1,18
similar appearance with other software	21	3,00	3,57	1,79	1,45
enhanced utilization of different mouse buttons	21	2,67	3,29	1,67	1,27
stylish looks	22	2,67	3,09	1,56	1,16
using with mouse only	22	2,00	2,68	1,87	1,50
modifiable appearance	21	1,33	2,62	1,65	1,28
instructions in my own language	21	1,33	2,24	1,64	1,27
whole software in my language	21	1,33	2,10	1,67	1,33

Appendix H: Interview questions

All interviewees were Finnish so the following pool of questions are also in Finnish and in a condensed form.

General to everyone:

Nimi, ikä, äidinkieli, kansallisuus, kauanko työelämässä, kauanko tällä alalla, kauanko tässä firmassa ja tehtävässä, asema organisaatiossa, mitä teet, mistä olet vastuussa, keiden kanssa työskentelet (lähimmät työtoverit, tiimi), keneltä ja miten saat tehtävissäsi tarvittavat tiedot, kenelle esität työsi tulokset, minkälainen koulutus (hyöty työtehtävässä, milloin viimeksi tarvinnut koulutuksessa saatua tietoa/taitoa, tietotekniikka, riittävyys työtehtäviin, kielitaito, sähkö/koneet), värisokeus tai muut haitat, kokemus tuotteista, kokemus samankaltaisista tuotteista, koulutus tuotteista, jäikö jotain olennaista kysymättä.

About a product:

Tuotteen (tai projektin) nimi, piirteet (yksi/monta käyttäjää, olemassaoleva/uusi, massa/custom), syy kehitystyölle (päivitys, aukon täyttö omassa valikoimassa tai markkinoilla, tekninen innovaatio), kohdemarkkinasegmentti, pääasialliset toiminnot, ketkä ovat stakeholderit, ketkä ovat käyttäjiä (loppukäyttäjät, asennus, tiedon vastaanottajat, palveluntarjoaja, ylläpito, osto, markkinointi, tuki, johto, sidosryhmät), mitkä ovat käyttäjän yleisimmät tehtävät tuotteella, muita tärkeitä, mutta harvinaisempia tehtäviä, oletko itse käynyt paikan päällä, missä tuotetta käytetään? kokemuksia ja havaintoja.

To sales and marketing:

Mitä myytte (onko softa muiden tuotteiden sivussa), kenelle, mihin, fokuksena asiakas vs. käyttäjä, kuka neuvottelee, kenen kanssa ja miten, kuinka vaatimukset saadaan, pidetäänkö niistä kirjaa, kenen näkökulmasta vaatimukset ovat, huomioidaanko käyttäjää, luotetaanko vain ostajahenkilöön, mitä odotuksia asiakkailla on tuotteista, mitataanko asiakastyytyväisyyttä, tuloksia, ketkä ovat käyttäjiä, missä käyttäjät sijaitsevat, käyttäjien painoarvo, alueittain, mitä tekevät tuotteella, mitä ovat käyttäjien kontaktipinnat ABB:hen, onko tiedossa sopivia asiakkaita, joihin ottaa yhteyttä (käyttöönotettava drive juuri myyty tai pian myytävä).

To developers:

Työn muoto (projektityötä?), missä projekteissa nyt kiinni, mistä tiedät, mitä ominaisuuksia ja toimintoja tehdä, kuinka vaatumukset saadaan, keneltä vaatimukset ovat, kuinka vaatimuksia hallitaan ja validoidaan, prioriteetit, vaikutukset muihin, generic/customized –näkökulma.

To support:

Ketkä ottavat yhteyttä, minkälaisia ongelmia käyttäjillä on, missä tilanteissa (näiden ongelmien vaikutus stakeholdereille), ongelmien ja yhteydenottajien toistuvuus, globaali jakauma, mistä tulee yhteydenottoja, yhteydenottotapa, seuranta, mitä tiedät (konteksti) käyttäjästä, joka ottaa yhteyttä ongelmansa vuoksi.

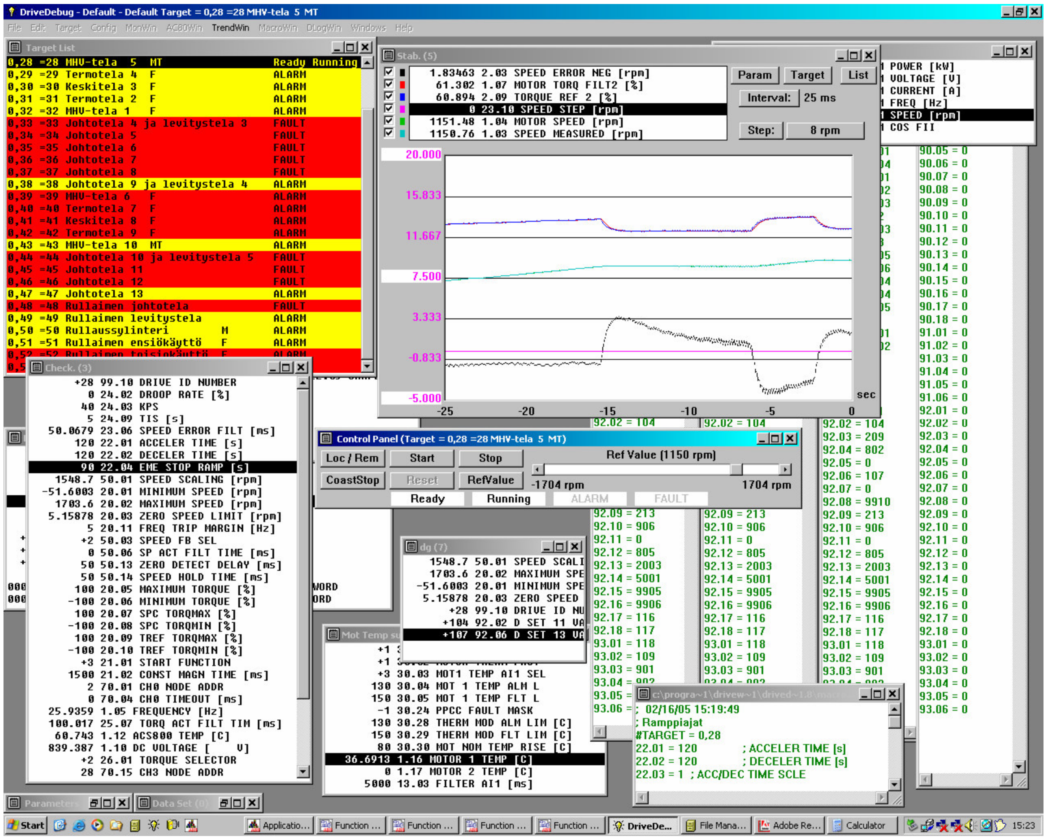
To end-users:

Mitä sovelluksia käytät, missä tilanteissa, kuinka usein, käytätkö yksin, tuntuuko softa helppokäyttöiseltä, tyypillinen käyttötilanne, tehtäväsekvenssi, mitä ongelmia olet kohdannut, tuntuiko laite/softa olevan koko ajan hallinnassasi, asenne tietotekniikkaan, käytätkö mieluummin softaa vai paneelia, missä käytät tuotetta (ympäristö), hallittavien käyttöjen määrä (+tyyppi), millainen olisi ihanteellinen softa X, tärkeimmät toiminnot, ongelmatilanteet (kiire, stressi), parametrien ryhmittely, tyypit (signaali/parametri), tietokone kotona.

Software specifics and micellaneous:

Mitä tapahtuu DriveWindow'n käynnistyksessä (kuinka server ymmärretään), meneekö käyttöönotto ulkomuistista (omia ohjelappuja, online-/paperimanuaali), backup package (roikkuu muistissa), node ID (1 varattu varaosakortille), tärkeimmät parametriryhmät ja parametrit, paperiteollisuudessa prosessin jatkuvuus tärkein; näkyykö omassa työssä, mitä ABB:n laitteita / käyttöjä / softia on käytössä, käyttääkö näppäinoikoteitä, mitkä tärkeimpiä, arvioi ohjelman X ulkonäköä (selkeä, sekava, ..), oletko tyytyväinen tavallisena käyttäjänä, haluaisitko vaikutusmahdollisuuksia työkalun kehitykseen, onko käyttäjillä omaa erityistä sanastoa (jargon), tietoteknisten termien ymmärtäminen, beginner/expert/problem solver/mundane performer, missä määrin työkalu (tuote) määrittää työtäsi, kuinka mieluiten opettelet uutta softaa, mitä teet virhetilanteessa X, hengittääkö asiakas niskaan, käytätkö DW:n personoitua työpöytää, onko nettiyhteyttä (ja muut informaatiolähteet), miksi et halua käyttää softia, millaista tietoa kulkee eri ihmisten välillä tilanteessa X, kuka asentaa softat koneisiin, oletko pettynyt joihinkin asioihin X:ssä, kuinka kauan parametrien asetus (yms.) kestää, oletko käyttänyt ACS800:n assistanteja, mitä muuta softaa käytetään samalla koneella, miksi DW eikä DWL (tai DD) tai päinvastoin, jakauma 800, 550 (tai vanhemmat), softan kanssa ja ilman, oma/laina/yhteislaitteisto jolla käytät ohjelmaa, onko kone kiinni Internetissä tai suojatussa sisäverkossa, luonnostele tehtäviin liittyvä organisaatio, useimmin toistuvat tehtävät, tärkeys, tukeeko nykyinen järjestelmä.

Appendix I: Screenshots



```
0,28 =28 MHV-tela  5  MT                    Ready Running
0,29 =29 Termotela 4  F                     ALARM
0,30 =30 Keskitela 3  F                     ALARM
0,31 =31 Termotela 2  F                     ALARM
0,32 =32 MHV-tela 1   F                     ALARM
0,33 =33 Johtotela 4 ja levitystela 3       FAULT
0,34 =34 Johtotela 5                        FAULT
0,35 =35 Johtotela 6                        FAULT
0,36 =36 Johtotela 7                        FAULT
0,37 =37 Johtotela 8                        FAULT
0,38 =38 Johtotela 9 ja levitystela 4       ALARM
0,39 =39 MHV-tela 6   F                     ALARM
0,40 =40 Termotela 7  F                     ALARM
0,41 =41 Keskitela 8  F                     ALARM
0,42 =42 Termotela 9  F                     ALARM
0,43 =43 MHV-tela 10  MT                    ALARM
0,44 =44 Johtotela 10 ja levitystela 5      FAULT
0,45 =45 Johtotela 11                       FAULT
0,46 =46 Johtotela 12                       FAULT
0,47 =47 Johtotela 13                       ALARM
0,48 =48 Rullaimen johtotela                FAULT
0,49 =49 Rullaimen levitystela              ALARM
0,50 =50 Rullaussylinteri         M         ALARM
0,51 =51 Rullaimen ensiökäyttö    F         ALARM
0,52 =52 Rullaimen toisiokäyttö   F         ALARM
```

```
      +28 99.10 DRIVE ID NUMBER
        0 24.02 DROOP RATE [%]
       40 24.03 KPS
        5 24.09 TIS [s]
  50.0679 23.06 SPEED ERROR FILT [ms]
      120 22.01 ACCELER TIME [s]
      120 22.02 DECELER TIME [s]
       90 22.04 EME STOP RAMP [s]
   1548.7 50.01 SPEED SCALING [rpm]
 -51.6003 20.01 MINIMUM SPEED [rpm]
   1703.6 20.02 MAXIMUM SPEED [rpm]
  5.15878 20.03 ZERO SPEED LIMIT [rpm]
        5 20.11 FREQ TRIP MARGIN [Hz]
       +2 50.03 SPEED FB SEL
        0 50.06 SP ACT FILT TIME [ms]
       50 50.13 ZERO DETECT DELAY [ms]
       50 50.14 SPEED HOLD TIME [ms]
      100 20.05 MAXIMUM TORQUE [%]
     -100 20.06 MINIMUM TORQUE [%]
      100 20.07 SPC TORQMAX [%]
     -100 20.08 SPC TORQMIN [%]
      100 20.09 TREF TORQMAX [%]
     -100 20.10 TREF TORQMIN [%]
       +3 21.01 START FUNCTION
     1500 21.02 CONST MAGN TIME [ms]
        2 70.01 CH0 NODE ADDR
        0 70.04 CH0 TIMEOUT [ms]
  25.9359 1.05 FREQUENCY [Hz]
  100.017 25.07 TORQ ACT FILT TIM [ms]
   60.743 1.12 ACS800 TEMP [C]
  839.387 1.10 DC VOLTAGE [    V]
       +2 26.01 TORQUE SELECTOR
       28 70.15 CH3 NODE ADDR
```

```
☑ 1.83463 2.03 SPEED ERROR NEG [rpm]
☑  61.302 1.07 MOTOR TORQ FILT2 [%]
☑  60.894 2.09 TORQUE REF 2 [%]
☑       0 23.10 SPEED STEP [rpm]
☑ 1151.48 1.04 MOTOR SPEED [rpm]
☑ 1150.76 1.03 SPEED MEASURED [rpm]
```

Interval: 25 ms

Step: 8 rpm

```
1 POWER [kW]
1 VOLTAGE [V]
1 CURRENT [A]
1 FREQ [Hz]
1 SPEED [rpm]
1 COS FII
```

Ref Value (1150 rpm)

-1704 rpm   1704 rpm

Ready   Running   ALARM   FAULT

```
   1548.7 50.01 SPEED SCALI
   1703.6 20.02 MAXIMUM SPE
 -51.6003 20.01 MINIMUM SPE
  5.15878 20.03 ZERO SPEED
      +28 99.10 DRIVE ID NU
     +104 92.02 D SET 11 VA
     +107 92.06 D SET 13 VA
```

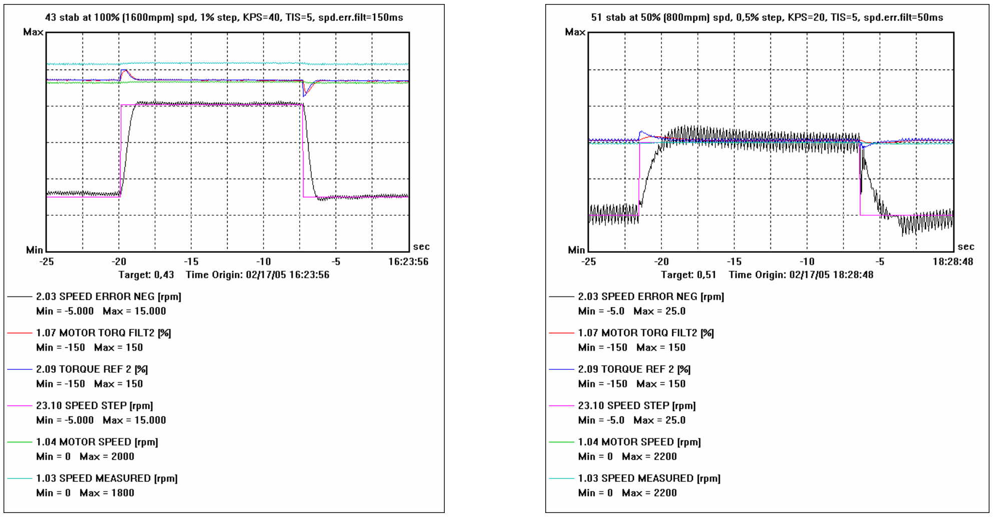
```
       +1 3
       +1 3
       +3 30.03 MOT1 TEMP AI1 SEL
      130 30.04 MOT 1 TEMP ALM L
      150 30.05 MOT 1 TEMP FLT L
       -1 30.24 PPCC FAULT MASK
      130 30.28 THERM MOD ALM LIM [C]
      150 30.29 THERM MOD FLT LIM [C]
       80 30.30 MOT NOM TEMP RISE [C]
  36.6913 1.16 MOTOR 1 TEMP [C]
        0 1.17 MOTOR 2 TEMP [C]
     5000 13.03 FILTER AI1 [ms]
```

```
92.02 = 104
92.03 = 209
92.04 = 802
92.05 = 0
92.06 = 107
92.07 = 0
92.08 = 9910
92.09 = 213
92.10 = 906
92.11 = 0
92.12 = 805
92.13 = 2003
92.14 = 5001
92.15 = 9905
92.16 = 9906
92.17 = 116
92.18 = 117
93.01 = 118
93.02 = 109
93.03 = 901
```

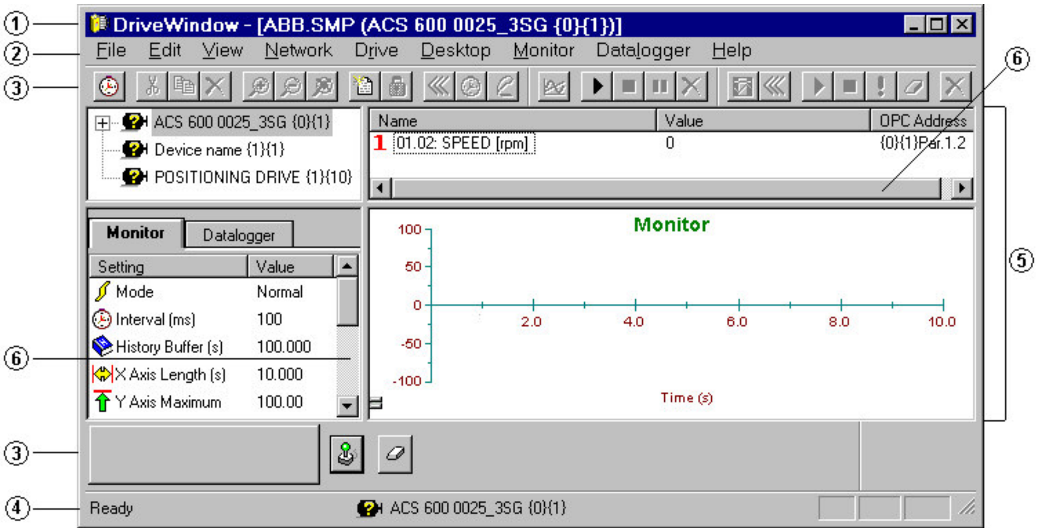
```
90.05 = 0
90.06 = 0
90.07 = 0
90.08 = 0
90.09 = 0
90.10 = 0
90.11 = 0
90.12 = 0
90.13 = 0
90.14 = 0
90.15 = 0
90.16 = 0
90.17 = 0
90.18 = 0
91.01 = 0
91.02 = 0
91.03 = 0
91.04 = 0
91.05 = 0
91.06 = 0
92.01 = 0
92.02 = 0
92.03 = 0
92.04 = 0
92.05 = 0
92.06 = 0
92.07 = 0
92.08 = 0
92.09 = 0
92.10 = 0
92.11 = 0
92.12 = 0
92.13 = 0
92.14 = 0
92.15 = 0
92.16 = 0
92.17 = 0
92.18 = 0
93.01 = 0
93.02 = 0
93.03 = 0
93.04 = 0
93.05 = 0
93.06 = 0
```

```
;  02/16/05 15:19:49
; Ramppiajat
#TARGET = 0,28
22.01 = 120           ; ACCELER TIME [s]
22.02 = 120           ; DECELER TIME [s]
22.03 = 1  ; ACC/DEC TIME SCLE
```

DriveDebug in full action in paper mill commissioning.



DriveDebug trend bitmaps of roll step tests in paper mill commissioning.

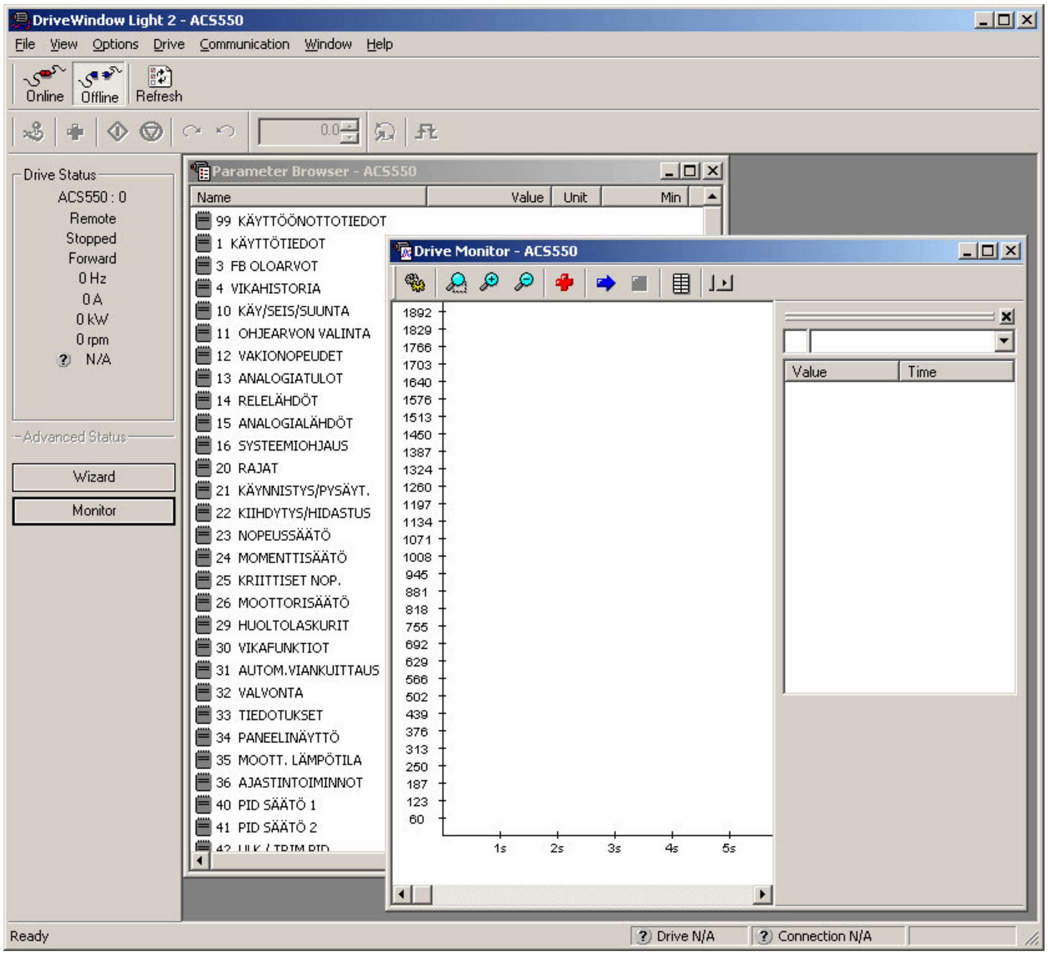

1
2
3
3
4
5
6
6
DriveWindow - [ABB.SMP (ACS 600 0025_3SG {0}{1})]
File Edit View Network Drive Desktop Monitor Datalogger Help
ACS 600 0025_3SG {0}{1}
Device name {1}{1}
POSITIONING DRIVE {1}{10}
Monitor
Datalogger
Monitor
100
50
0
-50
-100
2.0
4.0
6.0
8.0
10.0
Time (s)

Name	Value	OPC Address
1 01.02: SPEED [rpm]	0	{0}{1}Par.1.2

Setting	Value
Mode	Normal
Interval (ms)	100
History Buffer (s)	100.000
X Axis Length (s)	10.000
Y Axis Maximum	100.00

Ready ACS 600 0025_3SG {0}{1}

DriveWindow2.12 screenshot from program manual.



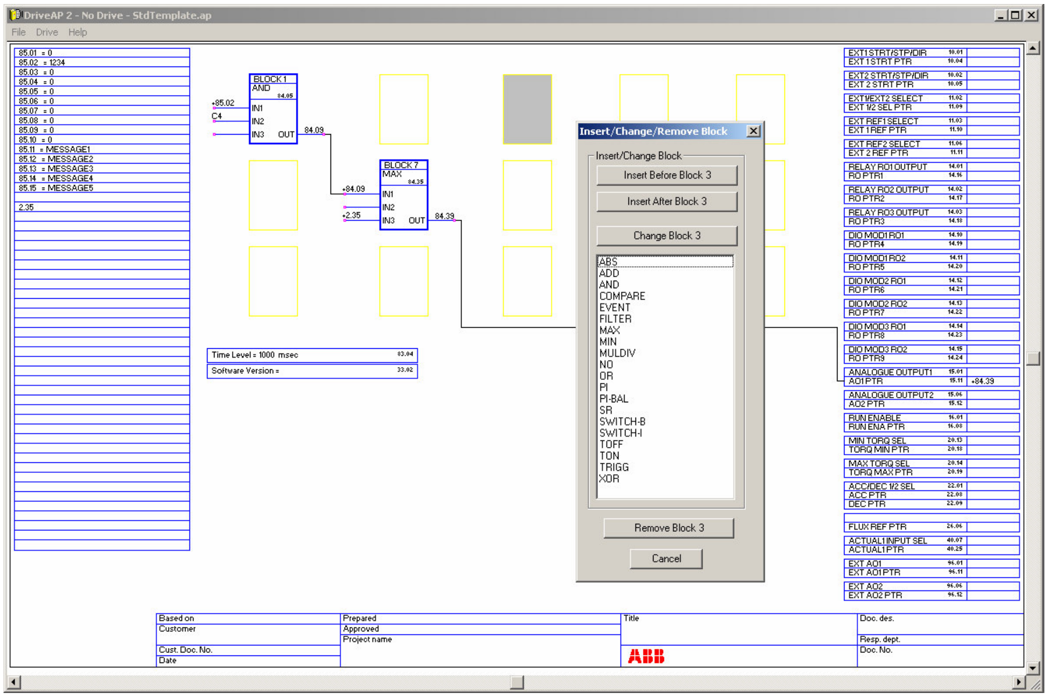
Drive Status

ACS550 : 0
Remote
Stopped
Forward
0 Hz
0 A
0 kW
0 rpm
N/A

Name	Value	Unit	Min
99 KÄYTTÖÖNOTTOTIEDOT			
1 KÄYTTÖTIEDOT			
3 FB OLOARVOT			
4 VIKAHISTORIA			
10 KÄY/SEIS/SUUNTA			
11 OHJEARVON VALINTA			
12 VAKIONOPEUDET			
13 ANALOGIATULOT			
14 RELELÄHDÖT			
15 ANALOGIALÄHDÖT			
16 SYSTEEMIOHJAUS			
20 RAJAT			
21 KÄYNNISTYS/PYSÄYT.			
22 KIIHDYTYS/HIDASTUS			
23 NOPEUSSÄÄTÖ			
24 MOMENTTISÄÄTÖ			
25 KRIITTISET NOP.			
26 MOOTTORISÄÄTÖ			
29 HUOLTOLASKURIT			
30 VIKAFUNKTIOT			
31 AUTOM.VIANKUITTAUS			
32 VALVONTA			
33 TIEDOTUKSET			
34 PANEELINÄYTTÖ			
35 MOOTT. LÄMPÖTILA			
36 AJASTINTOIMINNOT			
40 PID SÄÄTÖ 1			
41 PID SÄÄTÖ 2			
42 ULK / TRIM PID			

Ready Drive N/A Connection N/A

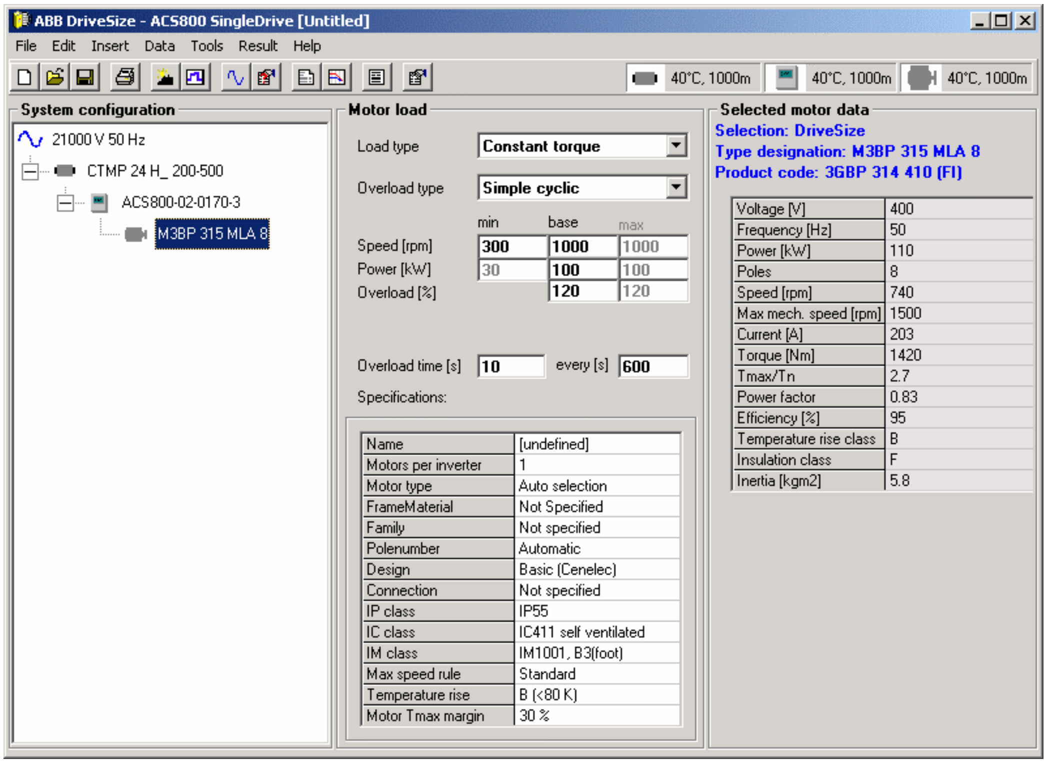
DriveWindow Light 2.31



85.01 = 0
85.02 = 1234
85.03 = 0
85.04 = 0
85.05 = 0
85.06 = 0
85.07 = 0
85.08 = 0
85.09 = 0
85.10 = 0
85.11 = MESSAGE1
85.12 = MESSAGE2
85.13 = MESSAGE3
85.14 = MESSAGE4
85.15 = MESSAGE5
2.35

Time Level = 1000 msec 83.04
Software Version = 33.02

DriveAP 2 in standard mode.



Load type: Constant torque

Overload type: Simple cyclic

	min	base	max
Speed [rpm]	300	1000	1000
Power [kW]	30	100	100
Overload [%]		120	120

Overload time [s] 10 every [s] 600

Specifications:

Name	[undefined]
Motors per inverter	1
Motor type	Auto selection
FrameMaterial	Not Specified
Family	Not specified
Polenumber	Automatic
Design	Basic (Cenelec)
Connection	Not specified
IP class	IP55
IC class	IC411 self ventilated
IM class	IM1001, B3(foot)
Max speed rule	Standard
Temperature rise	B (<80 K)
Motor Tmax margin	30 %

Selection: DriveSize
Type designation: M3BP 315 MLA 8
Product code: 3GBP 314 410 (FI)

Voltage [V]	400
Frequency [Hz]	50
Power [kW]	110
Poles	8
Speed [rpm]	740
Max mech. speed [rpm]	1500
Current [A]	203
Torque [Nm]	1420
Tmax/Tn	2.7
Power factor	0.83
Efficiency [%]	95
Temperature rise class	B
Insulation class	F
Inertia [kgm2]	5.8

DriveSize 2.3 screenshot from program manual.