Efficiency of communication in software user documentation

Maria Lahti
Pro Gradu Thesis
Department of English Philology
University of Tampere
July 2000

Tampereen yliopisto
Filologian laitos I
Englantilainen filologia
LAHTI, MARIA: Efficiency of communication in software user documentation.
Pro gradu -tutkielma, 137 s.
Heinäkuu 2000.

Tutkimuksen tarkoituksena oli selvittää, voidaanko dokumentaation tehokkuutta arvioida viestintäteorioiden, erityisesti Gricen maksiimien, avulla. Tutkimuksen tulosten perusteella Gricen maksiimit ja muutkin viestintäteoriat ovat hyvä tapa syventää dokumentaation analysointia. Dokumentaation käyttämisessä näyttäisi kuitenkin olevan erityisenä ongelmana käyttäjän sisäisten mallien ja dokumentaatiossa esitetyn kontekstin yhdistäminen, ja tähän eivät nykyiset viestintäteoriat anna vastausta.

Tietokoneohjelmistojen käyttäjädokumentaation tulee viestiä ohjelman käyttäjälle mahdollisimman helposti ja tehokkaasti hänen kulloinkin tarvitsemansa tieto. Dokumentaation kehittäminen on pitkälti pohjautunut käytettävyystutkimukseen, jossa pyritään tekemään ohjelmistoista käyttäjille helpompia ja miellyttävämpiä. Käytettävyystutkimus perustuu kognitiiviseen psykologiaan, jonka parissa löydettyjä havaitsemiseen, oppimiseen ja muistamiseen liittyviä säännönmukaisuuksia on pyritty hyödyntämään myös dokumentaation kehittämisessä. Sen sijaan kielitieteellisen tutkimuksen ja viestintäteorioiden hyödyntäminen dokumentoinnissa on ollut vähäisempää.

Tutkimuksessa kartoitettiin aluksi dokumentaation kehittäjien työssään soveltamia periaatteita ja tavoitteita, kuten hyvä käytettävyys sekä sisäiset mallit, joilla käyttäjät mielessään kuvaavat monimutkaisia järjestelmiä sekä niiden toiminnallisia syy- ja seuraussuhteita. Samalla esitettiin myös, minkätyyppisiä tiedon lähteitä ohjelmistojen käyttäjät tutkimusten mukaan pitävät parempana ja siis tehokkaampana kuin varsinainen dokumentaatio: kollegalta kysyminen sekä erilaiset kaupalliset (epäviralliset) ohjekirjat.

Tämän jälkeen käsiteltiin erilaisia viestintäteorioita, joiden voidaan nähdä pikemminkin täydentävän toisiaan kuin esittävän täysin vaihtoehtoisia ajatuksia viestinnän syvimmästä olemuksesta. Käsitellyt viestintäteoriat jaettiin neljään tyyppiin sen mukaan, millaisia taustaolettamuksia teoriat sisältävät viestinnässä siirrettävistä merkityksistä sekä viestinnän tehokkuudesta ja viestinnän onnistumisesta. Teoriatyyppeinä määriteltiin pintamerkitysteoria, kertyvän merkityksen teoria, tarkoitetun merkityksen teoria sekä tavoitesuuntautuneen merkityksen teoria. Tavoitesuuntautuneen merkityksen teorioista yksi esimerkki on Gricen esitys keskustelun maksiimeista, joilla kuvataan tehokkaan ja vaikuttavan keskusteluviestinnän perusperiaatteita. Maksiimien mukaan vuorovaikutuksen tulee sisältää asiaankuuluvaa, oikeanlaatuista tietoa sopivassa määrin ja riittävän selkeästi esitettynä. Oleellista viestinnässä on myös yhteistyö viestijöiden tavoitteiden saavuttamiseksi.

Tutkimuksen empiirisessä osassa havainnoitiin maksiimien noudattamista dokumentaatiossa. Tätä tutkittiin käytettävyyskokeella, johon osallistui viisi henkilöä. Kokeessa ilmenneitä ongelmakohtia analysoitiin maksiimeista käsin. Maksiimien soveltaminen oli tietyssä määrin tulkinnanvaraista, ja tuloksia ei voida yleistää. Testissä kuitenkin ilmeni, että määrän maksiimin rikkominen aiheutti pahimmat ongelmat: jos käyttäjälle tarjottiin liian paljon tai liian vähän tietoa, hänen oli erittäin vaikea tulkita tietoa ja löytää oikea etenemistapa.

Kaikkia ilmenneitä ongelmia ei voitu tulkita yksinomaan maksiimeista käsin, vaan osa juontui koehenkilöiden sisäisten mallien ja dokumentaation yhteensopimattomuudesta. Käyttäjät toimivat sisäisten malliensa mukaisesti, ja etsivät tai odottivat löytävänsä tiettyjä avainsanoja tai tehtävänratkaisutyyppejä dokumentaatiosta. Kun dokumentaatio ei tukenutkaan kyseistä ajattelumallia, käyttäjän oli vaikea hahmottaa dokumentaatiossa esitetty ratkaisu ja sen konteksti. Eri kontekstitasojen luominen näyttäisikin olevan ongelma, jota käsitellyissä viestintäteorioissa ei ole kuvattu.

Contents

1 Introduction

Software use is communication and interaction between the user and the computer. When users are not sure of what they should do at any given point, they will more readily ask another person for advice than consult a manual or an online help facility (according to the studies reviewed by Horton 1993: 26). User documentation does not seem to be regarded as an efficient enough source of information. Consulting another person is preferred; person-to-person conversation seems to be considered a more efficient and reliable method for finding the correct information.

Grice, a philosopher, formulated general rules which enable the exchange of meanings in conversation (1975). He defined a Cooperative Principle (CP) that implicitly governs all kinds of conversation, regardless of the topic. According to Grice, the CP supports a "maximally effective exchange of information" (1975: 47). Therefore, in order to be as efficient as possible for achieving mutual understanding, it is reasonable for the conversation participants to apply the CP. The CP is applied by observing the conversational maxims of quantity, quality, relation, and manner.

Since software documentation is most often used in a situation where specific information is needed, an efficient and effective way to convey it is desirable. Using documentation can also be seen as resembling conversation: users have certain questions that they want the documentation to answer. Therefore, it is my hypothesis that software documentation should follow the CP and the conversational maxims very closely.

In this study, I aim to find out how the Gricean maxims can be used as a tool for analysing the usefulness of software user documentation. For this purpose, an empirical study was conducted in which the use of documentation for three different software applications was compared and analysed in regard to Gricean maxims. On a more general level, it would be interesting to find out whether Gricean maxims, originally formulated to describe spoken

dialogue, can be used for defining instructive written texts in order to make them more usable and accessible. Differences between the traditional structures of written and spoken texts may affect the ease of use of instructive texts, for example, in issues such as referencing, establishing context and turn taking.

The purpose for this study is practical. I work as a technical writer, designing documentation for software applications. My aim as a technical writer is that users will find the documentation that I produce useful and usable, and that their satisfaction in using the software will thereby increase.

This study contains five chapters: this introduction, three main chapters, and a conclusion. In the next chapter, chapter 2, the characteristics of software user documentation are described by the goals, typical uses and users of documentation. A cognitive approach to documentation is presented with issues such as problem solving, mental models and usability.

In chapter 3, the ideas of communication success and efficiency are explored in different communication theories. Gricean maxims are compared to other communication theories, and the applications of communication theories for documentation are considered.

Chapter 4 contains a report of the empirical study that was conducted to test the research hypothesis, that is, whether the Gricean maxims are followed in documentation. Five people were asked to perform a number of tasks with the help of documentation. The test subjects were asked to think aloud while performing the tasks, and the tests were recorded on video. The results of the tests were then analysed in regard to Gricean maxims. An evaluation of the findings follows with a discussion of the usefulness of Grice's conversational principles to this study and to documentation.

Finally, in chapter 5, the study is concluded by reviewing and evaluating the findings on the efficiency of communication in software documentation.

This study belongs to the field of pragmatics with a focus on the language of documentation in the context of software use, especially on how readers (users of documentation) construct meanings from the text in the context of their work. In addition to linguistic theories, I will draw on cognitive psychology for theories of usability and mental models. My sources also include communication research from media studies and social psychology. The research method in the empirical part is qualitative.

This study follows my learning in the technical communication profession. It reflects the opinions and beliefs which I have acquired in the course of my work as a technical writer, based on practical experiences and discussions with colleagues, as well as on reading on technical communication research and related fields. The writing process took a long time on the side of a sometimes too interesting job; my work caused changes in the study and my study caused changes in my working methods. I revised the text as I learned new ideas and tried new practices.

The first impulse for this study was given to me by a colleague, Leena Salmi, who first introduced me to technical communication research in the form of John Carroll's studies on the minimal manual. I am grateful to Leena for the support and valuable comments she has given to my study; I am also indebted to a number of other people who have encouraged and adviced me in this long learning process. I hope that this study shows at least some of the ideas and insights that I have gained from those interactions.

2 Documentation

In this chapter, I review practitioners' views about creating documentation. The fields of technical writing and documentation are first defined and terminology is explained. The requirements and restrictions that users' needs impose on documentation are considered, and the concept of mental models is introduced. The prevailing notion of good documentation is that it is easy to use, which justifies a brief look into usability principles. At the end of the chapter, some thoughts on the discourse of documentation are presented.

2.1 Definitions

2.1.1 Utility texts

If texts were placed in a continuum by their purpose, and one end contained texts that satisfy artistic, self-expressive and social goals, the other could have utility texts that serve a specific, practical, concrete and immediate purpose. At the expressive end texts would include fiction, poetry, and other works of art, as well as exclamations and greetings. Religious texts, while sometimes used for immediate purposes, such as giving names and validating marriages, could be placed closer to the expressive than to the practical end. Somewhere further along the continuum would come texts with a more specific agenda, for example, newspaper columns that reflect the writer's values, or even manifestos with more outspoken views, aimed at emotional or intellectual persuasion in order to make readers interested in their issues. News items could be considered slightly less crafty, but their main purpose could be defined as to build a long-term world view rather than achieve an immediate effect on a hands-on issue. Nearer to the practical and concrete side would come such texts as first aid instructions, labels "push" and "pull" on doors, income tax declarations, TV guides, and nutritive information on packaged food. A continuum such as this is described in Figure 1.

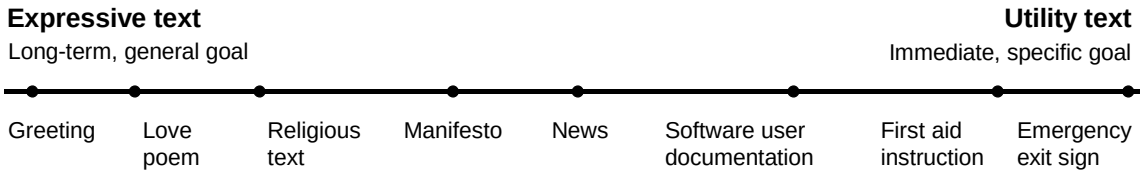

Figure 1. Expressive texts convey feelings, attitudes and values, while utility texts are more like tools.

Software documents are utility texts. The function of utility texts in this continuum can be defined as external or instrumental: the text itself is not in the focus, it exists for enabling or facilitating the use of another product or service. Expressive texts, on the other hand, can be defined as internal to the focus of communication, and they express or reflect the speaker's feelings, values, and attitudes. All texts contain an aspect of persuasion. On the other hand, one could argue that a love poem most certainly has immediate, concrete goals which are related to the physical reality. Even so, poetry often serves a purpose outside of those particular immediate goals, while a software manual would not serve a very useful purpose outside the use of software. Still, mnemonics, alliteration and rhymes are sometimes used for creating memorisable utility texts, but their expressive qualities usually come second to utility.

According to Pilto and Rapakko, utility text, procedural text and necessary text are possible names for this type of text (1995: 37-39). They assert that utility texts are instructional by nature. Cook defines necessary texts as purpose-oriented, information-containing messages written by specialists, often constrained by genre conventions, and having a practical and observable outcome (1995: 15). Widdowson includes in his definition contextual perceptions from the physical world and names them transactional texts after Brown and Yule (Widdowson 1997: 13).

Other text typologies may be and have been defined according to the features one wishes to emphasise. Brown and Yule mention two distinct functions of language, which they name transactional (for conveying information or "content") and interactional (for expressing social

relations and personal attitudes) (Brown and Yule 1983:1; they also report similar dichotomies made by other scholars). These functions of language are together present in all language use, while my typology refers more to the purpose of whole texts. Reiss defined three text categories that are useful especially for creating a translation strategy: informative (factual), expressive (creative), and operative (persuasive) texts (Niska 1999, chapter 4.2). In my typology, the aspect of persuasion is present at both ends of the continuum. A common feature for different classifications is the idea that sometimes texts create their meaning in the context of the (immediate) physical or factual reality, while on other occasions (or at the same time) they serve an expressive or social function with a more general context.

The term "technical writing" usually refers to creating texts that instruct the use of equipment or give guidelines for action in specific situations. A recent definition of the field states that any writing task that requires specialised knowledge is technical by nature and can therefore be included in the field of technical writing (Hayhoe 1999: 24). Thus, for example, origami instructions belong to technical writing along with cooking instructions, grant applications, scientific articles, parts catalogues and operating instructions. A more traditional definition of the field would state that technical writing has to do with technology or technological products. I assume the broader view, and redefine technical writing as the creation of utility texts. Technical writing implies a real-world goal, meaningful tasks with specified tools and a target audience.

Writers of utility texts are assessed to need communication skills more than subject matter expertise of the tools or engineering skills. Technical writers elicit information about the tool (product or service) from its designers, they find out the users' information needs and background knowledge, and then design relevant descriptions and instructions. As technical writers gather information, they usually shift the viewpoint from how the system works (from the production or engineering point of view) into how users can use the service or perform

tasks with the help of the product. Technological jargon and details that are irrelevant for the users should be left out, and new information content is created to support the users' goals. In this way, documentation can be seen as intra-language translation: technical writers translate system designers' notes into end users' language.

2.1.2 Types of software documentation

Software documentation is a partial subset of instrumental or utility texts. The word "documentation" is used in engineering science and elsewhere for any kinds of written documents or records that describe, for example, how things have been constructed or how they work. The relationship between utility texts and documentation is displayed in Figure 2.

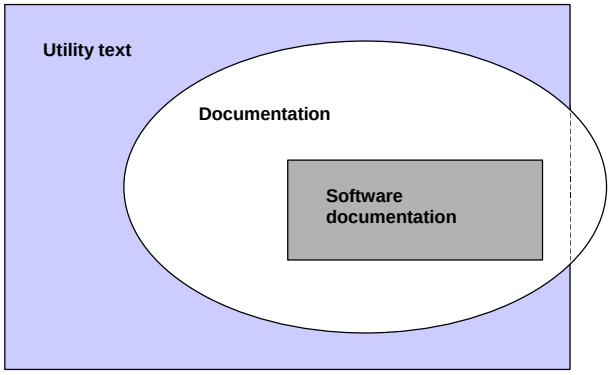

Figure 2. Software documents are utility texts.

Documents or records may also be used for tracking down different processes and their progress. Blueprints, contracts, receipts, order forms and performance statistics are examples of documentation. News, documentaries and even personal diaries are also records in this sense; but they are expressive texts rather than utility texts.

In software development, the word "documentation" is often used for the printed and electronic manuals and other documents delivered to the customer, describing how the software should be used and what the elements of the system are. On the other hand, the word may just as well refer to technical descriptions of software implementation, to project

management documents, or even to marketing brochures. Table 1 presents a classification of documents based on their purpose.

TABLE 1. Types of software documentation.

Type	Audience (reader)	Content	Purpose	Producer (writer)
Design documents Also: technical documentation, technical specifications, product documentation.	Software developers, technical writers, technical support.	Detailed descriptions of how the software code was designed and implemented.	To enable modularity with other software. To enable continued development even if the original developers leave.	Software developers
User documentation Also: customer documentation, end user documentation, product documentation.	End users, help desk, technical support, customers and purchasers, new software developers, marketing staff.	Descriptions about the software and its features; instructions for using the software; information about compatibility issues.	To make the software more professional and fulfill consumer law requirements. To inform users about tasks with the software. To reduce the number of technical support requests. To make users more satisfied with the product.	Technical writers; sometimes also software developers, training people and marketing communicators.
Project documentation	Project's steering group, project members, product managers, marketing officers.	Descriptions of the target market and target audience for the product; comparison of the development effort with the results.	To help assess if a project is worth executing. To convey information about the project to various interest groups.	Project manager and project members
Marketing material Also: product documentation, brochures, fact sheets.	Customers' purchasers, sometimes also end users.	Descriptions and comparisons of software capabilities; examples of how the product saves users' time and money.	To help convince customers to buy the product.	Marketing communicators, technical writers

As can be found in Table 1, writers of each document type have audiences that share their approach or orientation to the product. Marketing communicators write for business purposes and for business-oriented readers, project managers write for the project's steering group and project members, software developers write for technologically able audiences, and technical writers write for end users. Technical writers act as representatives or surrogates for end users in software development, calling for attention to the user's point of view.

In this study, I will use the word *documentation* as short for software user documentation. The word *document* is used for any single independent item of documentation, such as, for example, a guidebook or an online help file, regardless of its format or structure. For the purposes of the study, it is assumed that people use software to perform meaningful tasks for reaching external goals. Playing a computer game is a different matter: using the software (playing the game) is in that case a goal in itself, and thus the requirements for documentation would be quite different.

User documents are often divided into printed and online documents. Printed documents include different manuals and guidebooks, leaflets and quick reference cards. Online documents may be help systems, multimedia files or Internet pages. There are also documents that have been designed for print but are delivered in a file format to users, so that they must print it out themselves. The format of the document, printed or online, is a result of different requirements concerning, for example, costs of production and distribution, users' expectations and their working conditions, and conventions and restrictions of the underlying operating systems. The trend seems to be to put all documentation online, so that printed materials need no longer be delivered.

Usually, online document files have a separate user interface (document browser) that works independently of the actual software application. This means that users can browse through the documentation even if they have not started the application, and they can have instructions on screen at the same time as they are using the application. However, users may not know how to open the document file except by starting it from the application. Further, the help system and the application usually have different user interfaces and navigation methods. This requires more learning and adds to the cognitive load of the user. Users may not even know whether an open window on the screen is an application window or a help screen, or how to get from one to the other.

Online document browsers usually differ between operating systems. For example, Windows Help is a well-known and much used documentation browser, but the Macintosh operating system has no similar standard. Different Unix versions have various graphical and text based help systems. Cross-platform applications (for example, Lotus Notes and Adobe FrameMaker) which look alike in Unix, Windows and Macintosh, may have their own application-specific document browsers that are only used in that particular application. Therefore the user may need to learn how to use a new document browser simultaneously with learning a new application.

Some online document browsers make full text search possible, so that the user can search for information by any word or phrase mentioned in the document. The index search method that uses predefined keywords has the advantage of including synonyms and rephrased terms which have not been used in the document's text.

Some designers have suggested that instead of separating documentation from the application, instructions should be embedded right in the application windows. This would mean that users do not have to split their attention between the document and the application. A possible drawback in this approach is that users may grow annoyed by having to see the instructions on screen even after they have learned to use the program. Various user needs may be difficult to meet: novices, for example, require more, or different, instructions than advanced users. Further, searching and browsing through the documentation becomes nearly impossible if documentation is available only as text in separate application windows. Also, an overview of the features and possibilities of the system could be difficult to communicate in individual program windows. A solution could be to make sure that the menus, commands and other controls are named intuitively, and make use of tips that do not need to be opened and closed as separate user actions. By looking at these cues and hints in the user interface, the user should be able to deduce what the application may be used for. Otherwise, system overview

information should be available through other means, for example, in training. Another approach to embedded instructions are the so-called wizards that guide the user through a complex task by presenting each step in a separate dialog with an explanation of the available choices. Some criticism against existing wizards is that they do not support learning; the user knows that there are quicker ways to do the task, but those are hidden in the menus and hard to find. Therefore the user must always use the wizard in order to perform the task, instead of learning the right menu and command.

Web applications, programs that can be run over the Internet in web browsers, have their own user interface conventions. Since multiple simultaneously open windows are not common in this environment, web applications generally include more text (and more instructions) in a single scrollable and resizeable window. This convention is becoming more widely used in all applications. It means, for example, that text buttons can contain a whole phrase and not just one short word. Localisation is easier, too: words do not need to be fitted inside a predefined space in the window, and abbreviations are therefore not needed. Of course, the visual appearance suffers somewhat, when the window layout cannot be controlled precisely.

All the text that is displayed in the application windows, user interface text, is nowadays considered an important part of user interaction and documentation. Likewise, the development of a conceptual model and user interface metaphor for the application may be regarded as belonging to the expertise of user documentation developers as users' advocates rather than to software developers. (See Kaihu and Rouvi 1999 or Marcus 1998 for more information on user interface metaphors.) The role of user interface text and metaphor is very interesting, and exploring the fine line between application and documentation might even bring surprising results.[1] However, in this study I will focus on documents that are separate

from the user interface; I will study communication about the use of a software application rather than communication with the application.

2.1.3 Terminology used in this study

Software means the programs and applications that run in computers. *Programs* and *applications* refer to the collection of functions, commands and screen elements that are included in the named software entity or package. In technical jargon, "program" stands for a functional sub-part of the software, and "applications" consist of hundreds of programs. However, for the purposes of this study, program and application may be used interchangeably for referring to the software package as a whole and as percceived by the user.

Users or *end users* are people who work with the computer applications and use documentation. I will also refer to "users", rather than "readers" or "hearers", in the context of communicating. The users are not only reading the documentation, they are actively performing tasks with the software application, with the help of documentation.

Documentation refers to the printed manuals and leaflets as well as to the online documents, such as help systems and electronic guides. It can also be understood as any text that is visible on the computer screen, but that definition is not explored further in this study.

Online documents refer to documentation in an electronic format, either as text files, multimedia or help files in the computer, or as information available on the Internet.

Help systems consist of short pieces of information, *topics*, that can be accessed (opened) with commands or buttons in the software application. A well-known help system is Microsoft

---

1. The term *infoware* was used by Lammintaus to compare documentation with hardware and software, instead of regarding it as a part or a by-product of software. As Lammintaus predicts, "the GUI will become a harmonious mixture of functionality and information. The average user will not know, or even want to know, which part of the GUI is 'documentation', which is 'software'." (Lammintaus 1999: 241; see also Winograd 1999: 5.) As it is, many technical communicators already participate in usability and user interface design (Horton 1993: 32, Fisher 1998: 193).

Windows Help that can be opened by clicking a Help button or a command in the Help menu in most Windows software applications.

User interface (UI) stands for the appearance of the computer application on screen. It includes elements such as windows, dialogs, icons, buttons, fields and other controls with which the application is operated. When mentioned in the text, the names of screen elements are written with initial capitals as they appear in the program user interface.

Graphical user interface (GUI) refers to applications that use windowing layouts. In other words, the application can be operated by clicking the buttons and other controls on screen in successive dialog boxes. *Windows*, *dialogs* and *screens* denote the active areas shown on the display unit, containing a set of controls and options that the user may click. *Character based interfaces*, typical before the late 80s, required the user to type in commands with a correct syntax in order to operate the application. Graphical interfaces are considered to be easier to learn and use, since they do not require users to memorise the commands but enable users to pick and choose commands.

User interface text includes names of menus and commands, labels (names) of fields and controls, text in tool tips, status bar messages and system messages. *Tool tips* are short descriptions for fields and buttons, displayed when the mouse pointer stays over the object for a short while. *Status bar messages* are information that is displayed at the very bottom of the window or dialog box. They usually contain feedback about what the software is currently working on. *System messages* mean error messages, confirmation requests and warnings that are displayed to the user in specific situations.

Operating system defines the basic commands and methods for using software in a computer. Typical operating systems are different versions of Microsoft Windows (for example, Windows 95 or Windows NT), Macintosh, and versions of Unix (for example, HP-UX and Linux).

Platform is used here for referring to the underlying operating system.

Pointer is the visual object moved around the screen with a mouse. It enables the manipulation of objects with the mouse. When the pointer is inside text on screen, it changes shape into a vertical line and is often called the *cursor*.

Software developer, *engineer* and *software designer* are used interchangeably to refer to the people who develop, that is, come up with (design) and write (implement), the software code. It may be noted that not all developers are engineers, nor do all designers write code.

Human-computer interaction (HCI) studies *usability* issues, or how software applications could be made easier to learn and to use. Usability, also known as "human factors", is related to ergonomics, which studies product design in the physical context of use and users (Coe 1996: 1). HCI is based on the research of human physiology and cognitive psychology, and studies how product design decisions can help eliminate users' errors and reduce users' cognitive processing time.

2.2 How documentation is used

2.2.1 Interaction between user, application and documentation

The situation of software use means that the user manipulates the computer and interacts with the software. Software applications require initial activation, user input (user action), in order to function. If the user clicks a command, a new dialog box opens on the screen. This is like a dialogue between the user and a computer, and the application windows are actually called dialogs or dialog boxes. Users respond and react to the application's messages and interpret their tone, for example, in terms of politeness. The dialogue consists of verbal turns, such as giving commands by clicking them and displaying system messages or status bar text. Nonverbal behaviour includes, for example, long pauses from the user or computer, system beeps or changes in the pointer shape. Using graphic elements, such as icons, could be

interpreted as nonverbal behaviour or as metaphoric context. Screen elements provide navigational and contextual information; commands and window names provide guides for navigating.

The application asks the user to make choices and prompts the user for providing information, and then processes the information the user has provided. This is interaction between the user and the computer at simplest: the user acts, and the computer reacts. Documentation is used in similar paired turns for eliciting information: the user poses a question or a problem, and documentation responds by presenting a solution or narrowing down the problem area.

As usability studies have indicated, constant feedback is important in the learning and use of any software application (Nielsen 1993: 134-138; Dix, Finlay, Abowd and Beale 1998: 138). For instance, if the user is searching for mail recipients, and the process takes a long time, he[2] must receive an indication that the computer is actually working and has not simply got stuck. Software applications provide feedback to the user on their performance on commands, on the use of system resources and on occurrence of errors. Feedback is provided by displaying status messages, playing audio sounds and changing the pointer shapes.

In documentation, especially in printed manuals, feedback to the user is provided in a slightly different way. In order to find workable solutions to his problems, the user must first be able to verify which instructions apply. The contents of the topic must therefore be explicit enough so that the user can determine which tasks the instructions can be used for. Users typically look for matching words between the document, user interface and their idea of the task. Documentation topics often contain a short description of relevant tasks in a sentence or two, but users do not seem to read those descriptions very often -- a common scenario is that users rush right on to performing the steps, and half-way through start questioning if they have

2. For simplicity, I will refer to the user as "he" in this study, even though women probably use software just as much as men do.

the correct instruction for their task. In addition to verifying that they have the correct instructions, users need information on how they are proceeding with their task. For this, there are check points in the topics. Check points are descriptions of what should be happening on the screen, such as "Click the New button, and the Send window opens." This enables the user to recognise errors or unusual behaviour in the application. The recognition requirements for documentation are depicted in Figure 3.

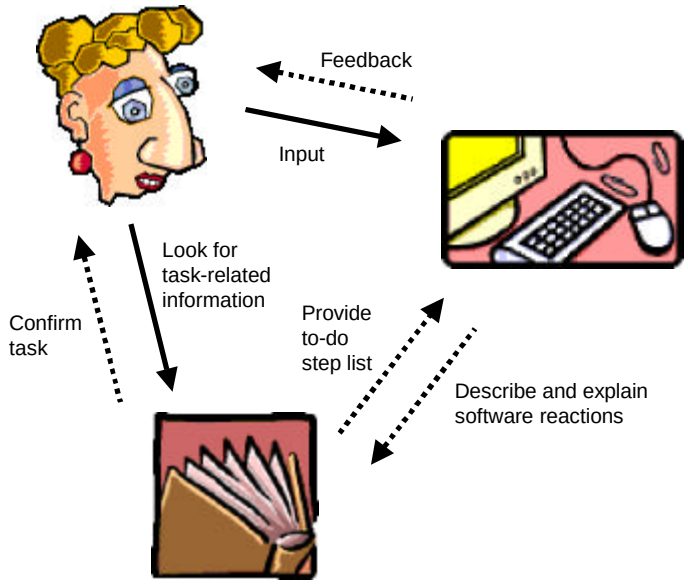

Figure 3. Synchronising documentation with software. References to the application and screen elements should be made unambiguous for the user.

The sequence of the instructions is significant: instructions should proceed at the same pace and order as things happen on the user's screen. If a step (user action) or a check point is omitted, the user may start wondering whether he is in a wrong dialog altogether. This can be compared to driving by an imcomplete map: an experienced driver might notice that the map indicates only some of the roads and landmarks, but a less experienced driver might be too occupied with handling the gears, wipers and indicators to be able to interpret the map.

If an error message appears on the screen, documentation should be able to tell the user what went wrong and how to correct the situation (and how to prevent it in the future). This is

challenging for the writer, because users' actions and probable mistakes must be taken into account in advance. References to system status and tips for error recognition are important elements of active support to the user and complement error recovery information. The writer may also have to make the user notice the most tricky steps well in advance, even before starting a task. This can be achieved, for example, by visually emphasising tips, notes and warnings in the text.

Usually a context-sensitive online help is included in software documentation. Context sensitivity means that help topics are connected with the application windows and controls, so that when the user clicks a Help button, the system displays a topic describing the dialog or field that the user is in. The software application and the document file thus interact, too: the application sends a signal (a "call") for the linked topic, and the help system responds by opening it. If no context-sensitive help is available, the document can refer to individual application windows by spelling out dialog titles and field names. The user can find hints in the user interface text and command names for searching relevant information. Finding information may depend on how well the user is able to use application-specific terminology, and on how well the document is structured and able to lead the user to the correct topic. Figure 4 presents a flow of interaction in using documentation.

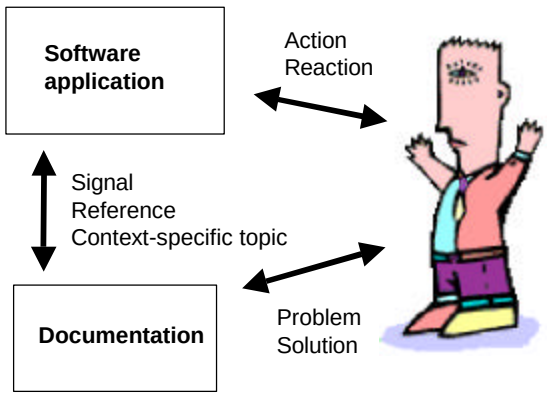

Figure 4. Interaction between user, application and documentation.

In Figure 4, the application and its documentation are displayed as separate entities. It may be argued whether users regard an online help system as an independent party in the interaction or as part of software. Winograd asks:

> Is documentation just a subset of the interaction a person has with a system? Or is it a separate domain, which is "meta" to the interaction design?
> (Winograd 1999: 4)

In this study I assume the latter view, that users regard documentation as separate from the application, guiding the use of the application. Therefore users are assumed to have separate strategies for interacting with documentation (for example, when searching for information) and for interacting with the application for accomplishing other kinds of tasks. "Interaction" implies a more personalised activity that the word "use"; the user is involved in a direct exchange with the software, in which the input and output are unique in different task contexts.

2.2.2 Information eliciting strategies

People typically use documentation when they are trying to accomplish a given task and if no other means of assistance is available. User documents are not read from beginning to end; people look for just enough information to get them going again with their immediate task. All other information is of no interest to them. People are economical and minimise their efforts of learning about the system, and for a good reason too, since most applications contain more features than they will ever use. Users are not interested in documentation as such. It is a necessary evil, and used only if needed.

Even if documentation is not considered an object of interest as such, people do read documentation. Schriver (1997: 213) presents the results of a study on using documentation for household appliances: every four out of five users said that they either scan the manual or read it as a reference. Users tended to use their manuals when they ran into problems or as they tried out new functions (Schriver 1997: 214). If users could not solve the problems they had, they

tended to blame themselves rather than the documentation. Schriver asserts that some of the problems were caused by the user's error, for example, when they misread an instruction (for example, sometimes they did not notice the word "not" in a sentence), or read it correctly but performed an incorrect action, or become impatient and acted without reading to the end of the instruction (1997: 219-220). However, one could argue (or at least consider the possibility) that those are not user errors at all but documentation shortcomings: confusing word orders and unnecessarily complex or boring instructions.

According to studies cited by Horton (1993: 26), the order in which users typically try to solve a problem with software is the following:

- Try and see what happens.
- Ask another user.
- Call the vendor.
- Search the online documentation.
- Read the manual.

Even if some people might be willing to try out documentation before calling a help desk or technical support, many users seek help from a colleague before turning to documentation, especially at the workplaces[3]. Conversation with a more experienced colleague allows the user to elicit information and make references to program parts more easily, and to check his assumptions and understanding about the system with direct and continuous feedback from the colleague. Thus the user can assimilate his knowledge *of* the tasks with knowledge of *how* to perform those tasks with the application. (See Coe 1996: 60 for a discussion of "knowledge of"

3. Hackos and Stevens (1997: 109) suggest that while "insecure users" might tend to ask their colleagues, others would be happy to look for solutions independently. They report that some users indicated a willingness to spend up to half an hour on trying to solve a problem themselves before turning elsewhere for help. However, Hackos and Stevens offer anecdotal information rather than test results to support their view.

and "knowledge how", declarative and procedural knowledge.) Conversation with a knowledgeable colleague is therefore an extremely efficient method for getting help. Documentation, on the other hand, mimics conversation at points, for example, by adressing users as "you" as if in a conversation, and using an imperative tone as if talking the user through a task. Troubleshooting sections (often titled "Frequently asked questions" or FAQ) are written as conversational question-and-answer pairs. A quick comparison between talking to a colleague and using documentation is displayed below in Table 2.

TABLE 2. Some differences between communication media.

Consulting a colleague	Consulting documentation
One-to-one communication; intimate	Many writers and many readers; public
Here and now	Disrupted as to place, time and task
Non-permanent	Easily repeated use
Shared context is wide (task, application, computer literacy, culture)	Restricted context (application)
Familiar vocabulary; easy to define the problem	Jargon; may have to search for some time before finding the correct instruction
Gestures and hesitation are effectively interpreted	Restricted use of nonverbal communication
Linear and cumulative	Nonlinear; may have to jump from topic to topic
Sufficient information with emphasis on the important parts	High probability of getting superfluous information; main points may not stand out from details
Both procedural and declarative information can be discussed as required	Procedures are often separated from overview information
May or may not have the correct answer; restricted information	Should have all the answers; complete information
May be able to recognise and help with software malfunctions (bugs)	Cannot help with bugs
Social activity; subject to conversants' moods	May be interesting and stimulating for example from a visual or technological viewpoint; has the same mood every time

Conversation with a knowledgeable colleague has the main advantage of being more easily relevant in regard to the concrete problem than documentation. On the other hand,

documentation is more likely to have the correct answer. There are differences between person-to-person conversations as well: consulting a dedicated software support person may be just as confusing as consulting documentation, if the conversants do not share the same terminology. The more familiar the colleague is with the user's tasks, software systems and computer skills, the more efficient his help can be. Documentation, on the other hand, should be as easily relevant as conversing with a colleague. Then it could become more optimal than asking a colleague, since its information is more complete. In Horton's view finding external support for software problems, such as a more experienced user or a printed manual, is not an optimal solution because they take more time (Horton 1993: 32). He suggests that as much relevant information as possible should be made available to the user in the product (for example, as user interface text), or at least in an online format, in order to reduce the time required for finding the information. This assumes that the online information is as readily understandable as help from a colleague; otherwise it would be difficult to argue against what he found was the users' best information eliciting strategy, conversation. As mentioned earlier, the idea of an intuitive application for which the user interface text is sufficient documentation is not elaborated in this study. Instead, I assume that separate documents are needed, and that they should aim at the efficiency of conversation in delivering information.

One way in which documentation could become more efficient is by anticipating and supporting the user's own methods for problem solving, which include trial and error, hypothesis testing, algorithms, heuristics and insight (Coe 1996: 121-125). Try-and-see, number one in Horton's list, is comparable to hypothesis testing: the user experiments based on an idea, mental model, of how the system works. Mental models are causal representations of what a real-world object is like or how it functions. Holmberg defines mental models in the following manner:

> A mental model is constructed by an individual to form a symbolic representation of some phenomena in the world. This model is used by the individual in order to solve tasks and problems related to the phenomena. The model both enables understanding and constrains what the individual conceptualises.
> (Holmberg 1995: 17)

If the hypothesis based on the mental model is not correct, the user might try to solve the problem with a heuristic approach or use the insights of a colleague sitting by, thereby redefining his mental model. If these methods do not help, the user needs a foolproof algorithm, provided as step lists in the documentation. And if all else fails, a random trial-and-error method remains; clicking all buttons and pressing all keys until something happens. Algorithms and the random method take more time and are less efficient in problem solving than the previous ones. Therefore it would seem that documentation should support users also by providing a useful basis for hypotheses testing.

An important purpose of technical writing is indeed to provide the users with a conceptual model of the technology they are using. This allows users to attach names to systems and system parts. Naming gives structure to perception and helps the user formulate ideas and questions about the system. As Mårdsjö explicates this purpose:

> The Russian theorist Volosinov (1973) uses to [sic] optical metaphors to distinguish between *reflection* and *refraction*. He asserts that human language not only mirrors - reflects - the world but also refracts it, i.e., creates prisms, tools for creating perspectives, seeing things from different ways and describing them from different angles. . . . The role of the writer is to *cut* reality, to name it and to create images of the 'world' (the technical devices) for the users as the producers wants [sic] them to see it.
> (Mårdsjö 1994: 187)

Cooper suggests that for each application there are three models at work: the implementation model, which is the actual technological construction that makes the application work; the manifest model, which is a simplification or representation of how the program works and is displayed to the user in (or as) the user interface; and the mental or conceptual model that reflects the user's perception of the system (Cooper 1996: 229). Cooper asserts that when the

manifest model resembles closely the user's mental model, the application is considered easier to use and understand (Cooper 1996: 230). Documentation may be of great help in smoothing out the edges of the manifest model. This can be done, for example, with a user interface metaphor, and by offering reasons, analogies and examples to the use of the system. Mental models are thus directly relevant to the efficiency of communication in documentation.

2.2.3 Types of user and information

There are documents called Beginner's Guides, User Guides, Administrator's Guides and Programmer's Guides. These documents vary in the kind of issues that they explain for the users: beginners are assumed to have different computer skills from programmers, in addition to having quite different sets of tasks for which they use the application.

Coe suggests a learning curve with four stages of expertise in the use of technical information: entry, beginner, intermediate and power user (Coe 1996: 47; depicted in Figure 5). In addition, Coe suggests that there is a fifth stage or role, an outsider, that does not use the system often enough to grow more experienced, and is therefore left outside the learning curve. This type of user has also been called "discretionary" or "casual user" (Santhanam and Wiedenbeck 1993, Nielsen 1993: 31). Coe has also combined stages of cognitive and psychosocial development which show a useful correlation to her user type curve (for details, see Coe 1996: 48-56).

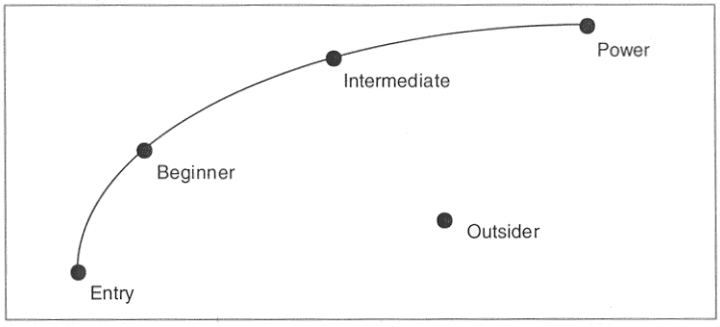

Figure 5. Levels of user's expertise according to Coe (1996: 47).

Even though Coe applies the user type definitions to technical communication usage, the classification can (and has been) applied to skills in using single applications. The far ends of the curve (also known as "novice" and "expert") are often quoted as example user types, but most users actually fall in between. There are great differences in how novices and experts perceive and use applications. According to Coe (1996: 120), novices tend to make lists of surface-level observations, rather than detect symptoms or connect their observations to the "big picture". Experts, on the other hand, automatically prioritise their observations, recognise problems easily, have a very refined mental model of the application and have a large reserve of problem-solution scenarios. This suggests that novice users' understanding of the application is vague and disconnected, and would benefit greatly from explicit support for building a conceptual model. Experts do not need quite as much general overview information. Thus the users' expertise and experience levels also affect the mental models that they form about the application.

Computer literacy is a term that refers to users' knowledge about working with computers. A person is usually said to be computer literate if he knows at least how to use a word processor, a World Wide Web browser and an e-mail application. Kasprzyk observes that in addition to knowing how to perform basic tasks with these applications, one is expexted to be able to learn new versions of the applications without difficulty, and to transfer one's skills to other similar products (for example, to be able to use Internet Explorer as well as Netscape) (Kasprzyk 2000). He gives the following definition of computer literacy:

> Computer literacy has two major components: an *understanding* of computer applications, computer hardware, and social and ethical issues influenced by computers; and sufficient *skills* or *knowledge* of application software to be functional in a computer-enabled environment.
> (Kasprzyk 2000)

This is a rather demanding definition, and more or less describes what technical writers and documentation developers assume of the users' skills. However, users often lack information

outside the use common tasks with specific applications. Application manuals, on the other hand, seldom provide information about hardware or operating systems, since application developers are not responsible for their functioning. It would even seem to be a waste of resources anyway to repeat information that had already been documented somewhere else. But learning about hardware or an operating system has no great appeal to users, and they seldom have enough knowledge to recognise which reactions in their computers have to do with hardware or operating system and which with applications. It would be a waste of users' time to learn everything beforehand if all that they do is write e-mail. On the other hand, knowledgeable colleagues would probably be able to provide this information if required, whereas documentation would not.

Users' information needs can thus be described in terms of domain, system and application skills (Nielsen 1993: 44). This means that in addition to knowing the application they work with and the field (domain) of their task, users need metainformation about how the system works. For example, a document often contains metainformation about how it should be read, what abbreviations stand for, what kind of notations have been used, where the glossary is, and so on. This is information on how the document should be used, and has no direct connection to the subject matter or to the user's tasks. Another example comes from furniture assembling. In putting together furniture, one needs tools and knowledge how to use them in addition to the materials. Instructions for assembling a bookshelf, for example, might or might not include the tip to measure right angles in order to make the shelf firmer. This is the kind of metainformation that a knowledgeable furniture-assembler might tell you, but it is most often not included in the instruction sheet.

Knowledge about system and application means knowing the most suitable method for achieving one's goals. Knowledge about domain means knowing the right goals and performing relevant tasks. Bits of missing information result in suboptimal performance. Users

typically perform their tasks with the methods that they already know, even when quicker and more convenient ones exist. This has been named "the production paradox" (Carroll and Rosson 1987: 82): users are not willing to take time away from their tasks to learn to use their tools more efficiently. In consequence, they lose time by working inefficiently, and their procedural knowledge about using the application is not developing[4].

Among others, Spool (1998) has suggested that a learning curve applies to both application and domain skills independently. Guzdial (1999) notes that in many cases, learning about the domain ('content') is a users' primary task, and learning an application is a secondary but parallel task. He calls this user type learner-as-user, and compares it to two other user types. One is learner-as-learner, for whom learning about the domain is the primary goal. Computer aided training applications should support him intuitively enough without the need to actually learn the applications. The other is expert-as-user, who already knows the domain and primarily needs to learn an application that can help him in his tasks. Van der Meij (1999) transcribed Guzdial's approach into tool and domain expertise, with the categories of full expert (of both tool and domain), part expert (expertise either in tool or domain) and full novice (both to tool and domain). The full novice would probably act like a "parrot" for the first times of using the application: he would repeat the required actions without understanding much either about the goals or the tasks. Users' expertise and their needs for application, system and domain knowledge thus affect the requirements for documentation.

Understanding the system and forming a useful mental model presumes a sufficient understanding of the vocabulary and the phrases used in the software. Users have to be able to adjust their domain and system vocabulary to the manifest model of the application, and

4. There are times when people do work on their procedural knowledge, and that is when they are observing other users. Learning keyboard shortcuts is perhaps the most typical instance: you ask "what was that" when someone performs an action without using the mouse, and you find out that pressing CTRL+SHIFT+ one of the arrow keys allows you to select words or lines of text, or that ALT+TAB allows you to switch between the open applications.

associate and apply the application-specific terminology of procedural instructions onto the screen elements (Nyyssönen 1997: 116, Rasmussen 1988: 183). Especially screen elements may be hard to associate with their technical names, such as 'field' or 'radio button'. The label of the element is a better indicator, because it is spelled out exactly the same in documentation and on screen (for example, 'Address' and 'Age range').

Terminology may vary according to domain, and sometimes according to organisation. This may lead to problems, especially if the application is not in the users' native language (see Pilto and Rapakko 1995 for a description of term-related problems). Single words (or the lack of an anticipated term) may cause confusion or disorientation. For example, I once found myself clicking the Copy command when I should have clicked the Paste command. I thought about handling a copy of the file but, in the user interface terminology, 'copy' is always 'to make a copy' and not 'put the copy here'. (For a discussion on the understandability of computer vocabulary, see Isomursu 1997.)

Translations sometimes produce unfamiliar or counterintuitive terms, such as, for example, the Finnish translation for 'file' ('tiedosto'), which has no connection to the office environment unlike its English original. On the other hand, Nieminen (1994) studied the understandability of a software application in three language versions for native speakers of each language. She found that some of the terms were in fact more understandable as translations than in the original language.

Users' skills and prior knowledge affect their requirements for documentation content and their perceptions of documentation efficiency. However, these issues have been explored by several scholars and will not be investigated further in the scope of this study. The focus of this study is set on communication methods rather than on communication content.

2.3 Requirements for documentation

The software industry is gradually acknowledging the conception of users as experts in their tasks. Human-centred design is an approach that emphasises usability principles and methods already in the planning stages of software development in order to make sure that users will be satisfied with the consequent applications (Denning and Dargan 1996). A user-centred approach is currently considered essential for documentation as well: technical communicators work together with software developers to gain information about the users and to apply that information in their design. Documentation is structured around the users' tasks in order to ensure its ease of use.

In this chapter, I will take a brief look into what the usability approach means especially for documentation development. The requirements for good documentation are considered here in terms of communication and technical solutions rather than in terms of content. The question of documentation quality is a closely related issue, but since it is not directly in the focus of the study, it will be left out from this discussion.

2.3.1 Acceptability

Nielsen has defined usability as part of system acceptability, by which he means users' willingness to actually start using an application (Nielsen 1993: 25). His definition is displayed in Figure 6.

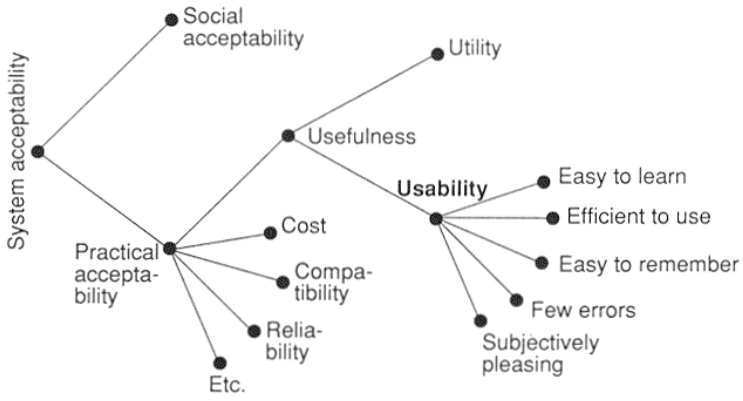

Figure 6. Components of system acceptability according to Nielsen (1993: 25).

System acceptability means that the system must do what the user wants it to do (fulfill utility requirements), with a tolerable processing power and robustness, as well as with acceptable costs for equipment and usage (practical acceptability), in a non-offending and efficient way (social acceptability and usability requirements)[5].

Practical acceptability, such as costs and performance issues, are affected by the efficiency of production. The process of creating documentation should be well-defined and controlled in order to ensure a cost-effective development of accurate, complete and faultless documents. Above all, documentation developers should automatically get all the information that is needed, and documents should be checked for correctness by subject-matter experts. (See Hackos 1994 for an example of a well-defined documentation process.) The usefulness of documentation, just like the usefulness of software, can be determined by its usability and utility. Usability is considered in the next section. Utility in documentation means, for example, that documents can be used in the users' physical environment. For example, hardware documentation cannot come only online, since the user may not be able to start the machine and open the document. Another requirement is that documentation should contain

5. Several similar definitions of usability (understood as Nielsen's system acceptability) exist; for example, usability as learnability, flexibility and robustness (Dix et al. 1998: 162), or the ISO 9241 definition: "the effectiveness, efficiency and satisfaction with which specified users achieve specified goals in particular environments" (in Dix et al. 1998: 192).

instructions for the tasks that the user wants to perform. Troubleshooting tips are the most needed sections of documentation; documentation is not very useful if it only states the obvious or gives very general tips, such as not to run with scissors. Another important utility requirement is that the information is factually correct.

Social acceptability can be verified in target market analyses about the users' needs and values. Many countries have consumer laws that require manufacturers to include documentation with their products. Some countries require that part or all of this information must be given in the official language of the country, which creates translation and localisation demands (Meriläinen 1993). In addition to legal requirements, consumers are aware of their power to require better products and better documentation. They also know that their ethical and moral stances have to be taken into consideration; manufacturers have to use "politically correct" and non-offending material. Software producers sometimes talk about the "user experience" that includes everything in or about the product that the user gets in touch with before, during and after the purchase. Everything that the user experiences contributes to his image of and satisfaction with the product.

2.3.2 Usability

Applications are made easy to use, because opportunity costs (for example, setting up a customer help desk) could easily outgrow the costs of improved usability. Ease of use is also an important user requirement in the competitive software market, since consumers are not likely to accept products that they find hard to use. Likewise, if documentation does not provide answers for the user's questions, the document has no value for the user: it is useless.

Physiological and cognitive research on perception has contributed to usability studies, and theories of cognitive load and mental models are widely used in usability design. Gestalt psychology has provided visual principles for dialog design (Nielsen 1993: 117, Riley and

Parker 1998). Reading research and psycholinguistics have suggested textual design principles, such as, using active sentences rather than passive ones because they are easier for people to understand (see, for example, Slobin 1979: 53).

Usability is achieved by following design guidelines and principles and from observing users as they work with applications (for an example of usability design guidelines, see Nielsen 1993: 20). Usability evaluation methods include observing and interviewing users, either in their typical working environments or in simulated application use tests. User representatives may participate in product design from early on. Usability tests are often arranged iteratively, so that design changes can be validated immediately. The use situation can be studied with storyboarding or in task walkthrough simulations. Usability methods are reviewed, for example, in Nielsen 1993 (chapters 6 and 7), and Dix et al. 1998 (chapter 11). In addition to usability studies, market research, such as investigating the demographic profiles of target audiences, are usually carried out. Documentation can also be evaluated with usability methods, and iterative user testing serves as a basis for changes and improvement suggestions.

Usability design encompasses the whole user interaction design, including visual matters, such as window layout and the choice of colour and font, as well as sequencing, such as the division of items into application windows and the sequence of turns between user input and program output. The ease of navigation, consistent design, accuracy and understandability of feedback, error tolerance and allowing for variation in task performance are examples of other usability concerns. As several researchers have pointed out, users should be able to start working as quickly as possible, without taking time for learning the application or reading documentation first. The user interface should therefore contain clear indications for how to proceed. Uncluttered dialogs are easy to perceive and to find information in. The size of fonts and icons affects the ease of perception. If users can deduce what they can do in a dialog, they do not need to memorise actions: their cognitive load (thought processing need) is minimised.

Set conventions, for example, key combinations, may be used in several applications, which enhances the users' learning and makes their work more efficient. On the other hand, breaking the convention results in users' confusion, errors and slower learning.

Documentation usability covers both physical and cognitive ease of use. Users should find the documents complete, accurate, informative, easy to apply and agreeable. Requirements for user documentation systems have been defined, for example, by Dix et al. (1998: 445) as availability, accuracy, completeness, consistency, robustness, flexibility and unobtrusiveness. Smart (1999:41) discusses usability as the quality of documentation, and defines three main elements: ease of use (i.e., task orientation, accuracy and completeness), ease of understanding (clarity, concreteness and style) and ease of finding information (organisation, retrievability and visual effectiveness). The physical qualities must be acceptable, too; for example, paper quality must be adequate and the print should be legible; it would also be nice if the paper edges did not cause tiny cuts in users' fingers. Nielsen mentions that user documentation is itself a requirement of usable software[6]. Documentation should be easy to search, focus on the user's tasks, give concrete steps for performing those tasks and not be too large (Nielsen 1993: 20). Documentation thus has twofold usability requirements: it should enhance the usability of the application, and it should be easy to use. The overriding principles of documentation usability are perhaps consistency, accuracy and succinctness[7]. The more general notions of legibility, readability and accessibility have been studied widely.

Legibility implies how easily the physical form of text and pictures can be perceived and recognised. For example, blue text is physiologically harder to perceive than text in any other colour. Creating a visually pleasing layout is not enough; there are several user groups that

6. Nielsen also states that clear error messages and user interface texts are vital to the ease of use (Nielsen 1993: 20). In his list of ten usability design principles, four are concerned with documentation in the broad sense.
7. Unfortunately the need for conciseness seems to be noted by others more often than by documentation developers.

should be considered in regard to layout and typography decisions. For example, font size should not be uncomfortably small for farsighted people, and colour blind people may find red and green impossible to distinguish. Legibility is also studied from the point of view of technology and ergonomics, for example, in studies of how the display characteristics of computer screens affect reading speed (Dillon 1992).

Readability implies how much cognitive processing is needed for understanding a text. Issues that affect readability include word choice, syntax (for example, active versus passive mode and the use of negations), the order and consistency of elements and sentence length (Weaver 1988: 309-312). Readability and legibility have been studied with psychological and psycholinguistic laboratory tests by timing differences in user performance. There are also different readability formulas, for example, for indicating syntactical complexity and the difficulty of vocabulary or suggesting the level of intended readers' processing skills. These formulas are not a very useful tool for making texts more understandable (Weaver 1988: 310).

The term "accessibility" is used for the ability of users to access, understand and correctly apply texts. For example, some audiences may not be able to read, or they may have disabilities or a different mother tongue. Therefore the text should be as easily understandable as possible (Chang 1987). Accessibility even encompasses product design: user interfaces should disclose the intended use, and instructions might be given in pictures and numbers rather than in words. The study of accessibility is focused on simplified or restricted vocabularies and syntactic structures. An example is the Plain English (or plain language) movement, which attempts to reduce "bureaucratese" and make official texts, such as contracts and forms, more understandable for laymen and incidental users (Dorney 1999). Another example are minority language laws in the European Union, which guarantee that citizens can receive education and communicate with government officials in their own mother tongue. Accessible texts are assumed to give choice and authority to people over matters that affect

them. Research on simplified vocabularies and controlled language also have implications for second language learning, localisation and machine translation. It is arguable whether a simplified or "dumbed down" vocabulary improves text accessibility for native readers (see, for example, Isseroff 1999). However, with strictly controlled terminology and phrases in documents, costs for computer-aided translation can be significantly reduced.

Even if legibility, readability and accessibility are not considered in those terms in document design, their basic findings are usually included in check lists and advice for writers. A large number of guidelines based on usability heuristics are available for technical communicators (see, for example, Hackos and Steven 1997, Schultz and Darrow 1993, Brockmann 1990).

2.4 Documentation as a discourse

In this section I will briefly consider some notions on the discourse of software user documentation. However, even if the subject is extremely interesting, it is not the main focus of this study and therefore only cursory and preliminary.

Software is used by manipulating the hardware (physical computer equipment) by touching the keys, pressing the mouse buttons or pronouncing voice commands. This physical manipulation has been made transparent: interaction between user and computer is described in terms of interaction between user and software (as depicted in Figure 7).

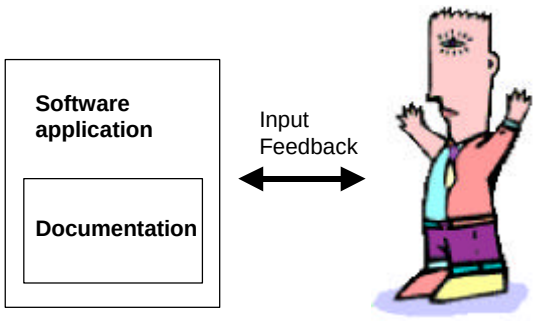

Figure 7. Hardware is invisible. Computer use is described in terms of using the software.

"Computer" thus becomes a metaphor itself; when we talk about using the computer, we usually mean doing things with software. This implies that the computer is just another familiar, everyday household tool, the use of which need not be taught. People are assumed to be computer literate. However, as discussed in section 2.2.3, people do not always have all the skills that go with the definition of computer literacy, and this leaves a gap between users' actual skills and what is assumed of them in the discourse of computers. Therefore the context is not completely shared between the writer and the user (reader), much to the user's disadvantage.

Knowledge brings authority for document writers, who represent the organisation that possesses the complete knowledge about the product. There is a power relationship between the writer and the reader: the user is at the mercy of the writer in regard to what information he receives[8]. Information about the system and the domain are not usually volunteered in official documentation. It is assumed that users either already have that knowledge, or that they are willing to look for it on their own. This inflicts a demanding role for the user, and if the user candidate cannot fulfil all the expectations, too bad; no survival hints are offered. Users even tend to blame themselves if they cannot find the information that they need. On the other hand, unofficial documentation (developed by third parties) often offers domain and system information, and the writer is seen as a helpful information gatekeeper that points out the important parts and leaves out unnecessary details. Third party documents, such as, for example, the Dummies books, is even seen as more trustworthy than official documents, since they can have no reason to ignore unpleasant facts about the application.

Third party documents are preferred to official documentation. Comparisons show that while the official documentation is more thorough and richer in information content, third

8. For a discussion on the roles of authors and readers, see, for example, Montogomery, Durant, Fabb, Furniss and Mills 1992: 223-239, Coney 1992, Coney and Chatfield 1996, Karvonen 1995: 17-19, 22 and Reynolds 1998: 42. For nested levels of discourse, see Rantamäki 1993, section 4.3.

party documents have more of social interaction (Walters and Beck 1992; Coney and Chatfield 1996). This includes a more prominent author persona. Unlike in the official documentation, a Dummies writer introduces himself and his background at length, thus creating believability for his knowledge in the application, domain and system. In Dummies books, the tone is friendly, casual, entertaining and irreverent, while in official documentation, the author's voice is subdued and neutral. Dummies writers are seen as friendly knowledgeable colleagues who volunteer their information, while writers of official documentation appear as rather obnoxious nerds who withhold information unless you know just the right phrase for asking. Likewise, the user may be upgraded to an anarchist with third party documents, and reduced to a novice or learner with official documentation.

Third party books are seen as disclosing solutions to software problems, while official documentation is itself part of software and therefore part of the problem[9]. Karvonen (1995: 12) suggests that textbooks are products of collectives and results of compromise because they have several authors and reviewers who have to come to an agreement about what to print. This is a valid assumption for the official documentation too. The group of authors and reviewers often remains anonymous, and humourous remarks are avoided so as not to offend anyone. Sometimes, though, both document types are produced by the same writers. For example Microsoft has stopped producing lots of official documentation and publishes "unofficial" user guides under the writer's name, as if to show that the documents are unbiased and that the writer is trustworthy.

Third party documents make it quite clear that a company and its software engineers have created the application and are therefore responsible for any bugs and annoyances.

9. The suggested unveiling of problems and solutions can be seen already in third party document names. Compare, for example, the official document "Getting Results with Microsoft Word 97: Real World Solutions for the Work You Do" (Microsoft) and a third party document "Word 97 Annoyances: Taking Charge of Word 97" (Leonhard, Hudspeth and Lee 1997).

Official documentation does not even mention possible annoyances, and therefore the blame for finding them lands on the user rather than on the manufacturer. Users, on the other hand, assign personalities to computers (Cooper 1996: 231; Moon and Nass 1996). The tone of interaction is an important factor in user satisfaction (Ridgway, Grice and Gould 1992). Interaction with documentation is a part of users' interaction with the computer. Bugs disrupt the interaction rather impolitely and abruptly. Users may thus judge "using computers" as easy or difficult, cooperative or uncooperative, based on the interaction style of individual applications and their documentation.

3 Communication

In this chapter, I will first look at definitions of communication, and offer ideas for what kind of communication documentation is. Since my hypothesis has to do with a single communication theory, Gricean maxims, I wish to put that theory in context by reviewing different approaches to communication. I will take a look at their assumptions and restrictions, and reflect on their views on meaning and communication efficiency. In the previous chapter I discussed the usability approach to user-computer interaction, and in this chapter, I will consider interaction from the communication point of view. The communication theories come from linguistics and from media and cultural studies.

3.1 Elements of communication

3.1.1 Defining communication

Technical communicators share knowledge, introduce new concepts and help users apply those concepts in practice. These are communication tasks. The communication should be effective and efficient so that users find it worth their while to study the document. This goal involves an explicit or implicit idea of what communication is and how the efficiency of communication is defined.

There are several definitions of communication depending on the point of view. A dictionary entry of the verb "communicate" includes the following definitions:

> 1. to make (opinions, feelings, information, etc.) known or understood by others, e.g. by speech, writing, or bodily movements
> 2. to share or exchange opinions, feelings, information, etc.
> 3. to pass on (a disease, heat, etc.)
> 4. to join, be connected
> (Longman 1990)

People communicate for more or less specific purposes; the transfer of information is only one of the goals of communication. Others include invoking, expressing or sharing ideas and feelings, making someone perform an action, demonstrating power or maintaining social relations (see, for example, Blakar 1985: 24; Grice 1975: 47; Haslett 1987: 5, 103, 115 for their definitions). Sometimes communication is defined as a directional and intentional activity, but this is far too restrictive. Communication is more than words; clothing, gestures and behaviour may communicate just as much as language (Fiske 1990: 18). In this sense, everything that we do (or do not do) may communicate something to others, and there are no distinct starting or ending points of communication.

In communicating, participants constantly interpret the verbal and nonverbal behaviour of others, in addition to scanning the environment. Social psychologists emphasise communication skills that make some individuals more likely to succeed in interpersonal communication than others (see, for example, DeVito 1992: 91). In order to understand each other's meanings at all, communication participants are assumed to need some kind of shared context (more on this in section 3.1.2). Among other things, culture carries structures and norms for communication that make it easier for people who share the same culture to understand one another. On the other hand, if the participants do not share a common understanding of cultural norms, miscommunication may occur. The information content or feelings may be misunderstood by misinterpreting words, gestures or non-verbal behaviour.

Successful communication has many definitions in theories of communication, each with its own view on creating and interpreting meanings. For example, Levinson (1983: 16) presents the notion that successful communication equals to the recognition of the speaker's communicative purpose or intention. Say that a person hesitates in accepting an invitation. The purpose of his hesitance may a reluctance to accept, and communication is successful when

that intention has been recognised by the other party. This example also shows that language is not a compulsory element of communication.

Communication events may be classified by different dipolar criteria: verbal and non-verbal, written and spoken, mass and personal communication. People communicate face to face or through a medium, concurrently or at separate times. Communication may be linear, building cumulatively on what has been disclosed previously during the communication, or nonlinear or fragmented, consisting of separate, unrelated contributions.

Communication studies are often divided into the study of written and spoken texts. Written and spoken communication differ from each other in information structuring and density, pre-planning, repetition, linearity and the use of extralinguistic cues. Tone and politeness get different realisations in written and spoken communication. These conventions and structures create expectations and limit the selection of available options on the process and on the contents of communication. The difference that all of these conventions make is highlighted in the question: "who reads a book as he hears speech?" (Brown and Yule 1983: 19).

Written communication usually follows a predictable structure. Macrostructures define the arrangement of text parts, such as, for example, exposition, presentation of complications, analysis, and conclusion. Paragraph and sentence level structures include topic–comment constructions or the given–new ordering of items. Written texts are usually regarded as linear, with a beginning, a middle and an end. Written texts are often formal and grammatical, sometimes with complex sentence structures, which may make it difficult to understand written texts when they are read aloud (as, for example, in formal conference presentations).

On the other hand, people's reading habits affect the interpretation of meanings in written text. Even if written texts are regarded as linear, people do not read them from beginning to end; they skim, scan, browse and look at pictures (Weaver 1988: 286, Schriver

1997, Swann 1991). Therefore predictable elements in the visual arrangement (layout) of a text can improve communication. For example, the newspaper structure is very much a standard in most cultures: we know to look for comics at the back and for the editorials on the first pages. In textbooks, there is a table of contents at the front and an index at the back. In software manuals, there is a disclaimer in small print on page 2 and little trademark signs after product names. Poems often have short lines of text. Thus layout elements can both work as genre markers and help readers use nonlinear reading strategies.

Spoken communication is studied by discourse and conversation analysts in terms of conversation structure, turn allocation, power relations and strategies, such as politeness and floor control. Spoken language also includes research areas for nonverbal communication, prosody, pronunciation and phonology. So there are quite different research areas for written and spoken communication, and communication success is not directly relevant to many of them.

3.1.2 Understanding a subject matter

How do we understand each other? The Sapir-Whorf hypothesis says that our language influences the way we think about and structure the world (Mårdsjö 1994: 187; Ellis and Beattie 1986: 60). For example, if we believe that there are four seasons, we might not understand if someone talked about a fifth season. We might think that he made a slip of the tongue. Wittgenstein (1933: 68, 88) even claimed that one cannot talk about concepts for which there are no words in one's language, and that the language sets the limits to one's world. However, new concepts are being introduced all the time, and people do learn new words and acquire new ideas about the world. Introducing a new concept is admittedly difficult. If we have no words to describe something, it is very hard to talk about it, or to properly understand it.

In computer documentation this is an immediate problem. Of course, one can take advantage of metaphors and of anything previously learned. For example, in the field of computer programs, the word "window" has acquired a new meaning that does not relate one-to-one to the concept of windows in a building. As with the concept of "clicking", people have been able to learn what it stands for in the context of computers. There are no correct words for new ideas; there are choices, some of which will remain in use as concepts become better defined. Science, and science fiction, often introduce new concepts through examples: a concept is described (or displayed as a picture), given a name, and the name is then used in context. This appears to be a useful way of providing people with context or background knowledge for communicating the concept.

The participants of a communicative event should have a common understanding of the subject matter and the frame of reference, otherwise meanings and intentions may be impossible to convey or understand. On the other hand, the sharing of knowledge or experience is achieved by communication as well as its prerequisite (Schiffrin 1994: 390). Two concepts that help in defining the subject matter of communication are topic and context. Topic is the immediate subject matter, while context creates the wider frame of reference. Topic and context decide the meaning that specific terms have in the communication.

The concept of context can be divided into general world knowledge ("commonsense knowledge" in Haslett 1987: 86), situation specific factors, and "co-text" (Schiffrin 1994: 378; the term co-text is from Brown and Yule 1983: 46). General world knowledge includes what we know and have experienced about the world, and what we can assume others to know too. This includes anything from natural laws to societal norms and linguistic competence. Presuppositions, implicatures and inferences fill in the gaps between what is communicated explicitly and what is implicitly obvious (Brown and Yule 1983: 28-35).

Situation specific context factors include at minimum a speaker, hearer, place and time, from which deictic references such as *you*, *that* and *now* derive their meanings (Brown and Yule 1983: 27). Other factors include, for example, nonverbal action, setting, channel, key, and purpose (Brown and Yule 1983: 37-39). Participants' status and role, their aims or goals and the social norms for interaction are also included in the situational context (DeVito 1992: 8). People react to what they perceive of the context and adjust their communication accordingly.

One of our main principles for interpreting the general or situational context is that "things will tend to be as they were before" (Brown and Yule 1983: 67). Another principle is to choose the easiest and nearest matching stereotype (Brown and Yule 1983: 58-60). Frames, scripts, plans, scenarios and schemata are this kind of stereotypes or memory units that contain a collection of facts and perceptions related to a situation or environment (Brown and Yule 1983: 238-250; Haslett 1987: 88-90). They are evoked automatically by real-world triggers. They provide expectations (and limitations) for the "default elements", such as, participants, feelings, actions, action sequences and objects, that belong to the situation (Brown and Yule 1983: 236). The difference between stereotypes and mental models is that while stereotypes are like loose collections of items, mental models contain causal explanations between the items about why the objects and actions work as they do.

The role of previous discourse or co-text is defined by Brown and Yule as creating expectations or constraints to what lexical items are possible next (1983: 47). Co-text strengthens interpretation and "creates its own context" (Brown and Yule 1983: 50). Isomursu (1997: 86) suggests that "when a computer manual constitutes the text . . . the computer is a part of the context, and part of the text appears on the computer screen." In this sense, distinguishing between text, co-text and context is irrelevant, since the text also creates the context; context is already internal to and conveyed by the language of the text (Karvonen 1995: 32). Texts are like nodes in a contextual network of related and referred texts.

There are several possible contexts from which participants choose the operative context, for example, by lexical choices and deictic cues that act as relevance markers (Haslett 1987: 124). In this way, individual terms not only get their meaning from the context, but also help participants recognise the context. The operative context is not fixed for the duration of communication, but may vary according to the goals that participants have. Nested contexts allow participants to follow dislocated stories (in regard to time, place and participants) and deduce their internal deictic references without confusing them with the current context (Brown and Yule 1983: 48). Context is related to the concept of relevance. People are assumed to be relevant, to focus on a purpose, while communicating. Participants use their ideas of the context and of relevance to interpret and to communicate.

In documentation, writers' assumptions about the shared context include knowledge of the tasks and purposes that the application can be used for, knowledge of how to use the computer, and knowledge about the terminology. If the user does not share the assumed contextual knowledge, he has to complete the missing information by making his own assumptions and inferences. These assumptions may lead the user to drawing wrong conclusions and selecting a wrong subcontext, for example, a wrong procedure for a task.

Genre conventions help choose the most relevant context out of possible contexts. For example, if the speaker starts with "Have you heard the one about...", hearers will recognise the genre of joke. Genres invoke the mood and attitude for communicating and prepare the communication participants for what can be expected; they create expectations of the form and content of communication (Brown and Yule 1983: 63-64). For example, numbered lists in a software manual are expected to contain instructions for accomplishing tasks. Numbered lists in glossy magazines are expected to contain perhaps rankings of the most enviable celebrities, pieces of diet advice or a quiz for self-knowledge. Genres contain textual and lexical markers, especially specific vocabularies that get certain meanings within that genre. For example, the

first definition that comes to mind of the word "mouse" in a biology textbook would be different than in a software manual. Recognition of the genre comes with experience.

Blakar (1985: 25) wanted to see what would happen in communication if the context was not shared, even though the participants thought that it was. He reports an experiment where a test subject had to communicate a marked route in a map to another test subject, whose map was slightly different. The task was, of course, impossible, and resulted in different ways of coping with the communication failure (Blakar 1985: 26-27). Blakar suggests (1985: 31) that in order to resume successful communication, a reason or cause for the communication failure must first be found. Otherwise the participants cannot trust themselves or each other to communicate intentionally towards their goals, and thus communication will not be considered useful anymore. In other words, if miscommunication occurs, it is very important to recognise the problem in order to resume communication, otherwise there is a risk of losing the communication channel altogether. This is quite a relevant issue in documentation. Users' feelings of confusion or of their expectations not being met can make them very reluctant to open the help or the guidebook again.

Topic is the more focused subject matter of communication, and in documentation, individual chunks of information (especially in the online help) are called topics. In written communication, layout helps to determine where topics shift. Headings reveal something of the content, and heading numbering is another useful cue for deciding which sections belong together. Paragraph boundaries indicate that the text contains closely connected information (Brown and Yule 1983: 95). Conversations include a lot of embedded topic structures, topic shifts and returns to previous topic, as well as false starts, hesitations and repetitions (Brown and Yule 1983: 18, 92). These indicate a low level of pre-planning, which would be rarer in written communication. Spoken communication is thus not as dense in information as written communication (Brown and Yule 1983: 18). Topics are introduced throughout the

conversation, and speakers make their contributions according to what they think the general topic is (Brown and Yule 1983: 90). If a participant is unsure about what the current topic is, he may fall silent so that the speaker would give more information (Brown and Yule 1983: 93). Topic shifts may be marked with changes in intonation ("paratones"), silence, fillers or nonverbal communication such as, for example, gaze (Brown and Yule 1983: 100-106). However, topic shift markers are optional in conversation and do not appear consistently.

Cues to the topic are available from the context and from what we know about the speaker and his motives. Brown and Yule assert that "any consideration of topic involves asking why the speaker said what he said in a particular discourse situation" (1983: 77). Thus determining the topic depends on the participants' analysis of the context and the relevance of what is being said. Once we believe we know the general topic, we interpret all contributions in regard to that topic and make connections based on that assumption (Brown and Yule 1983: 224).

In documentation, references to the context are often given at the beginning of topics in order to make sure that the context is recognised correctly. However, usability studies have indicated that users seldom read references to the context: they start reading at the step list, scanning for words that would indicate whether the instruction is relevant to their needs or not. Their strategy is to avoid wasting time in selecting the context first and the topic next: they want to go into the topic right away. Thus topic, context and terminology form an interrelated network of meaning-making in documentation.

3.1.3 Documentation as communication

Documentation uses conventions of both written and spoken communication. Non-verbal means of communication may also be included, for example as visual cues, system beeps or other sounds. Järvi (1996) has studied graphical user interfaces and their visual elements as

semiotic systems. Pointing with the mouse can also be defined as non-verbal communication; MacAogáin and Reilly (1990) have compared the use of the mouse in computer operation to deixis in person-to-person communication.

Widdowson states that written language (for example, text in the user interface and documentation) is fixed and non-negotiable since the second participant cannot affect its composition (Widdowson 1997: 7). Still, predesigned meanings are subject to interpretation in different contexts. With documentation, users can combine from several sources to create new meanings. Some of the documents may even be updated online, so that their composition is changed, perhaps in reaction to questions from users. Different topics may contain various points of view about the same issue.

Nystrand (1986: 39-40) argues that interaction is happening through written text as well as during a conversation. According to him,

> turn taking is not interaction per se but merely the way conversants accomplish interaction. The interaction of interest is what the turn taking accomplishes, namely an exchange of meaning or a transformation of shared knowledge. In this sense, writers and readers interact every time the readers understand a written text. Conversely, the failure to comprehend means an absence of interaction. (Nystrand 1986: 40)

This way, the user and the documentation interact and bring about the creation of new meanings. (See also Leppänen 1995 for a review of theories on text-reader interaction.)

Ramey compares computer use to reading and hearing, and concludes that it resembles hearing more, since reading includes possibilities for browsing ahead and back, while with computer programs, the user proceeds according to the sequence determined by program windows (Ramey 1988: 146). This suggests that computer use is a forcedly linear activity, like hearing, while reading offers a nonlinear mode and more control to the user.

Turn taking follows rules of self-allocation in conversation (Sacks, Schegloff and Jefferson 1974; Levinson 1983: 296-298). In software use, the user and computer also take

turns in communication, but the turns are allocated differently. The user may not be able to select the moment when his turn is over if the application has a fixed dialog sequence. In that sense, the exchange of turns resembles ritual or institutional (classroom) question-answer pairs or similar exchanges in which choices are rather limited (Tsui 1989, Karvonen 1995: 26).

However, in interaction with the application and documentation, participants also have the opportunity to interrupt, request or offer more information if the other participant does not react as expected. For example, tool tips appear over a user interface element when the pointer stays over it for a little while; falls silent in wonder, as it were. Another example are notes and warnings: they are used in documentation for "interrupting" the user with unexpected information, emphasised with boldface or a different colour font, so that it stands out from the rest of the text. In software, the problem of warnings can often be solved by coding the warning inside the application so that it appears on the screen when required and the user must click a button in order to acknowledge and hide the warning. Numbers in step lists make the user attentive to all the actions that are needed for the procedure.

Documentation can thus take an (inter)active role and elicit information from the user as well as suggest or point to other topics that the user is likely to be interested in. Troubleshooting and diagnostic documents ask questions from the user, and cross references help the user in navigation. These questions, cross references, cues and pointers lead the user to related information; they act, so to speak, as offers to change or redefine the topic of interaction. Conversation, unlike hearing, is not forcedly linear, and likewise, interaction with documentation as well as interaction with the application are not always a forced one-way proceeding. The linear and nonlinear elements vary inside topic (linear) and between topics (non-linear).

Hypertext is regarded as a nonlinear text type; for example Pilto describes hypertext as a nonsequential network of topics (1995: 108). Hypertext consists of nodes (chunks of text) that

are linked together to form a network in which there are several possible paths through the nodes. The main idea in hypertext is that text chunks are self-contained and modular, and do not require reading in a particular order. The user receives information to his immediate problem with a few clicks towards the direction that he is interested in[10]. Topics may also be nested through linking. Links are intended to mimic associations that people naturally make in their thoughts; they should be an intuitive way for people to find information. However, the problem in associations and links is that the producers of hypertext may not make the same associations as users do. Thus linking becomes arbitrary, and instead of following a focused line of thought through the network of text, the user may get lost in irrelevant details and seemingly unrelated issues.

Conversation can be seen as an example of hypertext. New topics are constantly introduced, and participants can choose which ones to follow up. In this sense, interaction with documentation can be seen as resembling conversation more than reading.

3.2 Theories of communication

In language studies, communication has been approached from various points of view. How meanings are understood (semantics) is one approach, others include the acquisition and development of first and second language skills, the structuring of communication, and "doing things with words". For example, Schiffrin discusses several branches of discourse analysis, distinguished by their methodology and focus (Schiffrin 1994: 2-12). Some of the branches (ethnography and interactional sociolinguistics) research the social and cultural aspects or "local interpretations" of communication and use very detailed contextual data. Conversation

10. By reducing repetition, the principle of information modularity is assumed to reduce writers' workload in updating information as well as decrease the costs of translation. Modularity is also applied in paper manuals: information is organised to be read when needed, not from beginning to end.

analysis looks for significant factors in patterns or surface structures of communication, such as, for example, how or why a conversation is organised in turns. Pragmatics is a more theoretical approach into the mechanisms of how understanding can be achieved, and draws on intuition rather than recorded data. Somewhat different assumptions and communication models are applied by different researchers (Schiffrin 1994: 386).

Various fields besides linguistics are interested in aspects of communication. A recent overview presents seven disciplinary traditions: rhetorical, semiotic, phenomenological, cybernetic, sociopsychological, sociocultural, and critical communication research (Craig 1999). The rhetoric tradition is concerned with how the most persuasive message for each audience can be created. Technical communication teaching in the United States is largely based on rhetorical study. Semiotics is interested in how sign systems (both natural languages and artificial codes) can mediate meanings between people. Phenomenologists are concerned with the experiencing of self and otherness and the authenticity of communication. Cybernetics studies information processing in systems, both technological and human. Cybernetics looks into the flow of information and shows how gaps in the routing can cause miscommunication. The sociopsychological approach is interested in interpersonal and group communication and in how people influence each other through interaction. Communication style is seen on one hand as a character trait and on the other hand as the skill of influencing other people. Sociocultural communication research looks into how communication creates and enforces societal norms and values. The critical tradition is a reflexive approach that studies the discourse of communication study and how that discourse enforces the values and power relations that are present in the social order.

Scholars have acknowledged that a single theory cannot be comprehensive enough to cover all instances of communication (Fiske 1990: 4; Sperber and Wilson 1986: 3). The definition of successful communication cannot be the same, for example, for art, dictionaries,

cocktail parties, novels, timetables or greetings. The different traditions of communication studies therefore complement rather than contradict each other. They illuminate different aspects of communication and its efficiency.

I will next present four types of communication theories that differ in regard to how meanings emerge from communication. I will also discuss the theories' implications to communication efficiency in documentation. I have grouped the theories under titles of surface meaning, acquired meaning, intended meaning and goal-derived meaning. Various scholars have made other groupings of communication, for example, into process school and semiotic school (Fiske 1990: 2-3) or into code models and inferential models (Sperber and Wilson 1986: 2). Schiffrin (1994: 386) makes a distinction between between code, inferential and interactional theories of communication, based on how the roles of participants, message, medium and intersubjectivity are considered. By comparing and contrasting communication theories I hope to illuminate the assumptions and restrictions behind individual communication models, and find out where the models can best be applied.

3.2.1 Surface meaning

The traditional approach sees communication as a linear transfer of predetermined meanings from speaker to hearer. The success of communication depends on how accurately messages are transferred. Many scholars from different fields share this theory (see Laitinen and Taramaa 1994: 11; Ellis and Beattie 1986: 3; and Levinson 1983: 11). Language, just like any code, carries the meanings directly, even if semantic variance may occur. Successful decoding of the message ensures successful communication.

Code model

Perhaps the most widely known theory of communication is that of Shannon and Weaver, published in 1949 (Fiske 1990: 6-13; Sperber and Wilson 1986: 4). According to their theory, the message is encoded into a signal, sent along a channel and decoded by the recipient. Communication is linear from the sender to the recipient(s). The model is displayed in Figure 8.

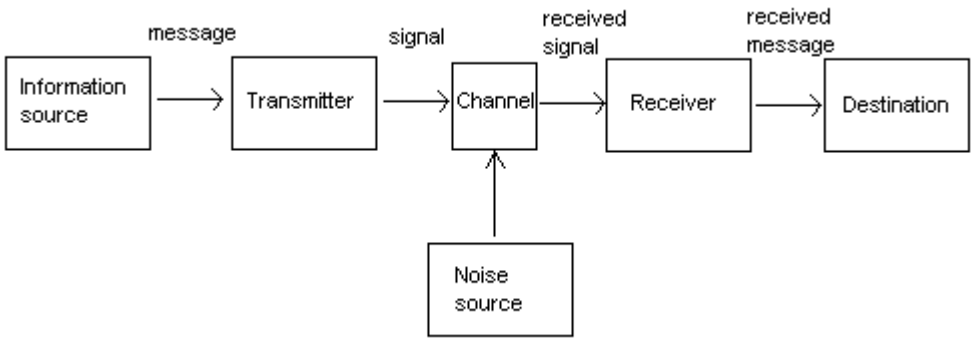

Figure 8. The code model by Shannon and Weaver.

According to the model, the sender's meaning is contained in the message, and the recipient should be able to transcribe the original meaning from the message. An explanation for possible communication errors is given: disturbance or "noise" in the channel can make it more difficult for the recipient to understand the message. Noise can mean anything that causes problems in understanding the message; for example, a bad phone line, too small print, a confusing layout or excessive wordiness can be interpreted as noise, because they make it more difficult for the recipient to understand the actual message (Schriver 1997: 7).

The concept of noise accounts for faulty transmission of messages; other communication failures can be assigned to faulty encoding or decoding, or to a mismatch between the encoding and decoding processes (Ellis and Beattie 1986: 4). In documentation, problems of encoding and decoding (and their mismatch) may be understood, for example, as

competence differences in the natural language or in computer jargon. For example, documentation may be written in English while the mother tongue of the user is Finnish. Or, a manual for administrators may include a lot of specialist terminology, which the reader may not be familiar with. In these cases, the reader might not be able to connect correct meaning to the code of the message. An encoding problem may occur, if the writer of documentation is not sufficiently familiar with the subject area of the manuals, and uses wrong terminology. Again, different software programs may use the same term for very different functions which inflicts a mismatch problem in the encoding and decoding.

Shannon and Weaver originally developed their model for telephone conversations, but it was assumed to describe any type of human communication (Fiske 1990: 6-7). Shannon and Weaver distinguished three kinds of problems in communication: technical problems (transmitting the symbols accurately), semantic problems (the symbol–meaning relationship), and effectiveness problems (how much the conduct of the receiving end is affected by the message). Shannon and Weaver were mostly concerned with improving the technology for transmitting messages.

The basic elements of the model are code, message and channel. The model relies on the assumption that recipients can understand senders' meanings perfectly if there is no noise in the channel. The efficiency of communication thus involves an accurate transfer of the message from sender to recipient. Communication success means effectiveness and it is determined by how the recipient is affected by the message. Fiske asserts (1990: 7) that "Shannon and Weaver see communication as manipulation or propaganda: that A has communicated effectively with B when B responds in the way A desires."

In documentation, technical writers hope that users will be able to perform their tasks with the provided instructions, and correct user behaviour could therefore be a good definition for the success of documentation as well. However, with documentation the user (recipient)

defines the subject areas that he wants to find information about. Therefore the writer (sender) is not the actual initiator of communication, and cannot predetermine how the user ought to perform and which the tasks are exactly that he should complete. The writer can only offer information, but the user must decide what it is that he wants to do. Therefore it is not the writer but the reader who must define the task and choose the course of action (in other words, how he responds) based on the information provided in documentation.

An application of the code model can also be found in software functionality. Online documentation is made context sensitive by linking the topics into software screens or other elements with "calls", references to the software code. If there are problems in communicating the calls or errors (bugs) in the code, the correct online documents may not be available. Also, if the user uses a wrong code for displaying documentation, for example, presses the F3 key instead of F1 for a Help screen, the program interprets the code in a different way from what the user intended. According to the model, encoding and decoding are the problem sources in these instances.

Feedback loops from the destination to the source have later been added to Shannon and Weaver's model (Fiske 1990: 21-22). Feedback allows senders to modify their message according to the recipients. In face-to-face communication, providing feedback is relatively quick and easy. The speaker can notice just by looking at the recipient (or by judging their verbal and non-verbal behaviour, for example, the length of silences) if there is a communication problem. Recipients can ask for clarifications directly, for example, if the phone line is bad and they cannot hear what the speaker said. In software use, however, such corrections are more difficult. The participants have to use the available context for determining, for starters, whether something went wrong. The computer usually provides some feedback as status bar messages or pointer shapes that can help the user deduce whether they are progressing in their intended direction. For example, if an online help screen does not

appear when the user presses the F1 key, he may try pressing the key again just to make sure that a stiff keyboard is not a source of noise in communication. The program may provide feedback as a message such as, "No help is available for this item" or as changing the cursor into an hourglass icon to denote some progress, if slow, in the program. Software programs usually have no way of recognising communication errors. For unusual commands or commands that lose data, the program may ask for confirmation in order to prevent unwanted actions stemming from communication errors. However, usually it is up to the user to act on possible communication errors. Often the best method is to repeat the action.

In documentation, certain type of feedback can be anticipated by linking. For example, term definitions and further explanations about subtasks are commonly needed. Cross references and hyperlink lists provide access to topics with information about related subjects. Other feedback methods include check points for actions that describe what should be visible on the screen after each step. Pointing to screen items can be interpreted as a request for information, and field specific ("What's This") help or tool tips display an explanation.

Further attributes of Shannon and Weaver's model include redundancy and entropy (analysed by Fiske 1990: 10-15). Redundancy helps in determining what the message is about; highly redundant messages are highly predictable and contain little new information. Conventions cause (or provide) predictability in messages; for example, greetings are very predictable and contain very little information. Redundancy helps to minimise the effect of noise in communication. Context is another source of redundancy, providing a fixed frame of reference for new contributions. Entropy, on the other hand, means high unpredictability. If communication is entropic, the participants (or recipients) do not know what kind of information is about to come next. Therefore participants have to pay more attention and make a greater effort in order to make sense of the communication.

In documentation, entropy is most obvious in indexes and other keyword lists. Users often cannot be sure, just by reading single keywords, whether those are at all connected to the subject matter they are interested in. Another issue is that users often find and even expect documentation to be entropic. The principle of modularity brings independence and discontinuity between succeeding topics. Users on the other hand focus on their problem and interpret everything from that point of view, which makes it rather hard to notice the irrelevance of the topic. And after users learn to expect entropy, they become more distrustful as to the relevance of the topic: unless the terms are exactly the same that the user has in mind, he may not realise that the topic is actually addressing his current problem.

Redundancy is sought in documentation especially in using a consistent structure and uniform phrases. If the user opens a page, they should be able to tell which part of the text contains step-by-step instructions and which contains references to related topics. Uniform phrases help the users recognise or remember what action is required from them. For example, if the phrase for clicking a menu or choosing a command varies from case to case, the user may start suspecting that the required action is somehow different in each case.

Redundancy by repetition is a problem in documentation. Users cannot be expected to know how to use the program or even the most common software commands, yet they should be offered task specific information. Most tasks require performing several subtasks (in the logic of the program), but instructions for the same subtasks cannot be written again for each new higher-level task. Therefore, cross references have to be provided, and this means that the user may have to jump forward and backward in documentation, which requires motivation and effort, and sometimes results only in the user's exasperation.

Heuristic rules

There are other ways to account for meaning transfer in messages. Different heuristics or check lists on message content aim to ensure that meanings are transferred efficiently. One such heuristic is the "5W" rule, often assigned to newspapermen: one should tell what happened, to (or by) whom, when or where, why and how (Tompkins 1982: 43-44). This list was developed in 1945, and still, an understandable piece of news is created if all the check list items are covered. The message can thus be made sufficient and complete.

The same kind of rules or check lists can also be applied to documentation: once the required issues are communicated, the message becomes complete, useful and efficient. Hart (1996: 139-145) describes how the "5W" rule can be applied in documentation and how it answers the needs of users. Hart suggests that the "what" answers should identify user's tasks and actions; that the writer should ask the "who" question about the audience, and on the other hand making it clear to the audience who is responsible for each discrete input and output (either the user or the software). "Who" should also identify contact information for technical support. Further, "where" should answer questions about locating the commands, icons, fields and other items in the software. It also means giving a navigation guide to the information content of the documentation. "When" requires explaining clearly the order or sequence of events, as well as stating the prerequisites to all tasks. The "why" answers provide motivation for required user actions, and of course troubleshooting information ("why does this not work?"). The "how" question is answered in step lists.

This kind of heuristics is useful in making sure that the information content is sufficient and that the right information is included. Question lists, document templates and skeletons and suggested tables of contents provide check lists for creating the message content. The check lists are task-specific: there are individual check lists for different communication tasks

or subject matters (for example, a project plan template or a suggested table of contents for a User's Guide).

Heuristic rules are concerned with creating the message. Shannon and Weaver's model is concerned with transmitting the message, and heuristic models aim at creating a strong enough signal to make sure that the message reaches the recipient. In other words, the message volume (both amount and type of information to be included in the message) is the variable that can and should be affected by the sender.

It is assumed that the readers' needs will be satisfied if the sender conscientiously fulfils the requirements concerning the amount and type of information. The writer's main task is that of an information provider: he must make sure that nothing important is missing from the document. Sometimes this may lead to an overflow of information, when everything even remotely connected is included in the documentation in an effort to satisfy all possible information needs. The writer is then no longer an information gatekeeper that would protect the user from unrelated details.

Heuristic models address problems of users' information needs and those of structuring the information, but they do not address problems of meaning, significance or interpretation between the writer and the audience. The idea of communication is a linear flow from creator to recipient, a simplified version of Shannon and Weaver's model (for example noise is not accounted for in heuristic models). The efficiency of communication is derived from the ideal that everything is included from the check list of requirements. The success of communication depends on how accurately the check lists or document templates satisfy the user's information needs. Communication can be improved by improving the check list.

3.2.2 Acquired meaning

Several theories on communication have looked into the semantics of the message. These theories acknowledge that meanings are not unambigously conveyed by the language (or code), and that factors outside the message itself affect the understanding and interpretation of meanings. Therefore communication cannot be seen as a straight-forward linear transfer of meanings, but rather as a nonlinear process that depends on the participants' individual personal, situational, social and cultural contexts. Messages and language inherently carry several levels or layers of meaning, and it depends on the interpretation which of the meanings become more prominent than others. Recipients' reactions may be anticipated by the sender; the direction of meaning transfer is not only from sender to recipient, but also from the context to sender and recipient. All communication is in this sense connected in a network of intertextuality: everything that has been said before affects new messages, and new messages induce new interpretations of old messages.

Applied code models and intertextuality

Applied models have been developed from Shannon and Weaver's original code model, sharing similar assumptions about the sender and the recipient and messages transmitted between them. One such applied model is presented below in Figure 9.

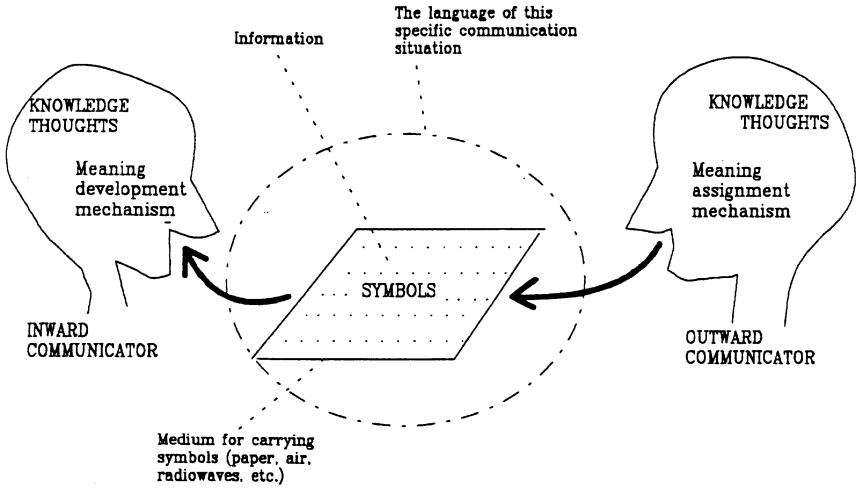

Figure 9. An applied code model of communication (from Laitinen and Taramaa 1994: 11).

This model takes into account the individual contexts of both sender and recipient. A more formal model of the same type was created by Berlo already in 1960 (Wood 1982: 26-27). He defined factors that affect the transferring of meanings from the creation of the message to its interpretation. These factors can be examined by the element they affect:

- Source and receiver: communication skills, attitudes, knowledge, social system, culture
- Message: content, elements, treatment, structure, code
- Channel: seeing, hearing, touching, smelling, tasting.

Thus encoding and decoding of the message are affected by the individual characteristics of the person doing the encoding or decoding. These characteristics include personal values, attitudes and knowledge, as well as social and cultural awareness. The way of sending and receiving the message also affects communication: all five senses are available channels. These properties of the message itself are not value-free; they have an effect on its interpretation. Another way to describe the aspects that affect the encoding and decoding of messages is in Figure 10.

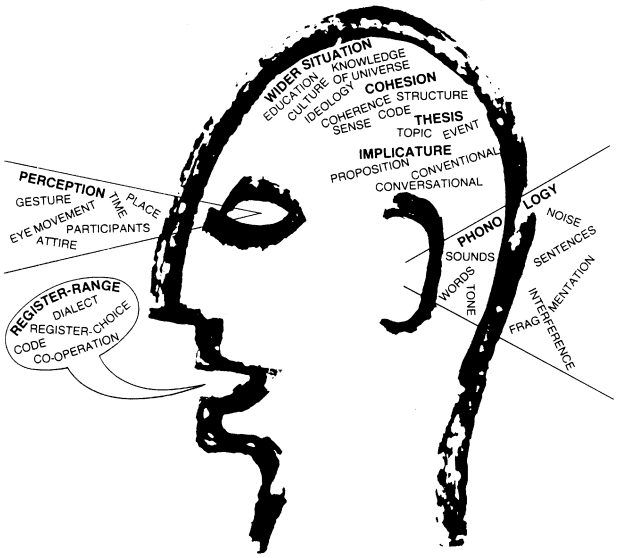

Figure 10. An individual context for communication (from Swann 1991: 21).

Applied code models are concerned with context and perception that affect the processes of encoding and decoding. The models make the point that meanings cannot be conveyed unambiguously, because they always depend on the individual background and perception of both sender and recipient. Messages always require interpretation: meanings cannot simply be deduced from the message. This is a point that reading and translation theories elaborate. Especially in cross-cultural communication, interpretation (translation) has a big role and one-to-one meanings are nearly impossible to transfer. These definitions assign more prominent roles to the message and to the participants as interpreters of meanings. Ultimately, individual experience is allowed for all participants. (See, for example, Leppänen's [1995] account of reading theories.) Misunderstandings do not mean communication failure, but indicate for instance cultural differences (Fiske 1990: 2). Cultural imagery and social background knowledge induce connotations and references.

The enhancement of applied code models to Shannon and Weaver's model is really only an elaboration of the encoding and decoding processes. The overall idea in applied models is still that messages are transferred and meanings are present in the message. In other words,

direction and meaning are still the most important factors that define communication. The success of communication in these models could derive from understanding the cultural aspects. The better the participants understand each others' cultural backgrounds, the easier it is for them to communicate. The efficiency of communication follows from sharing the various meanings included or referred to in the message, which means that no misunderstandings arise.

In documentation, aspects of culture and intertextuality are tackled in the concepts of localisation, internationalisation and computer literacy. In localisation, documentation and the software user interface texts are translated to another language, and other changes are incorporated into the software code in order to create a functional frame of reference for the software. Examples are supposed to work in localised software as well as in their original cultural environment. It would make no sense to talk about American tax laws in a Finnish version of a spreadsheet program, unless the intended audience were companies that operate in the United States and need that particular information. Texts that appear in the screen shots of an e-mail program should follow local conventions for e-mail style, politeness and tone. Humour, above all, is very culture dependent.

Internationalisation means that the application and its documentation are designed in such a way that they include as few culture dependent features as possible. This is to avoid potential problems for users in other countries, and to keep localisation costs down. It may result in less identification and commitment of the users, but since localisation is a costly and time-consuming effort, many software producers are willing to take that risk.

Computer literacy includes knowledge of terminology. Just like in second language learning, new terminology can cause difficulties for the user. Software manufacturers must decide whether to use the same terms as others in their field. Operating systems introduce the user to basic terminology, and thus create a frame of reference to which other applications

should confirm. Otherwise misleading intertextual cues may be introduced, and the user is led astray.

Non-linear models

Semiotic models of meaning or signification have been created by Saussure, Peirce, and Ogden and Richards (Fiske 1990: 41-45). Their meaning theories share three elements: the "real-world" object, the mental image that a person has of the object, and a sign that represents the object. All three elements relate to each other. Ogden and Richard's theory of meaning is described in Figure 11. In their model, "referent" refers to a real-world object, "symbol" to the sign and "reference" to the thought or mental image.

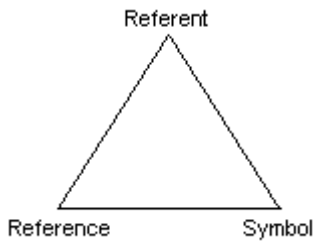

Figure 11. Ogden and Richards's model of meaning (from Fiske 1990: 43).

Fiske applied a semiotic model to communication (described in Figure 12). Elements in his model are defined according to their relationships with other elements rather than by their places in a linear communication process. Fiske suggests that when messages interact with real-world objects and with our images of them, meanings are created. He draws an analogy between message and symbol in the triangles. Fiske presents a meaning creation structure in which the participants of communication hold the same position: "Producing and reading the text are seen as parallel, if not identical, processes" (Fiske 1990: 3). In other words, participants create their meanings in the same manner during communication.

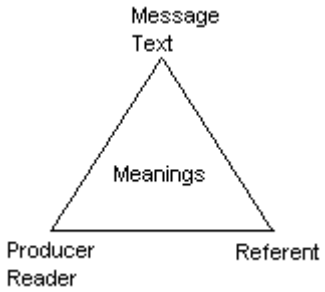

Figure 12. Structural model of communication (from Fiske 1990, 4).

The model assumes no single meaning that should be elicited from communication. Instead, there are several meanings that interact with the participants' thoughts of real-world objects and with the message, shaping each other during and after communication.

An illustrative example of meaning creation occurred in the film "Sleepless in Seattle" (Ephron 1993), as two characters discussed conventions of dating. One told the other that "tiramisu" is something that ladies might want on a date. The other had no idea what tiramisu was; he did not share that background knowledge. That did not keep him from creating meanings from what the other person said, and it did not mean that their communication had failed, since their aim was something else than just the transfer of information. However, if the character subsequently found out what tiramisu was, he would probably re-interpret that part of their conversation.

The focus in Fiske's model lies in mass communication, not only in language based communication but also in the visual arts. Symbols carry different connotations, and by meaning assignment mechanisms, such as myths, new or extended meanings can be created. Fiske also analyses the use of metaphor (Fiske 1990: 92-95). In addition to their conventional use of expressing something new or unexpected about a subject matter, metaphors can also be used when a new concept is introduced by referring to more familiar concepts. In software, such user interface metaphors are widely used. For example, electronic "files" are stored in "folders" rather than in "trees". Similarly, files are "opened" rather than "decoded" or

"displayed", even though "decoding" would in fact be a closer representation of what actually happens inside the computer. If a user interface metaphor is followed consistently, it helps the user to identify the concepts and orientate himself in the application. However, if the metaphor is too true-to-life and therefore restrictive, it may prevent the user from using all features of the software. An often-used example is word-processing software: if users think that the program is "like a typewriter", they will not easily learn to leave out carriage return marks from the ends of rows.[11]

According to Fiske (1990: 2), this approach aims to explore how people interact with and interpret messages. The sender's meanings are kept in the background, but the "text" or message or act of communication (verbal or non-verbal) is assigned more importance. Fiske's suggestion that meaning creation mechanisms are similar with all participants of communication seems valid. The meanings may be implicit or explicit, and they remain subject to reinterpretation. Efficient communication presumes interaction between people, messages and images of the world for creating meanings. Communication does not have a start or an end, so there cannot actually be an instance of unsuccessful communication.

Functional elements of communication

The previous models have discussed personal and contextual factors that affect the interpretation of messages. Fiske's model applies a semiotic point of view, in which meanings

11. While metaphor is a widely used strategy for the creation of meaning in software, other methods that Fiske discusses seem more problematic to documentation. These include symbolism, myths and metonymy (which means using a single item for representing a more complex situation). It seems that documentation, as a matter-of-fact genre, should avoid multiple meanings, ambiguities and open interpretations which are typical for the other methods. Documentation and user interface design do make use of symbols as interface elements, but the symbols should be very neutral in order not to create problems for localisation. Their use and interpretation should not become too creative or include extended meanings. However, even if they are interesting, questions of signification are not in the direct focus of this study; see Kaihu and Rouvi (1999) for an extensive analysis.

are achieved through interaction between messages, referents and people. Another point of view is to look at how meanings are conveyed in language.

A model for the functions of language was developed by Jakobson in 1960 (see, for example, Fiske 1990: 35-37 or Levinson 1983: 41-42). He defined the following functions in language:

- Emotive function: the relationship of the message with its sender
- Referential: references in the message to the context or ”reality”
- Poetic: the aesthetic aspect of the message itself
- Phatic: keeping the channel of communication open
- Metalingual: identifying the code that is being used
- Conative: the effect of the message on the receiver.

The success of communication depends on the function. The functions are used throughout any communication. In documentation, the emotive (or expressive) function would be minimised. Perhaps tips and hints to the users might be considered to involve a personal touch of the writer rather than an organisational, impersonal message. If the documentation is created by a third party, the manuals probably contain more of the writer's personal assessments and opinions, thus using the emotive function. Marketing text contains adjectives that express commitment to the program and aim to use the emotive function to the full. The amount of “marketing hype” in documentation varies from culture to culture, and seems not to be preferred, for example, in Finnish documents, while in American ones it is quite acceptable (Meriläinen, 1993).

Referential communication functions in documentation are present, for example, in the names of commands, and in the concepts and descriptions of the program elements. References are also made to a wider context, for example, in the names and organisation

structures shown as examples in pictures (screen shots) of program screens. They contain hints about the type of users and work environments that the program is intended for.

Poetic functions are not generally considered relevant in documentation, even if some playful use of language may appear, for instance, in the names used in examples. The poetic function is often intentionally avoided in order not to break away from genre conventions. Documentation is, after all, supposed to be informative, dry, perhaps even boring; at the least, very matter-of-fact. Any other tone could confuse the reader and thus cause noise in the message.

Phatic functions are present in documentation in elements such as titles, cross references and hyperlinks. Their meaning is to facilitate the locating of information. In this way, the user is offered continuation in order to persuade him to keep the particular communication channel open instead of throwing away the manual or closing the help file.

Metalingual functions appear, among others, in glossaries and definitions of computer terminology, as well as in sections about notational conventions. In software documentation, words like "window" receive their primary meaning from the field of computer programs rather than from the construction business. Also the physical and linguistic genre markers have a metalingual function; the user should be able to tell that the document is an instruction booklet or an online help file. These are all metalingual functions but they work on separate levels.

The conative functions are realised at their most explicit as step lists, which, often in the imperative form, tell the users what they must do in order to perform a certain task. Shannon and Weaver's model concentrated on the conative function of the message as its primary meaning, and it seems to be the most important one if the writer wishes to make the user complete a given task.

As Levinson (1983: 42) sums up Jakobson's model, "language is being used to convey more than the propositional content of what is said." In Jakobson's model, success and efficiency of communication follow from the fact that language can convey several layers of meaning at the same time. These layers are present simultaneously, and the meanings they convey may be in unison or in conflict.

3.2.3 Intended meaning

Language and communication are not only means of transferring information, they are also means of action. People can do things and perform actions by communicating, and people can make things happen by communicating. The producer of the message is assumed to have predefined intentions that may be realised as actions through communication. The effects of communication can thus be seen in the actions. Efficient communication conveys the preferred actions to participants, either for carrying them out or for acknowledgment. Successful communication produces an outcome of the preferred actions by communication participants.

Speech acts

Austin developed in the 1960s a theory of how people 'do' something just by saying it. He called these special kind of verbs "performatives" (see, for example, Schiffrin 1994: 50; Levinson 1983: 229-231). For example, with certain performatives, contracts can be made binding; "I pronounce you man and wife" said by a person with a certain type of authority means that the marriage is legal. If said by someone not possessing that authority, the performative is just another word and does not perform the action. People who have the same culture and language recognise and share the knowledge of these performatives; they are conventions in that culture.

Austin also defined three types of speech acts: locutionary, illocutionary and perlocutionary acts (Levinson 1983: 236). The locutionary act takes place when someone is uttering words; the illocutionary act means the form of the utterance: for example, the speaker is making a statement or an assessment, posing a question, or giving an order. The perlocutionary act means the effect that the utterance is supposed to have on the hearer (for example, confirm the statement, answer the question, perform an action). The locutionary act could be said to concentrate on the "channel" characteristics of communication, whereas illocutionary and perlocutionary acts carry the speaker's meanings, illocutionary act being the surface meaning and perlocutionary act his intended (indirect) meaning. The theory relies heavily on the idea of predefined speaker's meanings as well as with truth values of utterances, and it separates meaning (perlocution) from the message (illocution).

Meanings depend on the context rather than on the message's face value. For example, the message "It's rather cold in here" has different meanings as an answer to different questions. If the question was "What's the weather like?" the answer is a straight-forward cooperative statement of the facts. If the question was "Do you mind if I open the window?" the same message can be interpreted as a polite refusal "do not open the window." The speaker and hearer use similar methods for determining the illocutions and perlocutions, and thus they are able to produce similar expected interpretations.[12]

Action-centered software design is based on the idea that by studying users' actions and their vocabulary for actions, items and processes in the domain, software designers can create extremely useful programs that directly reflect and support the users' tasks in that domain. Speech acts are therefore one important source for the designers; their goal is to "connect the linguistic structure of the domain to the software structures that will support the patterns [of

12. This is like encoding and decoding meanings (instead of or in addition to encoding the messages); there is an encoded meaning inside the expressed (encoded) message. Context and background knowledge enable the participants to interpret the codes.

users' actions]" (Denning and Dargan 1996: 116). Finding out the action structures helps to create useful and usable programs that adapt to the user rather than make the user adapt to the program; therefore they do not require much learning or training. Denning and Dargan criticise the more traditional design which is based on gathering technical requirements and transforming them into design specifications and program features. They claim that the traditional design loses the connection between users' actions and software functions (Denning and Dargan 1996: 107). This disconnection makes the program's commands unintelligible jargon to the users and requires them to learn new action patterns and speech acts.

In software, commands on the user interfaces may be interpreted as performatives. For example, in speech-driven interfaces the user may speak out action words which bring about changes, for example, "Delete this file". In menu-driven interfaces the user has to click the action word (performative) rather than speak it, but the effect is the same. And of course, if the user does not have sufficient user rights or if he does not know the correct formulation of the performative (especially in character-based interfaces, in which the user has to type in the command), actions are not performed.

Much too often there are counterintuitive action words in software, and therefore software documentation has to make sure that the performatives are understood correctly. The command name (performative) cannot always be used in a straight-forward manner to refer to the action: there is a difference between what actions the user has to perform (for example, which commands he should click) and what the intended effects are (what the outcome or "real action" is). For example, it is better to say "To view the file, click the File Open command" than "Click the file open" or "Open the file", since the latter ones refer to the intended effect instead of spelling out the user's required actions. Similarly, the software may use performatives such as "copy and paste" instead of a user's intended action which he might consider to be "repeat". Sometimes the performative is placed at the end of a longer action path, for example,

"to mark the message important, click the Message menu, Message Properties command and select Important from the Type drop-down list."

The efficiency of speech acts in communication depends on whether the speaker's intention (perlocution) is recognised. The effectiveness depends on whether the intended action is performed and whether the performer has the required authority to carry out the action, provided that the locution is well-formed.

Rhetoric model

The rhetoric model of communication asserts that all texts can be adapted to their audience so that the intended meanings or values can be transferred efficiently:

> Rhetoric, or the concept of organizing words, sentences, and paragraphs in a particular way in order to achieve a particular end affects how credible an audience will perceive a given document, how seriously they will consider a particular message, and how they will interpret that message.
> (St. Amant 1999: 297)

The basic assumption is that if the message is designed appropriately to the audience, the users can understand and accept its contents more readily, and in consequence change their actions, attitudes or opinions. The producer of the message has to learn about the background knowledge, social and cultural context, and values and expectations of the audience in order to be able to anticipate the audience's reactions and create a message that transmits the intended meanings. Rhetoricians emphasise an ethical use of language and keep a distance from counterproductive manipulation attempts (Schriver 1997: 58).

Audience analysis is a central task for rhetoricians. By analysing their audiences carefully, writers are able to find out, among other things, the users' tasks, expectations and tastes or preferences. These may differ from culture to culture, and for example the same manual in English may not appeal to audiences in the United States and in Finland since they expect

different kinds of approach and language to be used in a professionally credible manual (Meriläinen 1993, St. Amant 1999).

Audience analysis should preferably mean literally talking to the users and asking their opinions. Schriver presents a study in which teenagers were asked what they thought about a number of drug education brochures (Schriver 1997: 167-203)[13]. The results showed that the intended audience was interpreting not only the subject matter itself but also clues to the writers behind the text. The audience evaluated the implied values and beliefs of the writers about the subject matter and audience, and since they generally did not share the implied writers' beliefs, the text did not work for them. This resembles the question about the popularity of third party manuals; their writers may be considered more credible and their attitudes may seem easier to identify with, which makes the manuals seem more accurate and useful.

The aim of the rhetoric approach in software documentation is to be able to help the users use the product as easily as possible. This requires on one hand extensive audience and task analyses, and on the other hand decisions about the form and medium of documentation. Rhetoricians emphasise the choice of a correct means (channel) to transfer the message. Thus, they may want to use animation, voice and picture instead of written text for certain messages and audiences. Communication is not necessarily only textual, and the ultimate goal is to find any means that will "speak" to the audience:

> Sometimes that means writing an instruction manual; sometimes I create instructions that you can use electronically on the computer; and sometimes I might create a video -- it depends on the situation.
> (Houser 2000: 4)

13. This is actually seen as an example of ethical persuasion or manipulation; as Schriver asserts, it is "a context in which good writing and visual design have the potential to make an important difference." (Schriver 1997: 167)

The intended meaning to be transferred in software documentation includes empowering the users, or making them able to help themselves with using the software. Nonethical goals are not considered professional.

The rhetoric approach involves "encoding" the message in such a way that the audience may "decode" the intended meaning from it as easily as possible. This does not mean that the message should be explicit or straight-forward; on the contrary, it may require the use of irony or other figures of speech, depending on the audience, culture and purpose of the text. The rhetoric model questions the premise of technology and science being value-free and objective, which creates an illusion that the language about them is and should be value-free and objective as well. Rhetoricians have showed that users benefit greatly from audience analyses that results in carefully crafted instructions, taking into account users' tasks, context and their background knowledge.

Effective communication in the rhetoric tradition means that the intended meanings and propositions are transferred to the audience. Efficient communication means that this is performed as painlessly to the audience as possible, considering the audience's background knowledge, expectations and values, their cultural, social and task contexts, and their skills and abilities.

Analysing speaker's intentions

The interpretation of clues about the originator of the text can be structured and analytical, as in Lasswell's mass communication analysis (Fiske 1990: 30-31, Underwood 2000). Lasswell's model includes five questions not unlike the "5W" of Burke:

- Who?
- Says what?

- In what channel?
- To whom?
- With what effect?

The original purpose of the model was to critically analyse mass media and propaganda, but it has similarities also with other communication models. The five main factors in the model are the producer of message, the message, the channel or medium, the recipient of the message, and its effects. The effects are an additional factor to Shannon and Weaver's model. In comparison to the rhetoric model, Lasswell's model emphasises more the role of the producer of the message, but also elements such as the message itself, the medium and the audience are seen as central factors of communication. According to the rhetoric model, the intended effects or intentions are predefined and should support the audience's own goals, but in Lasswell's model they are subject to careful analysis and even criticism.

Lasswell's model aims to show how communication shapes our views of the world. For any piece of news, for example, one should think of the political and economic liaisons behind the scene that affect the way that the news is being presented. How is the message producer involved in the issue? What image of the world is conveyed by the selection of items to be communicated? Is the medium accessible and appropriate for communicating the selected information? What is the role given to the audience and what are the values and attitudes implied in that role? What are the meanings that the producer of the message intends to transmit, and does the audience accept or reject those meanings?

The very same questions can be asked about software documentation. This again brings up the question about third party documents and how (or if) they differ from official documentation. For example, if the Microsoft Press publishes a manual which is not a part of the official documentation set, how does that affect the users' expectations about the manual? Users may think that since it is published by the Microsoft Press, it is probably very accurate

and complete, and that since it is not a part of official documentation, it does not need to present an overly favourable image of the product, which should make the manual truthful and useful. Thus knowledge of the originator can create expectations about the message.

3.2.4 Goal-oriented meaning

Communication can be seen as purposeful action designed and defined by the participants' goals. It is an activity which participants take part in and which they continuously interpret in terms of their aims, which may have nothing to do with the exchange of information. The aims or goals may be, for example, sharing of feelings or having a task done. Contributions in the communication provide stimuli for the participants, and the contributions are interpreted in regard to and through the individual goals that participants have in mind. The context and the goals determine what kind of meanings are elicited, and predetermined or external meanings are of minor importance.

The efficiency of communication depends on the effort that participants must invest in the communication for accomplishing their goals. Communication failure means that the aims could not be achieved, while successful communication ensures accomplishing the goals.

Interaction model

An essential feature of communication is that it is action which is aimed at fulfilling one's aims. Speech act theory suggested that actions can be performed in or by speech, but Haslett applies action theory to communication. According to that, people plan their moves in accordance to their goals (Haslett 1987: 102). In other words, communication is as purposeful action as anything else people do. Communication goes on until the participants have achieved their goals. If a communicative goal is not met one way, another approach (or figure of speech) can be applied. The distribution and management of turns is a joint operation, and in order to

enable efficiency, relevance is the most important single guideline in communication (Haslett 1987: 16, 133). The view is shared, among others, by Sperber and Wilson (1986).

Haslett defines variables that affect interaction before, during and after the actual communication event (1987: 117-125). Participants have a knowledge base of their real-world experiences, cultural values and norms, interactional practices and routines, and language use. This knowledge base along with their goals or aims influences the expectations they have of the interaction. During interaction, the variables are activated. Participants' interaction skills, attitudes and motivation are additional factors in the communication, as are the social and physical settings of the interaction. During interaction, participants constantly react to each other's contributions:

> In order to be effective communicators, participants must continually adapt their utterances to one another and to the context in which their interaction takes place. Interactants plan their actions on the basis of anticipated effects, and judge them in terms of effort, face, and other considerations.
> (Haslett 1987: 145)

After each turn in the communication event, previous text or dialogue is considered, as well as the social relationships and likings that participants have towards one another. These again affect each participant's expectations of the next turn. The experiences from previous encounters accumulate and are activated at each new communicative event.

In their interaction with software, users bring into the situation their expectations and experiences of the application and of other applications, together with their goals. During software use, they apply their computer skills, and their opinions about the software are confirmed or revised with new experiences of using the program. Finally the users evaluate whether and how easily they got their tasks done. Documentation use is a part of this interaction, and follows the same outline. The users bring their expectations and experiences of manuals and online helps to the interaction and start from that context.

Communication reflects motivation and attitudes based on participants' goals and their previous communication experiences. As Haslett puts it (1987, 124): "over time, we may build up an adverse attitude toward someone because past encounters have frequently been painful". Thus, if users have had unsuccessful experiences using documentation, for example, if they did not find what they were looking for or understand what the instruction said, they were perhaps unable to complete their task or just felt silly, which would not be a great motivation for trying to use documentation again. The users' general world knowledge or computer subculture knowledge may also include assessments such as, for example, that documentation is considered useless, or that only the most novice or stupid users (or, the other way round, only technology lovers, geeks and nerds) read documentation. If the user cannot identify himself or herself as part of a documentation-using group, he may decide to find other means to overcome problems with the software, for example, by asking for training instead. The aims of documentation (or of technical writers) include focusing on users' tasks and avoiding ambiguous interpretations. At the same time, documentation shares the openness to reinterpretations that all contributions in the interaction have.

Inefficiency of documentation can result in misunderstandings that hinder task completion or cause errors; inefficiency also consumes time and may make the user stop using documentation altogether since it is not helping him. Relevance is a key issue in communication efficiency. The question of relevance may well be the most difficult communication problem in documentation: how can the goals and contexts of user and documentation be synchronised in individual topics? According to Haslett, changing the context may require face-to-face interaction (1987: 103); this would be quite a challenge to documentation and might be interesting to look further into in another study.

A shared context has a lesser role in the interaction model than in other communication models (Schiffrin 1994: 401). This is because a mutual understanding of the speaker's

intentions is not required; each participant is free to make his own interpretation of the communication from his perspective. The context is meaningful only in relation to the participants' aims and goals. In other words, each participant's personal context is relevant to him only. Communication events are highly dependent on the individual participants' aims and their contexts, not on one mutually perceived context.

Cooperation model

The cooperation model of communication is based on the fact that it is useful for participants to be cooperative. They have means for achieving their goals, if they follow certain conventions and adjust their contributions accordingly. Grice described a set of principles that underlie any mutually understandable communication. He wanted to find out how meanings can be understood, especially in cases when the meaning is not the same as the (surface) message. Grice set out to describe the strategies that people apply in order to interpret the meanings behind what is actually said, for example in the following exchange:

> A: Smith doesn't seem to have a girlfriend these days.
> B: He has been paying a lot of visits to New York lately.
> (Grice 1975: 51)

Grice uses the term "implicature" to denote ideas that are not conveyed in the actual (spoken) words of the exchange (in speech act terms, in the illocution). In this case, speaker B implies that Smith might have a girlfriend in New York; so the two remarks are not disconnected as they might at first appear. Grice describes the train of thought how a participant of an exchange may deduce conversational implicature:

> He has said that *p*; there is no reason to suppose that he is not observing the maxims, or at least the CP; he could not be doing this unless he thought that *q*; he knows (and knows that I know that he knows) that I can see that the supposition that he thinks that *q* IS required; he has done nothing to stop me from thinking that *q*; he intends me to think, or is at least willing to allow me to

> think, that *q*; and so he has implicated that *q*.
> (Grice 1975: 50)

Grice concentrates on the hearer interpreting the speaker's implicature, but of course the same interpretations are made by the speaker as he creates the implicature.

Grice points out that conversation is usually a cooperative effort, in which the participants have the mutual goal of continuing the communication until agreed otherwise. Some moves in the course of conversation are considered unsuitable by all participants, and therefore those moves should not occur. Grice calls the mutual effort the Cooperative Principle (CP), and defines it as a requirement for all turns in the exchange:

> Make your conversational contribution such as is required, at the stage at which it occurs, by the accepted purpose or direction of the talk exchange in which you are engaged. (Grice 1975: 45)

The CP can be followed by observing a set of rules, maxims, that control the quantity, quality, relation and manner of each contribution. Participants are assumed to be rational in their choice of topic and the amount and type of information that they contribute to the interaction. The rationality is measured against the goals of each participant; therefore the CP does not require absolute conciseness, being matter-of-fact or sparing in the contributions, but rather, adjusting one's contributions to one's aims. Maxims are discussed in detail in section 3.3. It should be noted that the CP, even though it is usually mutually followed, is not an explicit or conscious rule. As people communicate, they want to be effective. The CP "rule" follows from an obvious requirement: unless you are willing to communicate in an effective manner, you cannot achieve your communicative aims because the other participants will not be able or willing to recognise those aims. Effective communication means that participants achieve their goals; efficient communication means that the goals can be achieved with as little effort as possible; for example, by creating and interpreting implicatures.

3.2.5 Efficiency and success of communication

The communication theories have certain assumptions in common, and differ in their point of view or focus. Below in Table 3 is a list of some of the similarities and differences that the four types of communication theory have.

TABLE 3. Differences and similarities in communication theories.

Surface meaning	**Acquired meaning**	Communication is a means (a tool with several functions)
Intended meaning	**Goal-oriented meaning**	Communication is an end in itself (action)
Communication flow is linear	Communication flow is non-linear	
Messages are decoded for meanings	Messages are interpreted for meanings	
Meanings are predefined by message producer	Meanings are negotiable in terms of participants' goals	
Message context is immediate	Message context is extended	
Focus is on message producer	Focus is on recipient	

On the whole it can be said that the surface meaning and intended meaning theories are focused on how the producer of the message encodes the meaning in the message, and how it can be interpreted or encoded by the recipient of the message. The direction of communication goes one way at a time, from producer to recipient. On the other hand, the theories of acquired and goal-oriented meaning emphasise the recipient's individual interpretation of messages, and the larger context of social, cultural and individual norms, values, expectations and skills that affect the interpretation of messages. These two theory types assert that there are several layers of meaning, rather than a single encoded meaning that should be resolved. Communication is always a part of a network of other communication events.

The theories of surface meaning and intended meaning differ also in directness. In theories of surface meaning, the meaning is present in the message, but in theories of intended

meaning, the meaning may be deducible from the context rather than from the message itself. The theories of acquired meaning and goal-oriented meaning differ as to the activeness of the hearer. Acquired meaning theories acknowledge several levels of meaning, some of which may be noted and interpreted by the hearer. Goal-oriented meaning theories describe the participants as active makers of meaning. They actively interpret everything that seems relevant and process it from the point of view of their goals.

Another way to look at the four theory types is through the role they assign to communication. Theories of surface meaning and acquired meaning see communication more as a tool for accomplishing something than as action in itself. In other words, communication enables speakers to make the hearers perform certain tasks that lead the speaker nearer to his goal. Meanings are transferred in order to affect the recipients. In contrast, the theories of intended and goal-oriented meaning are based on the idea that communication is action, and that people optimise their communicative behaviour just like any other kind of action. This means that while communication is not necessarily a goal for interaction (although it may be that), communication is seen as purposeful action towards a goal. In these theories, communication is not a means for making others act or making hearers perform actions for the speaker; instead, communication is the action that can achieve a goal.

These communication theories have different assessments of the efficiency and effectiveness of communication. Efficiency means the cost-benefit ratio; most often it is the effort put into communicating compared with the the ease of understanding. The efficiency can be improved with better methods for communication. Effectiveness means how successful communication is, or how well communication satisfies its purpose. Below in Table 4 is a description of the four types of theory and their approaches to the questions of efficiency and effectiveness.

TABLE 4. Efficiency and success in communication theories.

	Surface meaning	**Acquired meaning**	**Intended meaning**	**Goal-oriented meaning**
Basic concepts:	Source, channel, destination Encoding, decoding Noise	Context, culture, intertextuality Functions of language	Intentions Speech acts Audience	Goals, actions Openness to reinterpretation
Efficiency:	Lack of noise; channel characteristics To include all required information in the message	No mis-understandings that depend on background or contextual knowledge (e.g. cultural norms)	Authority and truth conditions are sufficient Well-formedness Adapting to the audience	Observing shared rules for purposeful, goal-oriented action
Success or effective-ness:	The meaning is transferred accurately	Meanings find a place in an intertextual network	Communication produces a desired effect on recipients	Participants achieve their goals
Issues concerning document-ation:	Encoding to the "user's language" Feedback loop Check lists for content	Localisation, internationalis-ation Computer literacy and language competence Genre of computer documentation Several levels of interpretation Conceptual gaps	Commands as performatives; terminology Audience analysis Ethical use of language Implied values, authorial voice Designing for maximally effective interpretation	Expectations and previous experiences of documentation and software use Users' goals instead of their tasks Reinterpretation, new meanings, multiple interpretations Relative or individual contexts Rationality and cooperation

For software documentation, all the four definitions of communication success are required. The message should be transferred to users accurately; users should be able to connect it to other relevant texts and meanings; they should be able to perform actions that are needed for certain tasks; and they should be able to complete their work with the help of software.

Schiffrin refers to Grice's model of communication, and to the speech act theory, as "inferential" communication theories, since they concentrate on how the speaker's intentions and encoding can be interpreted and inferenced by hearers (Schiffrin 1994: 397, 405). In my opinion, Grice's theory provides useful tools for understanding the overall creation or

negotiation of meanings in communication. The theory is not restricted to studying predefined speaker's intentions, but describes communication and its logic more broadly. The maxims are descriptive principles of meaning creation, of making sense of communication; they are not prescriptive principles of meaning transfer or encoding.

3.3 Gricean maxims

3.3.1 Definitions

The CP requirements can be fulfilled by observing (at least) four sets of rules: the maxims of quantity, quality, relation, and manner. Grice defined the maxims in the following manner (1975: 45-46):

1. Quantity

   Make your contribution as informative as is required (for the current purposes of the exchange). Do not make your contribution more informative than is required.

2. Quality: Try to make your contribution one that is true.

   Do not say what you believe to be false. Do not say that for which you lack adequate evidence.

3. Relation: Be relevant.

4. Manner: Be perspicuous.

   Avoid obscurity of expression. Avoid ambiguity. Be brief (avoid unnecessary prolixity). Be orderly.

The maxims are defined so that the outcome of the conversation could be as effective as possible. This, like the action theory Haslett applies, is based on the assumption that communicating is rational behaviour and therefore the participants avoid wasting too much

effort at each stage of the conversation. The maxims of relation and quantity can be seen as ensuring the effectiveness of communication: exactly the right amount of relevant information is transferred, while the maxims of quality and manner can be seen as guidelines to efficiency: participants need not be concerned about misunderstandings or worrying about the accuracy of information by default. However, Grice points out that some additional social, aesthetic or moral rules (such as "Be polite") may also observed (Grice 1975: 47).

Grice says that his definition of the maxims is related to a "maximally effective exchange of information" (1975: 47). On the other hand, the purpose of an exchange could just as well be "influencing or directing the actions of others" (Grice 1975: 47). Another purpose could be, for example, displaying solidarity or friendliness to the next person while waiting in the queue, in which case the information content might be very limited. Therefore, another way to describe the purpose of the CP might be that it helps to achieve the aims of the participants as efficiently as possible. These aims may be, for example, displaying friendliness, exchanging information, inventing puns, or something else. The four maxims constitute the foundation for understanding and using implicatures (meanings behind the surface message) in communication.

The logic for participants' observing the maxims is based on general features of cooperative transactions (Grice 1975: 48). First, the participants have a common aim or need for communication, even though their ultimate aims may be conflicting. Further, their contributions are dependent on each other's contributions. Finally, there is usually a mutual understanding about when to end the transaction.

In documentation these features could be realised, for example, in the following manner. The user's immediate aim could be attaching a file to an e-mail. The documentation would typically address that aim in a section about sending mail or handling files. The ultimate aim of the user could be finding a job (by sending his application by e-mail); the ultimate aim of

documentation is to make the user happy with the e-mail program. The interdependency of contributions can be seen in sequential instructions. The documentation includes step-by-step instructions for attaching the file. Step *n* always requires that the previous step (*n-1*) has been performed. The user goes through the steps in the correct order; he cannot skip steps, and often he cannot even understand the steps further on until performing the next step that, for example, opens a new program window. But if the user does not follow the steps, instructions after step 1 are useless. Therefore the contributions (instruction - action) are mutually dependent. It should be noted that the user's contribution need not be verbal. And finally, ending the transaction is made in a perfect understanding: the user stops reading the documentation after he has completed the task, and the instructions end after the described task has been completed. However, if the instructions were to end too soon, for example, without a return step to the main window, it might not seem cooperative to the user; they might be left guessing if the task is actually completed or not.

Grice lists four reasons why a maxim might not be observed. First of all, a participant might (unintentionally) violate a maxim and therefore mislead others. Second, someone may "opt out" or refuse to cooperate. Thirdly, two or more of the maxims may clash and the speaker can only follow one maxim at a time. And last, the speaker may exploit a maxim by deliberately (i.e. it being obvious to other participants of the exchange) not observing them. Grice calls the last reason "flouting" the maxims, and states that it characteristically generates conversational implicature (1975: 49).

The example above about Mr. Smith's New York girlfriend is an example of observing the maxim of relation. Even if the exchange parts seem unconnected, A has no reason to believe that B might be violating the maxims; instead, B should be observing the maxim of relation and therefore his statement about New York must be interpreted in that sense (Grice 1975: 51). Implicature depends on what the speaker actually said; what the CP and the maxims

require; what the context is and what background knowledge is available; and whether both participants share the same assumptions (1975: 50).

Maxims are flouted, for example, in metaphors: files are not stored in real folders but in some kind of electromagnetic devices. Irony and hyperbole are other examples of statements that are not strictly speaking true, and in which the maxim of quality is flouted (Grice 1975: 53). The maxim of relation is flouted if someone makes a complete change of subject (1975: 54). The maxim of quantity can be flouted in irony as well, and the maxim of manner is flouted, for example, in Lewis Carroll's beautifully obscure nonsense verse "Jabberwocky".

Documentation is generally not expected to flout the maxims. Instead, people easily regard instances of missing information or unclear instructions as cases of the documentation-participant wilfully opting out, and consider documentation (or its producers) uncooperative, impolite or unfriendly. If information is not available in documentation, or if the user's questions lead to a loop of unhelpful links, documentation can be interpreted as ending the communication before the user-participant has agreed to do so. This impoliteness may mean that the user loses face; this affects his expectations about the next communication event, and he may assume a similarly hostile attitude.

Developments to Grice's theory have later been introduced by several scholars. Sperber and Wilson suggested that just the maxim of relevance is enough to describe communication (Sperber and Wilson 1986: 46, 50). Relevance is said to produce and result in the efficiency of communicative transactions. Nyyssönen states that for relevant texts, sufficient understanding is achieved with the minimum of processing effort: "Such a text has optimal relevance. It also has optimal accessibility" (1993: 25). People are lazy by nature, or like to save their effort for something else, and therefore optimise their communicative performance by being relevant.

Additional communication rules have also been suggested. Leech (1983) discusses the politeness principle and the irony principle, as well as several other principles such as

processibility, clarity, economy and expressivity principles, that make communication interesting and varied enough to be participated in. Haslett (1987: 51) reports on other principles such as morality and charity maxims. The politeness maxim has been most widely studied as another norm of communication.

3.3.2 Cognitive metaprinciples

Riley and Parker (1998) argue that Gricean maxims of conversation are parallel to the Gestalt principles of visual perception, and that both in fact describe general cognitive principles.

Gestalt psychology was developed early in the 20th century. It accounts for perception with a set of principles (those of continuity, figure/ground, closure and constancy) that govern our understanding of visual elements. When we look at images, we tend to perceive figures in similar ways. If part of an element cannot be seen in the figure, we still expect the element to continue similarly the way it started out; we expect figures to display a prominent object for which everything else forms a background; we interpret parts of objects as whole; and we expect the objects to stay the same even if the background changes (Riley and Parker 1998: 176).

From these Gestalt principles and Gricean maxims, the following metaprinciples can be formed (Riley and Parker 1998: 184):

1. Cohesion

   Perceiver interprets stimuli in the most obvious way requiring the least effort

   (Continuity principle and maxim of relation)

2. Clarity

   Perceiver imposes a single interpretation on a stimulus

   (Figure/ground principle and maxim of manner)

3. Completeness

   Perceiver interprets the stimulus as whole

   (Closure principle and maxim of quantity)

4. Correctness

   Perceiver interprets the stimulus at face value

   (Constancy principle and maxim of quality)

Combining the sets of principles into general cognitive strategies displays similarities in understanding visual stimuli and language. It even suggests that there might be universal deep structures that govern all of our cognitive activity.

4 Empirical study

4.1 Hypothesis

In studying communication theories I assumed that they can be applied to documentation as well as a specific type of communication event. Since Gricean maxims provided a well-defined strategy for successful communication, I assumed that they could describe documentation use as well. I therefore made a hypothesis that Gricean maxims can explain communication efficiency and success in documentation use; more specifically, the maxims could be used to describe the causes of possible problems in documentation use.

In order to test this hypothesis, I decided to carry out usability tests on documentation. The purpose of the tests was to observe users as they performed predefined tasks with software and communicated with documentation for achieving their goals. By analysing the users' behaviour and using the Gricean maxims to explain some or any of the potential communication problems resulting in unsuccessful task accomplishment, I would be able to identify instances in which documentation failed to observe the maxims and therefore did not communicate successfully. As these instances were identified, it should be easier to correct the communication problems and thus make documentation more usable and useful. I also expected to observe where the documentation worked well and was accurate and useful, and interpret those instances with the help of Gricean maxims too. However, I did not include making and verifying corrections as part of my test plan; my main goal was to find out whether Gricean maxims would work at all in identifying communication problems. I also restricted the scope of the tests to inspecting specific pieces and uses of documentation, and not generalising those to documentation use in general.

I expected to apply the maxims into documentation in the following manner:

- Maxim of quantity: the needed information exists in the documentation.

- Maxim of quality: there are no errors in the information.
- Maxim of relation: user finds the exact information that he needs, and does not mistake some other piece of information as the one that he needs.
- Maxim of manner: the information is easy to understand and does not require extensive processing or interpreting.

These applications seemed intuitive as a starting point and able to ensure successful communication and accomplishing the users' tasks.

4.2 Test design

4.2.1 Tested applications and documentation

I decided to use three different software applications and their documentation for the usability test. This way I hoped to find more issues than with a single program as well as avoid one-sided observations. The applications were TeamWARE Office, developed by Fujitsu, Microsoft Exchange and Lotus Notes by IBM[14]. They are so called groupware applications, consisting of programs for electronic mail, electronic calendars and other information sharing systems. The purpose for choosing these applications was that I wanted to have the same tasks done with all the programs, and these had been designed for similar uses. Moreover, I had access to all three of them, which made it possible for me to arrange the tests.[15]

The three groupware applications are competing products and they are aimed for the business market. At the time of the tests (in spring 1997), Notes was the market leader, while TWO had a strong position especially in Finland and Japan. MSE was a small new competitor

14. Hereafter referred to as TWO, MSE, and Notes respectively.
15. I am grateful to TeamWARE Group and Trantex Oy for the opportunity to carry out the tests.

that had been gaining foothold due to Microsoft's favourable bundling strategy. Nowadays, MSE is almost as popular as Notes, which remains the market leader.

All three applications work at least in the Windows operating system. Because they are competing products, documentation is one aspect that can add (or reduce) their competitive advantage; other aspects include price, performance, customisability and ease of use. All applications were English language versions and had graphical user interfaces. The documentation consisted of online help files as well as printed documents:

- Lotus Notes Help
- Lotus Notes Beginner's Guide (printed document)
- TeamWARE Office Help
- TeamWARE Office User's Guide (printed document)
- Microsoft Exchange Help

Unfortunately I had no printed documentation of MSE available.

Of the three applications, MSE and TWO were most alike in that they both shared a tree hierarchy based user interface metaphor with folder structures. Their help files were both realised as Windows 95 helps, while the Notes help file used Notes' own database implementation and a slightly more complicated tree structure.

Lotus Notes

Lotus Notes looks similar and works in the same manner in several operating systems. It is aimed especially at large organisations and requires quite a lot of configuring before it can be taken into full-scale use. Its design is based on databases that hold all information in the system; for example, the e-mail and calendar programs are realised as database solutions. System administrators can build several different customised views of the same data for

different user groups and purposes. These and any other databases are presented as icons on the Notes main window (which is called "the workspace"). Its versatility, scalability and customisability to different purposes offer more possibilities than MSE or TWO, but on the other hand Notes requires a lot of configuration at system level before it becomes useful. End users will typically need training for using Notes because it contains rather technical terminology and the concepts may be unusual for people who are familiar only with Windows terminology and practices. For example, Notes' design is not hidden from the user, so that the user actually has to learn the word "database" in using the system.

The end user's view to the system does not require (nor allow) much customisation. Notes can have several windows open at a time, either within one frame (the workspace) or independently anywhere on the screen, as chosen by the user. Notes is started from a single icon from the desktop, and the different programs (such as e-mail or calendar) are started by clicking the icons on the workspace. The user interface font size is quite small and the toolbar buttons ("Smarticons" in Notes' terminology) are not very distinctive. Below is a screen shot of the e-mail window in Notes.

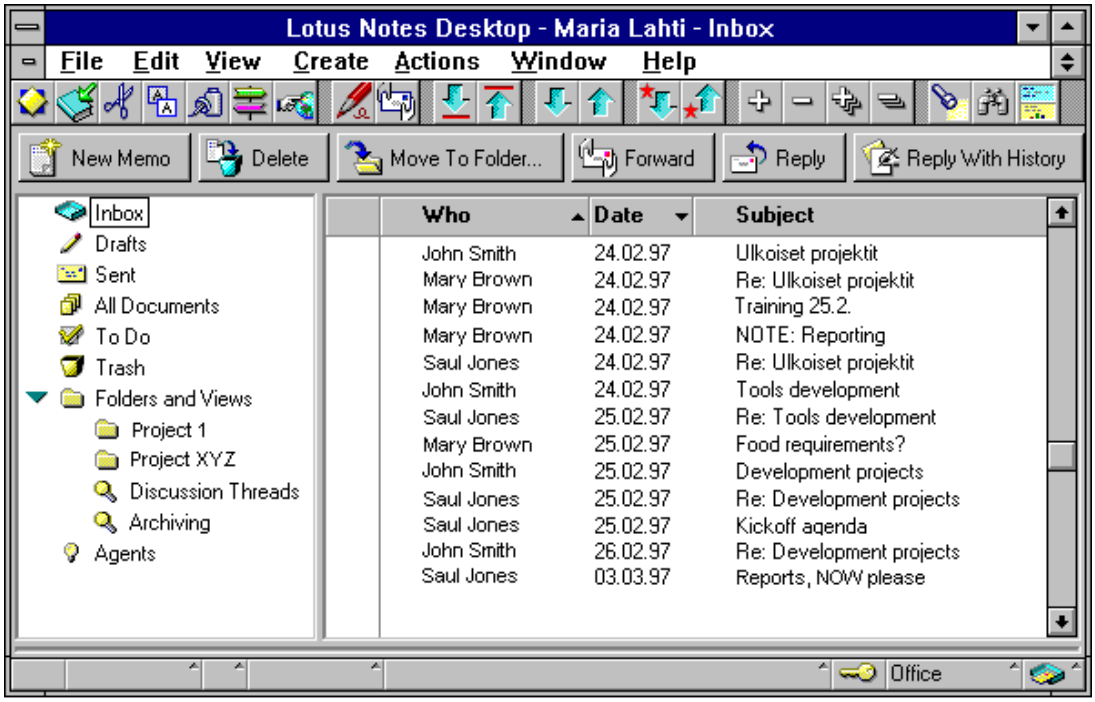

Figure 13. The e-mail window of Lotus Notes.

The tree structure on the left pane of the window contains default folders and the user's own folders containing mail messages. However, the tree structure also includes different views that filter a selection of the same messages onto the screen according to given criteria. Another thing that the tree structure offers is access to creating one's own agents, mini-programs or macros that perform simple tasks for the user. Since all of these are contained in the same tree structure, it is not easy to notice their differences at the first glance.

Notes has an extensive online help database with slightly different functionality from Windows 95 help. Its navigation has a different structure from typical Windows 95 helps, and some of the links are different as well: clicking a link preceded by a triangle opens a short list of topics to the same view, and the user can click another link from the list to open a new topic.

TeamWARE Office

TeamWARE Office works in the Windows operating systems (like MSE). TWO is a collection of programs, such as Mail, Calendar, Library and Forum, that can be used individually or together to distribute information on individual and organisational levels. TWO does not require quite as much configuration at system level as Notes does. Its user interface is less cluttered than Notes' and it offers more customisation possibilities to end users. A screen shot of the TeamWARE Mail main window is presented below.

Figure 14. TeamWARE Mail.

The folder structure only includes message storages; views and other functionality can be accessed through the menus, and message list sorting is possible by clicking the field headings on the right pane of the window. Overall, TWO uses Windows-like terminology and functionality.

Each program of TWO has its own icon on the Windows desktop and a separate main window, even though the programs can be accessed and used also from inside each other. A common framework to TWO is handled by a TeamWARE Session window which is normally minimised at the bottom of the screen.

Microsoft Exchange

MSE is aimed at smaller organisations that do not invest heavily in customising their groupware systems. MSE contained only an electronic mail program at the time of the tests. MSE is integrated into the local area network, so that all user accounts in the network automatically receive e-mail user rights in the system. It is thus quite easily maintainable, synergic with managing the company internal network, and it does not require heavy configuration at system level.

The MSE user interface is quite easy to use. However, Windows terminology and standards were not fully used in the version that I tested, since the application had only recently been acquired and renamed by Microsoft. The differences to Microsoft's standards were most obvious in the online help, the layout and structure of which did not follow typical well-defined Windows 95 help systems. Not unexpectedly, my tests on the available MSE help identified problems that would have been avoided with a more standardised help implementation.

A picture of the MSE main window is displayed below.

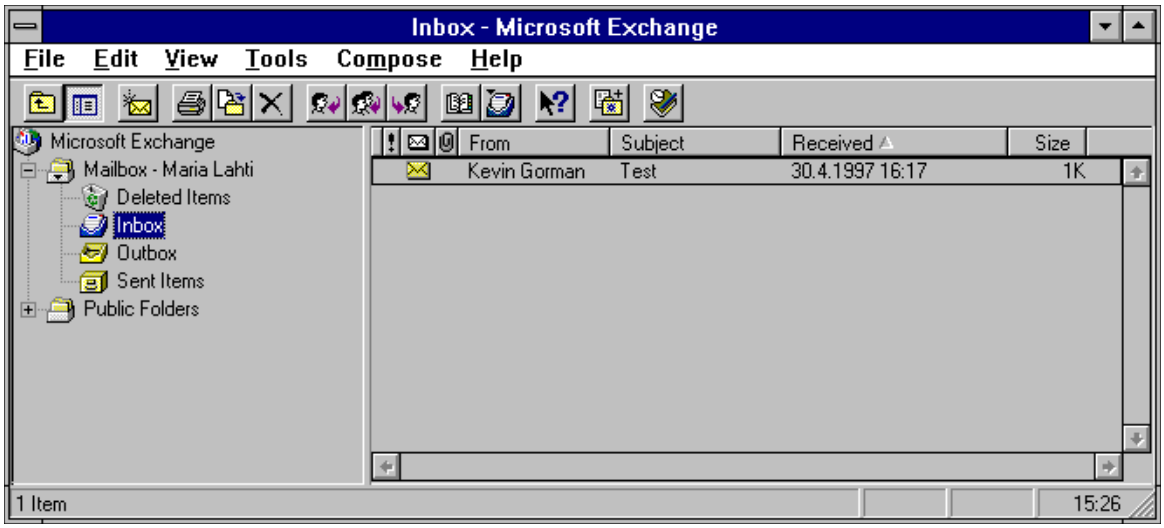

Figure 15. Microsoft Exchange main window.

The folder structure on the left is nested and allows navigation into organisation-level (public) items as well as into user's personal messages. The order of the mail folders is interesting;

Deleted Items come first by default. The message list pane on the right contains similar sorting possibilities (by sender, subject, and so on) as TWO, and the icons in the message list headings are quite similar.

4.2.2 Tasks

I made up five tasks that I considered typical with groupware applications and which new users would want to learn quickly right at the beginning if they wanted to use the application efficiently. All of the tasks could be performed with each application, even though they were named with slightly different terms. The tasks mainly had to do with using e-mail. They were:

1. Starting the application and changing one's password.
2. Creating a single mail alias for a group of people.
3. Sending an e-mail message to the group with two attachments and a copy of the message to another person at one go.
4. Forwarding a received e-mail message.
5. Quitting the application and verifying that others cannot access one's mailbox.

The users were to perform each task with the help of documentation so that they should first try to find instructions for the task, and then perform the task according to the instructions. Before and during the first task, the test users were encouraged to explore the application and its documentation. In order to avoid using the terminology particular to one application, the tasks were given in Finnish. Each test user had to think of suitable English terms that would describe the tasks in the documentation.[16]

The test users were given a user name and a password, as well as file and path names and other information required for the tasks. They were free to invent a name for the group alias, as

well as formulate the message freely. The first and last task were added because starting a program and quitting it can sometimes be difficult for the user. Changing the password is the first thing that a new user should do with an e-mail program. The last task was aimed at including a conceptual issue about the meaning of "logging out".

4.2.3 Test users

Since my goal was to verify whether Gricean maxims are a useful tool for assessing the quality of documentation, I did not need to collect statistical data but could use qualitative methods instead. I needed specific instances of problems in the communication with documentation, but I did not need to find out which of the problems occurred most often or with which type of users. Therefore I decided to recruit only five test users: each one would perform the tasks with one application, using one of the documents. This way I had only one test user for MSE and two for both Notes and TWO, but since I did not intend to compare the usability of the applications with each other, I decided that one user for MSE would be sufficient. Usability tests are typically conducted with less than ten test users for two reasons: the analysis phase is very time-consuming and most of the problems can be detected with 3 to 6 users anyway. Thus the benefit of having more test users is lost by the more rapidly growing workload.

There were two criteria for selecting test users: they had to have adequate English skills, and they had to be able to use the computer. Two of the users were to use the printed guides and three the online help files. The test users were found by asking friends and informing them about the purpose of the study. All were native Finnish speakers, two male and three female. On a rough scale, two of the users were advanced computer users, while three were less experienced. The advanced users were aware of how software logic generally functions and

16. As noted earlier, the applications were English language versions. The choice was based on convenience; I had the English versions readily available, while Finnish versions would have been rather difficult to obtain and set up.

how different kinds of programs usually work. The intermediate and novice users were familiar with some applications that they often worked with, but they did not have a lot of experience of other programs or any extensive knowledge of Windows terminology or Windows controls. All had used electronic mail applications before, and two or three had used graphical (Windows based) e-mail. None of the test users had experience of the particular application that they tested.

4.2.4 Procedure

I chose to use the "think-aloud" protocol for my tests because I expected it to yield insight to the users' thinking and because I was already familiar with the method. (For a discussion on the usefulness of the think-aloud method, see Salmi 1998.) In order to make sure that I could check the details and verify my interpretations of the user's actions and comments, I decided to videotape the sessions so that the users' speech and the mouse pointer movements on the computer screen were recorded. I had reserved one one-and-a-half hour tape for each session, but decided not to put a time limit for performing the tasks. In addition to the actual performance with tasks that the think-aloud tests would reveal, I wanted some background information of the test users. I therefore included a written questionnaire about their familiarity with computers (computer literacy), and a conversational interview in which I wanted to find out any additional comments that the user might have about the test, the tasks, application and documentation.

The tests were conducted in the early summer of 1997. Each test session was held separately for each test user at my workplace, with the test user and me present. Sometimes another person was working in the room at the same time. Before the test, I explained that the purpose of the test was to find out how well the documents were written and not evaluate the

user's performance on the tasks. I told the users that the tests were made as a part of my master's thesis and asked if they would mind my making a videotape recording of the test.

I then asked the test users to "think aloud" during the test; in other words, to describe their thoughts, reactions and expectations during the test. The software and the documentation were in English, but the test tasks were given in Finnish, and the test subjects were asked to use Finnish (their mother tongue) in thinking aloud. I explained that my role would mainly be to sit there quietly and observe. I promised to help in translations if they did not understand the language. If there was a problem with using the computer or the software that was not directly related to performing the test task, I gave some advice. I also assisted if the user got stuck and could not solve a problem on his own.

I handed out the tasks to the users on pieces of paper one at a time; they got a new one after concluding the previous one. The users did not know how many tasks they would be performing because I did not want to inflict any time or performance pressure. In fact, two of the test users did not perform task 4 (forwarding a message); one because the time ran out and one because I had not been able to prepare an original message.

Before the test, each user was asked to answer the questionnaire. After finishing the tasks, I interviewed the users about specific problems that I had noted during the test, and about their overall impressions of the test, the application and the document they were using. I did not tape those interviews but took notes instead in order to create a more relaxed atmosphere. After the tests, I thanked the users for participation.

4.3 Analysis

4.3.1 Methods

After the tests, I transcribed from the videotapes the test user's speech, the pointer's movements and user's actions with the application, and my own comments. Usability test

results may generally be assessed according to different criteria, some of which include task success rate, time used or optimality of steps for performing the task. It seemed to me that since the users had to translate the task into English as well as find out a way to work with a new software application, they would need to apply problem-solving strategies. This meant that the users would test their mental models or hypotheses about the problem definition and solutions for it. I did not intend to assess the usefulness of their hypotheses or mental models, so I decided not to use performance time as a deciding factor in my analysis. This also made me reject the optimality of steps (finding the easiest way to performing a task) as a main criterion. I assumed that as long as the user was able to continue working and proceed with the task, they were performing successfully. This required, in my opinion, that the application and documentation would provide sufficient support for the users for creating and adjusting mental models about the tasks and the software. Thus, failures to proceed with a task or instances of frustration would point out problem areas in which the application and the documentation could not communicate successfully with the user.

I assumed that the users worked their way from an initial hypothesis of how to perform a task into mentally dissecting the task (or problem) into subtasks (or solution attempts), for which they needed to find action sequences in order to carry out the original or main task. I therefore based my analysis on interpreting from the users' actions the task and subtasks that they were trying to perform. New subtasks often arose from items or actions on the UI, but mostly they depended on how the test user decided to approach the main task. I listed the tasks and subtasks, noted the method that the user used for completing the subtask and whether the user had succeeded or not with that method. I also tried to assess where the user got the idea for each solution attempt, and noted any comments they had about the problem or documentation, and specific issues that had caught my attention. An example of this analysis is presented below:

- Main task: Start Lotus Notes and change your password.
- Subtask 1: Start Notes.
- Attempt: Double-click the Lotus Notes icon.
- Succeeded: Yes.
- Source of idea: Experience with computer software.
- Comment (by user): Nice pointer character (a running man).

I wrote my analysis in a table and allowed for three levels of (sub)tasks. I arrived at this method of analysis after a few attempts. In my opinion, it presented a fair possibility to isolate specific instances (subtask solution attempts) in software and document use, and allowed interpretation for the background and the reasons for the choices that the user made. The interpretations and assumptions that I made about the users' tasks and subtasks are, of course, best guesses. The users' speech that was recorded on videotape did not prove a rich source of information, since they did not remember to talk aloud all the time. This could depend on two reasons: either because they considered their actions obvious, or because their cognitive load grew too much at some points. This means that they were busy figuring out the problem and possible solutions for it and therefore forgot to keep talking at the same time. I found that the mouse pointer movements alone were quite a good indication for what the users were trying to do. And of course, occasionally the users named the items that aroused their interest or talked about their current problem, which was helpful.

I also noted whether the user had been able to accomplish each task or not. Some of the subtasks were left indeterminate, with no result, often because the user's attention was caught with a new subtask or because the task became irrelevant as the user adopted a new approach to the main task. Sometimes I had to help the user proceed in his subtask; I made a note of what kind of issues were most likely to require help from me, rather than from documentation.

Some of the subtasks I assisted the user with were completed successfully, and some not; I considered my help a source of solution idea for the user rather than a separate solution type.

Next I focused on those subtasks that the user had not been able to accomplish. They reveal problems in program use that the user could not overcome with the help of documentation, in other words, those are the instances where communication has for some reason failed between documentation (or software) and the user. The user's problem solving methods were not supported by the program and its documentation. I categorised the problems to see what areas of software and documentation use were the most problematic.

The four categories I came up with were problems in application functionality or using its controls (including the workings of the help system); problems in using general Windows features that are not features of the actual application (for example, using the x mark on the right-hand upper window corner to close a window); problems with documentation or UI text (for example, trying to cope with misleading, ambiguous or missing information); and finally, problems in the test set-up and task definitions (such as whether contact names for the group alias had already been added in the address book or if they needed to be inserted before creating an alias). I found this categorisation suitable, because it allowed me to account for all problems while focusing on documentation issues. The purpose in separating problems in the use of Windows and in the use of the specific application is that a basic knowledge of how to use Windows is usually assumed from readers of documentation. As the reasoning goes, there is no point in writing Windows documentation again for Microsoft and users should look up Microsoft's documentation if they cannot use Windows. This, of course, is not a very user-friendly approach, because the users can hardly be expected to know what is a Windows feature and what is specific to another application that they are using; and they will assign all problems to the application and expect help from its user interface and documentation.

After identifying unsuccessful subtasks and their problem types, I analysed each case as to which Gricean maxims, if any, had been violated since the users could not reach their aims with the help of documentation only.

4.3.2 General observations

Here are intuitive observations that I made of application and documentation use before and during the tests.

Overall, the test users usually stuck with the method of searching they had first adopted, even when that method did not work. One user tried all kinds of searching methods in the online help, but two others seemed to get lost in hyperspace when they tried several navigational controls in the help. Using the different kinds of controls required so much attention that users seemed to forget or become distracted from their actual task.

Users were generally unwilling to read long topics. Another interesting feature was that users generally expected only one piece of information in each topic. This was seen in that users did not notice the information they were looking for. It could be that users assumed from the topic name that the scope of the topic would be limited, or perhaps they assumed that text before or after a step list would not contain any useful information whatsoever. On the other hand, users did notice the tips and subheadings that were marked clearly. This would suggest that different pieces of information should be marked very distinctly. It seemed as if the users, even though looking for related information, shut it from their view. Their comment quite often was, "if only I had read the topic carefully the first time".

Users looked for words they know from other programs. Sometimes the transference from what they had previously learned made it difficult to understand the terminology used in another application. The English language also caused some difficulties, but it is hard to say how much depended on computer terminology, since that was not clear in Finnish either.

When users grew more desperate and frustrated, they tended to look for single words in topic headings in an attempt to find any information about the subject matter. Sometimes the users were unable to think of search terms and relied on suggestions in the table of contents or keyword index. As expected, all users said they would normally try out the tasks rather than look in the documentation.

Lotus Notes

On the whole, Notes did not seem very easy to use. The purpose and the database architecture of the program was far more complex than what the test users were expecting (which seemed to be a simple e-mail program). Especially navigation was difficult, since the database icons could not easily be interpreted as clickable icons. Terminology was a real problem that neither of the two Notes test users could overcome. For example, Notes uses at least the terms "document", "message" and "memo" for one single thing, and this was as good as impossible to deduce from documentation. Similarly, clicking rules (whether to click once or twice) seemed to vary from case to case, and the test users did not learn the rules during the test but had to test the clicking method each time. Another problem was that sometimes the pointer turned into a "wait" (hourglass) symbol without any apparent reason. This made the users wonder if they had clicked something by mistake and started an action that they had not intended to perform.

Notes had a bug in the help system that made the Help window open as a very small but not minimised window at the bottom of the screen under certain conditions (see Figure 16). This was not easy to notice unless one knew to expect it; instead, the users thought that the help window did not open at all. The window had to be resized in order to display the help, which also seemed rather difficult or at least not obvious to the test users.

Figure 16. The bug with the Lotus Notes help window made documentation difficult to find.

The Notes help system is actually another database in which most application commands are available at the same time. This would have been an easy way to perform the tasks, but it was not noticed by the users, since help was assumed to be a separate system from the actual application. Therefore, instead of performing the task at hand with menus and commands in the help window, the users retraced their steps to close the help window before proceeding with the task.

Notes help uses pictures of open and closed books for navigation, just like Microsoft Help. However, the book icons work differently in Notes; the last clicked book always remains open, whereas in Windows each book can be opened and closed separately. The users clicked the book icons several times in order to close the open book, which could not be done.

Notes help index contained long lists of entries and it was hard to determine by the name whether a topic would be relevant or not. Topic names were listed as first level entries which made the lists very long indeed. The nesting level of index keywords sometimes seemed to be hard to distinguish. However, the actual help topics were short and readable.

Help topics contained several linking methods. Also pop-up definitions were included, marked by a box around the word. Links to other topics were marked with underlining, and expanding sections within the text were marked with triangles. Their clicking rules differed somewhat, and one user did not find out how to follow some of the links. The help also contained a Favourites folder into which the user could have stored topics for easy access. However, users did not take time to explore that feature.

The manual for Notes[17] was printed on thin and rather transparent paper so that text from the other side was showing through. There were lots of pictures and screen shots integrated with step list instructions, up to 3-4 per page. Text and graphics placement varied, and letter spacing varied, too. This all made the layout somewhat restless. There was a table of contents but no index. The manual used very informal language, for example, one heading was: "Using those strange things that you see in documents". It took a moment or two for the test user to understand what was meant by the informal titles. The manual had a cartoon character called Sherman who acted as a guide. The test user liked the character. The manual did not contain all of the information that was needed for the test tasks, so the test user had to look in the help as well. The instructions seemed good and thorough for the tasks that were included, and the screen shots provided excellent references to the application.

TeamWARE Office

The address book of TWO, TeamWARE Directory, seemed rather difficult to use. It displayed an empty window when the user initially opened it, and the syntax for searching for addresses was strictly controlled; for example, just typing the first letters of a name was not allowed; instead, the user had to insert an asterisk (*) to denote missing letters in the search term. This made users perform searches for which they got no results, which caused confusion and made the users wonder if they were in the right place.

The TeamWARE Session window was another source of confusion. It is a minimised icon at the bottom of the screen that appears when a user logs into TWO. It contains information about the session, and it is the only place where the user can change his password; otherwise, a typical user would not do much with it. Neither of the TWO test users was able to locate it without assistance. The documentation only said it would be "on the desktop" which

17. The full title was *Lotus Notes Release 4 Step by Step: A Beginner's Guide to Lotus Notes.*

was not sufficient; besides, the users did not know what the "desktop" actually meant, the open window or the whole screen. Otherwise TWO had helpful cues in the user interface that enabled the users to find to the right places.

TWO help contained very long topics, some of which were quite similar or had only little noticeable change. For example, the address book had two similarly working parts for the individual user's and the organisation's addresses, and therefore it also had two very similar and long help topics that only seemed to vary in the title, which were also very long ("To search for members for a private group in the Organization Directory" and "To search for members for a private group in your Personal Address Book"). When the users opened those topics, they closed them again as quickly as possible; probably because they could not bother to read that long topics and because the topics seemed equally unhelpful for the simple problem that the user had in mind.

The manual for TWO[18] contained a table of contents that spanned six pages, and a keyword index as well as a glossary. The test user, however, only used the table of contents as the main navigation method. There were few pictures, which were mainly screen shots. The manual also contained references to online help keywords, with which users could look for more information, and specific sections for questions and answers. The instructions were generally given as plain text sentences rather than step lists. The manual did contain the required information for performing the tasks, but its test user could not make sense of the instructions at points and chose to look at the online help instead. The manual contained at least one error about how to add user names to the group alias. In addition, the user complained about cross references between sections in the manual: he was forced to jump from one place to another in the middle of a task.

18. *TeamWARE Office User's Guide*

Microsoft Exchange

Since MSE was tightly integrated with the operating system and the user rights in the local network, changing the password required changing one's Windows password to the network. This was not mentioned at all in the help. Otherwise the program seemed easy to use and the user interface provided helpful cues.

MSE help was not quite as easy. Topic titles differed from each other very little; for example, there were topics called "Overview of Sending a Message" and "Overview of Sending Messages". The index contained long lists of topics (up to 20-30) under a single keyword. This made it difficult to locate a topic from the index; and since the index was the preferred method used for navigation by the test user, it caused a lot of frustration. The table of contents was also rather long and not very clearly organised.

The chunking of information into topics in MSE help was not clear. There seemed to be several overview type topics about the same issue, while procedural instructions sometimes seemed impossible to find. Some of the topics were quite long and most used technical terms, such as, for example, "Overview of the Microsoft Exchange Client" in which the term "client" was not explained, and it did not appear in the user interface either.

One topic type included a menu-and-command explanation, which the user considered excellent because that would show right away how to use a command or perform a task. The topics usually contained links to related topics at the top. However, the user did not notice those links at all and did not follow them. Instead, the user tried to find the topics through the index, which was almost impossible because of the multitude of entries.

4.3.3 Results

These are the findings of my analysis in brief. A more descriptive account of how I arrived at these results is given in section 4.3.4.

I calculated rates of problem types and subtask accomplishment, but of course this was no quantitative or statistical research. I was looking for specific instances in which I could apply elements of Grice's theory of communication and assess the usefulness of the method. Therefore, even if I am making generalisations and noting trends, those are just indicators for how the documentation perhaps could be approached in later, more sophisticated studies. The usefulness of the tested documents cannot be determined only on the basis of the tests I made. But for my limited purposes, the data is sufficient.

The numbers of subtasks and their success rates can be seen in Table 5. Users 1 and 2 were using Notes Help and Notes Beginner's Guide respectively; user 3 had MSE help, user 4 had TWO help and user 5 had TWO User's Guide as their primary source of information. Even if each test user was to have only a single means of documentation, both users of printed guides used the online help as well.

TABLE 5. Subtask accomplishment.

	Successful	Indeterminate	Unsuccessful	Total (Success rate)	Time
User 1[a]	100	21	54	175 (57%)	92 min
User 2	76	10	61	147 (52%)	89 min
User 3[b]	44	7	37	88 (50%)	41 min
User 4	49	6	30	85 (58%)	32 min
User 5	58	5	25	88 (66%)	52 min
	327	49	207	583 (56%)	

a. Only three of the user's tasks have been analysed, because the videotape ran out.
b. User 3 did not perform task 4 because I was unable to arrange its prerequisites.

As can be seen from the numbers, Notes users (users 1 and 2) performed far more subtasks than the other users, and took more time overall. If we consider subtask success rate as a potential source of frustration to the user, no great differences are shown in the numbers above. Judging intuitively and subjectively, users 3 and 5 showed the most frustration at certain points. If nothing else, subtask success rates reveal how problem solving strategies are used to adapt and form new solutions attempts: we make best guesses as to what to do next and how to

proceed, and if an attempt does not work, we try something else. The first trial does not need to be successful.

On the other hand, the number of subtasks may show something of the users' typical problem solving methods. One could infer, for example, that the Notes users relied more on their assumptions, while the others proceeded more carefully, looking for more clues in the instructions before making a first guess or forming a subtask. It would be interesting to study whether different user types (beginners to advanced users) differ in this respect. However, in this case I believe that the main issue was the difficult user interface of Lotus Notes that made the users check and correct their assumptions more than was needed with the other applications.

The Notes users (users 1 and 2) needed my assistance in 30 and 40 subtasks respectively. This was far more than what the other users required; I assisted user 3 only one single time, and user 4 and user 5 needed my help for 4 and 3 subtasks. Most of my help was needed for indicating and explaining bugs in the software and helping the user to open and close or change between the application windows. The Notes users needed more help because of bugs in the application, and because they were less confident with Notes controls, some of which worked differently from other Windows applications.

Apart from the bug and Windows and application controls issues, assistance was asked for when users had trouble locating, identifying or recognising an item on the screen. The documentation stated what they have to do and in which window, but sometimes the named window seemed impossible to find, and required that I physically point it out or explain where on the screen they should be looking. This has to do with deixis and reference between documentation and on-screen items. Sometimes the users did not know if they had the correct instructions, and I had to confirm whether they should follow the instruction or not. This was a problem of establishing the relevance of the topic. However, I did not analyse the assisted

subtasks separately; those that failed are included in the analysis of unsuccessful subtasks. However, this might be an interesting area to look further into in another study. What kind of information does concurrent person-to-person communication convey that documentation cannot explain clearly?

The problem types with unsuccessful subtasks are presented below in Table 6. A subtask was considered unsuccessful if its result was not what the user had expected. Some of the subtasks that did not succeed, however, were considered to have an indeterminate result if the user believed that he had completed the task successfully and if it did not obstruct completing the higher-level task.

Some of the users had problems with standard Windows functions, such as switching between the open windows or recognising the difference between closing and minimising a window. Application-specific functions were also problematic at points: the applications are not intuitive enough to enable a new user to immediately understand what should and could be done with all the controls in the user interface. Usability tests usually focus on finding these issues from users: where the application controls are difficult to use, where they are easy to use, and how their usability could be improved. According to Table 6, documentation caused the majority of problems as indicated by the percentages[19].

TABLE 6. Problem types in unsuccessful subtasks.

	Windows controls	Application controls	Documentation	Test settings	Total (Doc. problems)
User 1	10	23	21	-	54 (39%)
User 2	-	17	41	3	61 (67%)
User 3	3	6	28	-	37 (76%)
User 4	-	11	18	1	30 (60%)

19. One (Notes) user had more problems with Windows and application functionality than with documentation. This could depend partly on the user not being very familiar with using Windows applications, and on the difficulties of the user interface. It may also suggest that the document (Notes Help) was rather helpful.

TABLE 6. Problem types in unsuccessful subtasks.

User 5	7	1	17	-	25 (68%)
	20	58	125	4	207 (60%)

With documentation problems I mean everything that concerns either the user interface text or the help and manual texts. The difference between problems in application controls and in user interface text can be described with examples: an application problem means that the user does not understand, for example, how addresses can be added to a mail alias, and a user interface text problem means that the user does not understand, for example, that the name "Directory" refers to an address book and that he should go there in order to perform certain tasks.

These results might be interpreted to show that documentation in fact causes more problems in software use than anything else. However, one has to keep in mind the test settings: the users were specifically asked to find the solution first in the documentation before trying to do anything with the application. This is a very untypical situation with software use, since people normally just go ahead and try out things with the application rather than read documentation. As the purpose of the test was to find out where there were problems in the documentation, rather than in the application use, the results did produce instances of failed communication and thus it served its purpose, but no conclusions can or should be made about the relative numbers of problem types.

Assigning the problems to Gricean maxims was rather difficult, since I found that violations of more than one maxim were often involved in any given communication problem. It was difficult to determine, for example, whether a problem was caused by difficult terminology (maxim of manner) or unclear relevance (maxim of relation). I therefore assigned a single problem to more than one maxim when needed. I found that the maxims of relation (91 instances) and quantity (81) were violated most often, and the maxim of quality most seldom (6 instances). The maxim of manner was violated in 53 instances. Software bugs caused the

problem in 13 subtasks; those could just as well be assigned to the maxim of quality, since the documentation does not describe the bugs at all or gives erroneous information concerning application functionality in those instances.

Problems with the maxim of relation were especially typical for Notes. TWO and MSE suffered from problems with the maxims of quantity and manner, because they included very long topics from which it was difficult to find what one wanted. Notes and MSE sometimes used rather technical language, which also violates the maxim of manner.

4.3.4 Applying Gricean maxims

I applied the Gricean maxims to the problems in subtask accomplishment in the manner described below. I will refer to the test user in each case as "he", even though there were both male and female test users. I do not want to emphasise the gender of the user or give away their identities, because those are not relevant to the issues discussed in this study.

Maxim of quality

I interpreted violations of the quality maxim as breakdowns in communication in which the user cannot or should not believe what is on the screen (or in print); in other words, it concerns false information. This applies both to software bugs and errors in documentation. I also added a less obvious incident in this category, namely, overemphasising a trivial step in a task. In those cases the users tried to repeat a step that they had, in fact, already done.

As a result of a quality maxim violation, the user performs an action that he is not supposed to perform.

Example 1 - Lotus Notes. The user is attaching files to a mail message. The instruction starts with the phrase: "With the document in edit mode" (help topic "Attaching a file to a document"). The user thinks that he has to choose an 'edit mode', so he clicks the Edit menu

to find the command. In reality, 'edit mode' means that the user has enough rights to modify the document. Mail messages in Notes are documents that the user (their creator) always can modify, so the instruction is completely unnecessary in the user's context and results in needless action. The note about the edit mode is not relevant, so this example could just as well be interpreted as a violation of the maxim of relation.

Example 2 - Lotus Notes. The user clicks the Help menu, Help topics command in order to open the help window. Because of a bug in the application, the help window appears very small at the bottom of the screen, instead of opening at the center of the screen. The user clicks the same command again, because he does not notice the small help window.

Example 3 - Microsoft Exchange. The user clicks the Print button in the help window. Because of a bug, the help window is closed and the topic is not printed. The user repeats the action, because he wonders if he could have clicked another button (for instance, Close) by mistake.

In Examples 2 and 3, normal documentation cannot really help the communication problem, since the application is clearly not designed to work that way. However, a list of software bugs and how to recognise them can help the user realise what is happening. Bug lists are usually given as Release Notes which often seem quite technical documents because the bugs relate to specific hardware or software combinations. Information about bugs is usually told person to person and found out after repeated trial and error. This is a problem area for which it is very hard to think of improvements, other than better testing for program code and fixing the bugs before the product is released.

For Example 1, an improvement would be to write a separate topic for attaching files to mail messages, or to mention 'edit mode' as a tip or check point if the user cannot perform the task otherwise. The most typical situation is that the user does not need to be concerned with

edit mode. Therefore, it should not be included in the step list since it implies the need for an action to be performed by the user.

A generic correction for Example 1 type problems would be to check all default settings and move their check points from within the action steps to outside the step list.

Maxim of relation

I regard as a violation of the relation maxim a situation in which the user is confused about whether the instructions apply to his task. A typical case would be a user judging a topic as not relevant for his current task, even though it were exactly the right topic. Its counterpart is the user's opening a non-related topic because it seems to him relevant for the task. Another case type for relation maxim violations would be the user's looking for information under a wrong heading or subject matter. A related problem is not knowing where to look; instead, the user tries to find clues for action or task names in the help index or in UI command names or screen elements.

Example 4 - TeamWARE Office. The user is supposed to create a group alias. He is not sure what the correct terms would be in the current application, so he goes to the help Contents page and finds an overview topic "Sending mail messages". He expects to find clues there as to the use of aliases. In this case, he did not find the clue he was looking for in the help. Instead, he went to the Address Book window in the application and noticed an interesting looking button with the tool tip "Create Private Group". This is where he finally found the relevant terms.

Example 5 - Lotus Notes. The user looks for a way to go to his address book, and opens the topic "Ways to enter user information in your Personal Address Book". He probably noticed the word "enter" that triggered recognition. Especially when the topic names are long, users seldom read them completely but react to single words in them.

Example 6 - Lotus Notes. The user is searching for instructions on adding attachments to mail messages. He has opened and closed the help topic "Attaching a file to a document" several times, and once again as he opens the same topic he sighs "Oh no it is that document topic again" in a belief that it is not relevant to his task. As a matter of fact, the topic is exactly right for the task, but the terminology difference between 'mail' and 'document' is so confusing that the user cannot see the connection. Both Notes test users had the same difficulty. The problem could perhaps be assigned to the maxim of manner as well, since after all, it is the terminology or wording that is obscure. However, I interpreted violations of the manner maxim somewhat differently (see below). Not being relevant means that the user cannot locate the correct instructions (even if they exist). Sometimes, of course, instructions may be missing. Consider the following example.

Example 7 - Microsoft Exchange. The user is looking for instructions about changing his password. The help contains some password related topics, for example, "Changing your advanced security password" and "Adding or changing a password for a personal folder file". However, those describe other tasks than the one he should perform. He reads the topics and even tries to perform the steps in one, although the topic title sounds dubious to him. Since better alternatives are not available, he deduces that one of these should be relevant to his task. In reality, the task required changing the Windows password and was not included at all in the MSE help. The user was quite experienced and went to the help of another application (Windows Program Manager) where he found correct instructions.

In Example 7, conceptual knowledge should have been given to the user about sharing the password with Windows. The information may have been included in one of the overview topics, but it was not accessible from the keyword index. Missing information is obviously a violation of the maxim of quantity, but, as seen in Example 7, it may also imply false relevance to existing topics. Therefore the user's tasks with the computer should be regarded as one

continuous session, not as separate sessions with different software applications. The user cannot always tell which application he is using (see also Windows controls problems).

What could have been done in example 4 and what writers often imply relevance with is by adding "Related topics" or "See also" links to other topics. Lotus Notes had only three "See also" topic links in their help screens, which seemed to me a very useful practice: it saved the user's effort in choosing and following the links. Another way for implying relevance is, for example, grouping topics by subject matter in the Contents pages which the user in Example 4 also tried.

Users' means for deducing relevance include elimination. For example, if there were more than 3 subtopics in a group (under a keyword), it seemed to become quite difficult to determine by the topic heading which single topic would be the most relevant. Therefore, the users opened each topic in turn in order to compare or discard their contents (or selected altogether another keyword with fewer subtopics).

The difficulties in assessing topic relevance are closely dependent on the user's mental model of how the application works and what the task contains. Single terms may cause the user either to believe firmly in the relevance of a topic or to reject it. As seen in Example 6, the user had repeatedly opened the correct topic but was not able to interpret its contents before he was told that it was in fact the right topic. After that, he soon understood how the concepts related to his task and corrected his mental model of the application.

Maxim of quantity

Violations of the quantity maxim mean, in my interpretation, a situation where the user gets too little or too much information about the subject matter or task. Typical difficulties with too little information occur in locating screen elements or opening a correct application window. Too much information means too long topics: the users cannot be bothered to read them

through to elicit a tiniest piece of information they are interested in. Similar problems are long lists of related topics or index entries.

Example 8 - TeamWARE Office. The user is trying to change his password, which requires opening a correct application window by double-clicking "the TeamWARE session icon on your Windows desktop" (User's Guide, section "Changing your password"). The user, even though he is quite familiar with several Windows applications, has no idea what "desktop" means and where to find the session icon. The user becomes very frustrated and requires my assistance to find the right place at the botton of the screen. Both TeamWARE users had the same difficulty. The wording did not give enough information about where to go on the screen. The other user commented, interestingly enough, that "it's a typical problem: they don't tell you where exactly one should click".

Example 9 - Microsoft Exchange. The user is about to start writing mail. He looks for a topic in help index with the keyword "sending". A huge list of entries opens in the Topics found dialog with several screenfuls of topic titles to scroll through. The user at once closes the Topics found dialog and tries second level keywords under "sending". All mail-related keywords in the help seem to contain long lists of related topics, and the user gets very frustrated. Several topics that he opens are too general and do not contain the step-by-step instructions which he is looking for. Some titles are very similar and only differ by a singular or plural in topic name (for example, "Overview of sending a message" and "Overview of sending messages"). Their difference is very hard to determine even after opening each topic.

Difficulties with the quantity maxim seem to cause severe problems and frustration. The difficulties as in Example 8 could be avoided by inserting pictures of screen elements with labels in the documents. The problem could, once again, be assigned to the maxim of manner because of vague or unclear terminology. However, in my interpretation, the problem is not in using the term but in not being more explicit as to which corner of the desktop the icon would

be. For example, adding a comment that the session window is "minimised" (rather than call it an icon) and "at the bottom of the screen" might have made the difference.

The difficulty in Example 9 also caused the user to lose his way completely. He got so frustrated that he did not even notice the Related topics buttons at the top of the help topics that might have guided him to the correct instruction quite soon. Still, the use of Related topics links does not, in my opinion, justify too long keyword lists. This is because each user seemed to be sticking to one single method of searching (typically, index); therefore, the navigation methods should be independently efficient and not designed for complementary use.

Maxim of manner

Violations of the manner maxim include instances where the user's language (or idiolect) skills are not fluent or specialised enough to interpret the instruction unambiguously. Another violation of the manner maxim is wordiness, or writing too long topics.

Example 10 - Lotus Notes. The user reads about attachments (help topic "About attaching files"). The concept is quite new to him and he tries to understand what kind of files can be attached. The topic says that "binary files" may be attached to messages. The term is unknown to the user and adds to his uncertainty about the concept of attachments. In reality, binary files refer to any files that contain other elements than unformatted text, i.e., almost all files are binary files.

Example 11 - TeamWARE Office. The user is trying to use the address book which is empty by default. The search for addresses is rather difficult, since it requires strict syntax in the search term and causes some bewilderment in the users. The user clicks the Help button to open a long topic "To search for members for a private group in your Personal Address Book" which he looks at but soon closes. He cannot be persuaded to read such long topics. Some time after that, the user is in another window of the address book and opens help again. This time

the topic name is "To search for members for a private group in the Organization Directory". Otherwise, the topic seems very similar to the previous one. The user comments "It's too long for me to want to read it now -- I'll just try and see".

The users' impatience with long topics becomes especially noticeable when they skip over the information which they were looking for. Lotus Notes' help topics seemed shorter and more readable than in the other two applications; but then again, Notes used quite technical terminology.

4.3.5 Context recognition

Some of the problems in documentation use could not be assigned to a single maxim at all, or the maxims seemed inadequate in describing and accounting for the problem. The instructions were there, but the users were sometimes unable to "see" them, because of their expectations or preconceptions. One example of this is when a user rejected a topic as irrelevant time after time, and only after he was finally told that it was the right topic, could he interpret the instruction as applying to his situation.

I think that this is an issue that concerns the users' mental models. Users form ideas of what the system is like, how it works and what elements are included in it. As they work with the system, they compare the on-screen happenings with their mental model and make adjustments or corrections when required. However, changing the mental model may not be very easy, and that may cause the user to take a lot of time or more than one trial to become convinced enough to change the model. This can be noted, for example, as users retrace and repeat their actions to see whether the same (unexpected) output is produced each time. Mental models facilitates system use, since they suggests working assumptions of causal relations in the system: if I do this, the system does that, and in order to achieve that, I first need to do so and so.

Highly specified terminology seems not to be a part of mental models. Most, if not all, of the test users were somewhat unfamiliar with Windows terminology, even if they were otherwise proficient computer users. Users do not usually worry about individual terms that they do not recognise, but if an action is not succeeding as expected, they quickly start doubting their understanding of the term and wonder whether the application is in the state that their mental model suggests. Thus, when terminology was a problem, the users first checked their assumptions about the meaning of the term, rather than changed their mental model. It was hard for the users to know what specific words the application would use for a given item or concept; but usually only one or two trials with keywords was enough to find a correct topic in documentation.

Once the users succeeded in expanding or correcting their mental model, even difficult tasks became quite easy. On the other hand, if the mental model did not permit a certain train of thought, users could not understand it from the documentation either (as seen in example 6 above).

The shared context or background knowledge that communication theories present as a prerequisite for communication seems indeed to be the most significant factor in ensuring communication efficiency. Mental models can be seen as a context for performing tasks with software. The immediate problem then is, how can mental models be shared in documentation? Documentation should be able to anticipate and support individual users' mental models and prompt the users to correct their models if need be. This would probably be easiest to do by providing an explicit overview of the concepts in the system; but users seldom read overviews unless they really become desperate. They want to learn by doing and meet the concepts in their natural context. However, if the documentation mentions a concept in a wrong task context (such as the "edit mode" in example 1), users may easily become distracted.

This suggests that there are at least two levels of context here: the overview or mental model of the system and its parts, and the individual and specific task context that has to do with real-world goals for which the system is only a tool. The task context, of course, is more important to the users; but the mental model or system context may cause the users to reject the instructions for proceeding with their tasks.

The problem of context is thus a separate issue from Gricean maxims, and the maxims cannot be used for explaining it. The communication theories seem not to provide solutions for recognising the context. Topic recognition can be done on a textual level, even with syntactic cues. Context may be deduced, for example, from general terminology, genre and references to times, places and ideas. The context becomes more difficult to recognise when the references to external stimuli are not familiar or known to both participants in advance, as is the case with using software. Other challenges include incorporating the task domain context, which may vary from one organisation to another, and accounting for software bugs, which may prevent mutual understanding altogether. Establishing and recognising the context or contexts in this kind of unfamiliar, unspecified and unknown territory seems like a very interesting research topic for further studies, and it could bring new insights to communication theories in general.

4.4 Conclusions

The purpose of the empirical test was to find out whether Gricean maxims can be used in explaining communication problems in the use of software documentation. For that, usability tests were conducted in which test users worked with different software applications and used their documentation. The tests differed from traditional usability tests in that the user were asked to study the documentation first, before trying to perform the tasks. Also, the analysis of the results was different than with traditional usability tests, in which the main point is to find

specific problems in application use. The tests in this study were analysed in terms of all problem types, not just the application specific ones; and the main concern was finding out where documentation ceased to be helpful.

The results were interpreted through the Gricean maxims and some improvement ideas were found. Violations of the maxim of relation make sense intuitively if we think of user documents as being off the point and not saying everything we wish to know. The maxim of quality was not violated very often, which also makes sense intuitively if we expect documentation to be truthful and accurate. However, software bugs (which may also be interpreted as violations of the maxim of quality) sometimes prevented users from proceeding with their task. Violations of the maxim of manner did not cause total frustration, but violations of the maxim of quantity caused total blocks that the users did not seem to be able to pass without extra help. It appeared that not being informative enough, or being too informative, caused the most serious problems. These problems included, in my interpretation, locating items from the screen. The design implication from this is that we should start making documents more referential, adding pictures and screen shots instead of insisting on the use of technical computer terms. It is also erroneous to expect users to know their Windows basics, because they do not. Other design implications would be to keep topics short and to add subheadings for every piece of information included in a topic. Hyperlinks from topic to topic were seldom followed; plain text topics would therefore seem better, provided that they would be easily accessible from the index. Too large an index frustrates the users by offering too much choice for each selection.

It seems that difficulties in documentation use can certainly be assigned to communication problems and be interpreted in terms of Gricean maxims. However, improvement ideas cannot say "add more relation!". An issue for further research is how the violations of Gricean maxims could be anticipated and corrected in documentation, and how

the situation-specific nature of the maxims could be taken into account in designing documentation.

However, not all of the problems that emerged in the tests could be explained with the maxims only. Recognising and establishing a context can be seen as an open issue that none of the communication theories sufficiently account for. While terminology, topic and context are tightly interrelated in recognising the context, the results of this study indicate that terminology was the least likely factor to be trusted by users in determining the context. Terms were interpreted from the idea of context rather than used primarily for establishing the context. Mental models, on the other hand, were a decisive factor for interpreting context. In documentation and computer use, the users' mental models caused the most difficult problems. Mental models are changed as new evidence is received, but sometimes they seem to freeze to a rigid set of assumptions in which new evidence cannot be integrated anymore, perhaps because the cognitive load grows too big in using a software system that may contain lots of unexpected features. On the other hand, mental models may also greatly enhance communication. It would be most interesting to study the workings of the mental models, their formation and how they can be affected and changed.

The context may be a multi-layered concept in documentation, containing system, domain and task levels. Sometimes the user is looking for a single piece of information that required decoding it from documentation. Sometimes the user wants solutions for his real-world tasks, which required communication with the software (and its documentation) in order to specify the task, the needed elements, and the procedure for accomplishing the task. The single pieces of information help to refine the users' mental model of the system that again helps or hinders the understanding of new concepts and new pieces of information about the system. The levels of context are an interesting issue and would certainly benefit from further study.

Overall, the empirical test was successful and confirmed the hypothesis, as well as brought out a new research issue of the context. However, the interpretation and assignment of the maxims to the problems was arbitrary, and other researchers might interpret the maxims differently. Therefore any conclusions on the relative importance of each maxim can only be preliminary and should be verified in further studies. The testing method seemed adequate and provided sufficient data. Differences between individual users and their computer literacy were not considered. Those aspects could shed light on the type of problems encountered by beginners to advanced users, and help in providing the correct contexts for different type of users, but the sample in this study was too small to consider differences between user types.

5 Discussion

This study considered software user documentation with respect to efficiency. User documents are utility texts, whose main purpose is to enable the use of another product or service. Since user documentation is supposed to be useful, it should be efficient, effective and accurate, so that the user loses no time from his actual task with the product.

When people use software, they prefer asking a colleague for help to using documentation. This may depend on a number of reasons; for example, a desire for social interaction, efficiency gained from the organisation-specific domain and task knowledge that colleagues share, more familiar and less threatening vocabulary among colleagues than in computer jargon, and help in identifying and pointing out the context-dependent important items in the software. Even when users do use documentation, they prefer third party documents to official documentation. Studies have indicated that third party documents have richer social interaction, which is regarded as their main advantage to the neutral environment of official documentation. Official documentation, on the other hand, has the advantages of being available also for repeated use and having a more complete information content.

Software user documentation development is currently largely based on usability methods and ideas from cognitive psychology instead of communication theories. A usability ideal is user-centred design that focuses on finding out and building on the users' knowledge of the domain and the tasks, and designing the software to reflect that information. Similarly, user documentation should be based on users' tasks, so that a good topic would be, for example, "Sending mail" rather than "The send function". The requirement for audience analysis is a shared feature with the rhetoric approach to communication

Cognitive psychology studies the framework in which people both physically and mentally notice, process and learn new concepts, procedures and ideas. This framework includes issues such as legibility, readability and problem solving. It also suggests that users

have mental models that help explain and predict the functioning and behaviour of systems, and imply the causal connections inside systems. Users continually check their mental models against the workings of the system.

Usability and cognitive studies have contributed to a number of techniques for creating "good" documentation. These include check lists for using easily understandable language and syntax, skeletons for topic types and document structures, and methods for studying users and their working contexts. However, documentation development still has to face requirements for production efficiency. This sets limitations, the most important of which is probably the need to be universally acceptable in order to reduce localisation costs. In other words, user documents cannot be targeted too tightly to specific groups of users. This means that different domains, organisation specific cultures or differences in users' computer literacies cannot be very well taken into account. Further, instructions for co-using other software, such as the workings of the operating system, are not provided, since users are expected to have those instructions provided by someone else. These restrictions are not necessary in third party documents, which makes them more helpful. In addition, third party documents give shelter to the users from information overflow; they act as gatekeepers and help users judge what details are important and what not. These are some reasons into why third party documents may seem more believable and thus more efficient than official documentation.

Since there are different kinds of communication with the user that are considered more efficient than using the official documentation, I wanted to find out how communication theories approach the efficiency of communication and whether that could be used in assessing or improving software user documentation. There are tens, if not hundreds, of communication theories that address specific issues. Rather than contradict, these theories complement each other and emphasise their points of view as answers to their specific research problems. I considered four approaches to communication, which I called theories of surface meaning,

acquired meaning, intended meaning and goal-oriented meaning. I explored their implied and outspoken definitions of the efficiency and success of communication, and how they apply to software user documentation. Some of the theories originally describe spoken communication, so I wanted to find out whether they could be applied to written communication as well in the form of user-software or user-documentation interaction, which often follows a conversation pattern of question and answer in an effort to define the problem and find a solution. All of these theories can be used to explain some aspects of documentation.

Theories of surface meaning are based on the assumption that there are predefined meanings which can be conveyed one-to-one from speaker to hearer. The question of efficiency is whether all the information is included, and whether the hearer acts as the speaker intends him to. Noise, interpreted as any distraction away from understanding the message, is to be minimised in order to be maximally efficient. A feedback loop is an important element of communication.

Theories of acquired meaning place communication instances in a network of other such instances or texts. Not only the actual communication creates meanings, but meanings are acquired also by references to and reminders from other interactions. There are several layers of meaning, and several possible interpretations for any given instance of communication. It depends on the situation, which interpretation becomes more prominent than another, and the interpretation may change over time. Cultural and social elements have an important role in the interpretation, and may cause conceptual gaps in understanding the message. Participants' background knowledge, for example, computer literacy, is another important factor for interpretation. Overall, meanings are in constant interaction or fluctuation with other meanings and messages. Genres offer one indication for placing the message in the intertextual network.

Theories of intended meaning again assume that predefined meanings can be transferred, if only the audience can be taken into account well enough. Actions can be performed through

words, if participants can be persuaded to agree with the conditions. The audience interprets not only the message but also its sender's values, attitudes and beliefs, and decides on those premises whether to accept the meanings or not. The reliability of the sender is central to the acceptance of the message.

Theories of goal-oriented meaning are based on a tool-like assumption of communication. People communicate in order to achieve different goals, some of which may be just to communicate. Others may include performing real-world tasks. Therefore communication can be a rational, efficient and purposeful way for achieving one's goals. All previous instances of communication bring experiences and expectations to the new instances, and thereby affect interpretation. The success of communication depends on whether goals are achieved; it does not require a shared understanding or transfer of intended meanings. The participants are free to make their own interpretations of each contribution; and they make their interpretations with respect to their goals.

Grice's Cooperative Principle suggests that communication is successful if the participants can achieve their goals through communication. The four maxims of quantity, quality, relation and manner ensure an efficient and effective way to achieve one's goals through communication, and therefore it is rational for participants to observe those maxims and also to assume that the other participants are observing the maxims. In the use of documentation, one can analyse communication problems through the Gricean maxims as a failure to observe one or more maxims. The empirical test made as part of this thesis indicated that the worst problems occurred when the maxim of quantity was violated. Too little or too much information was thus worse than erroneous information (violation of the maxim of quality), which the users could notice themselves and assign the blame to the document. Violations of the maxim of quantity were also rather frequent; only the maxim of relation was violated more often. On the other hand, something that user documentation is often blamed

for is its use of difficult terminology and jargon; but the problems that were caused by violations of the maxim of manner were less frequent in the test than those caused by violations of the maxims of relation and quantity. Therefore it seems that users are able to perform tasks without taking too much notice on terminology, unless something goes wrong. This may also explain why they so not seem to learn the terms; there is not direct need for memorising a correct software term in order to use the software.

The establishing of context is a matter that communication theories do not discuss very much. A shared context or background knowledge is recognised by many scholars as an important prerequisite as well as a result of communication. However, as emerged also from the empirical test in this study, if the user and the documentation had different ideas about the context, the interaction became unsuccesful. Users had serious difficulties in trying to locate a topic that would share their idea of the context; and they did not recognise the context that the documentation suggested, if they were not already assuming it. This is almost a vicious circle, and can only end when the user somehow becomes able to change his mental model or preconception about the context. This might happen through the help of a colleague, through training, or if they could be persuaded to read conceptual information. The concept of context may include several layers, for instance, those of task, system and domain. These can be compared to the similar levels of knowledge that are present in problem solving, as discussed in cognitive psychological theories. Some of the communication theories recognise context recognition as a problem, but have no means of explaining or solving it. Therefore I see this as a potential area for developing the communication theories further.

This study confirmed my research hypothesis that documentation should follow Gricean maxims. It became clear that communication theories and especially Gricean maxims can take the analysis of documentation further than traditional usability studies. However, the empirical test also revealed the problem of context recognition, which was something that was not

anticipated at the beginning of the study. In the interrelation of topic, context and terminology, terminology did not seem to have a decisive role in helping to determine the context. However, mental models have a very important role in determining the context. Mental models may block the recognition of context altogether, unless the user already has a correct mental model. The importance of mental models shows that communication theories are not enough to explain documentation success and efficiency; research on usability and cognition and communication studies complement each other, and technical communicators can benefit from both.

References

Blakar, R. 1985. "Towards a Theory of Communication in Terms of Preconditions: A Conceptual Framework and Some Empirical Explorations" in: Giles, H. and R. St. Clair (eds.) *Recent Advances in Language, Communication, and Social Psychology*. London: Lawrence Erlbaum Associates.

Brockmann, J. 1990. *Writing Better Computer User Documentation: From Paper to Hypertext*. New York: John Wiley & Sons. 2nd ed.

Brown, G. and G. Yule. 1983. *Discourse Analysis*. Cambridge: Cambridge University Press.

Carroll, J. and M. Rosson. 1987. "Paradox of the Active User" in: Carroll, J. (ed.) *Interfacing Thought: Cognitive Aspects of Human-Computer Interaction*. Cambridge, MA: MIT Press, 80-111.

Carroll, L. "Jabberwocky" in: Untermeyer, L. (ed.) 1960. *Collins Albatross Book of Verse*. London: Collins.

Chang, F. 1987. "Writing Text for Many Readers: A Role for Technology" in: Wagner, D. (ed.) *The Future of Literacy in a Changing World*. Oxford: Pergamon Press, 302-318.

Coe, M. 1996. *Human Factors for Technical Communicators*. New York: John Wiley & Sons.

Coney, M. 1992. "Technical Readers and Their Rhetorical Roles". *IEEE Transactions on Professional Communication* 35:2, 58-63.

Coney, M. and C. Chatfield. 1996. "Rethinking the Author-Reader Relationship in Computer Documentation". *Journal of Computer Documentation* 20:2, 23-29.

Cook, G. 1995. "Principles for Research into Text Accessibility" in: Nyyssönen and Kuure (eds.) 1995.

Cooper, A. 1996. "Three Models of Computer Software". *Technical Communication* 43:3, 229-236. Reprinted from Cooper, A. 1995. *About Face: The Essentials of User Interface Design*. IDG Books Worldwide.

Craig, R. 1999. "Communication Theory as a Field". *Communication Theory* 9, 119-161.

Denning, P. and P. Dargan. 1996. "Action-Centered Design" in: Winograd, T., J. Bennett, L. De Young and B. Hartfield (eds.) *Bringing Design to Software*. New York: ACM Press and Addison-Wesley Publishing Company, 105-119.

DeVito, J. 1992. *The Interpersonal Communication Book*. New York: HarperCollins Publishers. 6th ed.

Dillon, A. 1992 "Reading from paper versus screens: a critical review of the empirical literature". *Ergonomics* 35:10, 1297-1326.

Dix, A., J. Finlay, G. Abowd and R. Beale. 1998. *Human-Computer Interaction*. London: Prentice Hall Europe. 2nd ed.

Dorney, J. 1996. "Plain Language History". Available in the Internet at: http://www.plainlanguage.com/ERIC.TXT.

Ellis, A. and G. Beattie. 1986. *The Psychology of Language and Communication*. London: Weidenfeld and Nicolson.

Ephron, N. (director) 1993. "Sleepless in Seattle." A film produced by TriStar Pictures.

Fisher, J. 1998. "Defining the Role of a Technical Communicator in the Development of Information Systems". *IEEE Transactions on Professional Communication* 41:3, 186-199.

Fiske, J. 1990. *Introduction to Communication Studies*. 2nd ed., London: Routledge.

Grice, H. P. 1975. "Logic and conversation" in: Cole, P. and Morgan, J. (eds.), *Syntax and Semantics*, Vol. 3: *Speech Acts*. New York: Academic Press, 41-58.

Guzdial, M. 1999. "Supporting Learners as Users". *Journal of Computer Documentation* 23:2, 3-13.

Hackos, J. 1994. *Managing Your Documentation Projects*. New York: John Wiley & Sons.

Hackos, J. and D. Stevens. 1997. *Standards for online communication*. New York: John Wiley & Sons.

Hart, G. 1996. "The Five W's: An Old Tool for the New Task of Audience Analysis". *Technical Communication* 43:2, 139-145.

Haslett, B. 1987. *Communication: Strategic Action in Context*. Hillsdale, New Jersey: Lawrence Erlbaum Associates.

Hayhoe, G. 1999. "Technical Communication: A Trivial Pursuit?" *Technical Communication* 46:1, 23-25.

Holmberg, L. 1995. *The thoughts and practices of user documentation developers*. Gothenburg: Department of Education and Educational Research, Gothenburg University.

Horton, W. 1993. "Let's Do Away With Manuals... Before They Do Away With Us". *Technical Communication* 40:1, 26-34.

Houser, R. 2000. "What Does It Mean To Be a Technical Communicator?" *Intercom*, February 2000, 4.

Isomursu, T. 1997. "In a Way It's a Problem You Don't Know That Data Terminology - On the Relationship Between Vocabulary and Text Accessibility" in: Nyyssönen and Kuure (eds.) 1997.

Isseroff, A. 1999. "Comment on Technical Writers Gain Control". *TC-Forum*, September, 4.

Järvi, O. 1995. *Språket på datorskärmar: en undersökning av användargränssnitt i tre datorprogram*. Licenciate thesis. Vaasa: University of Vaasa.

Kaihu, H. and T. Rouvi. 1999. *Käyttöliittymämetaforat ja käytettävyys*. (*User interface metaphors and usability*.) Unpublished Master's thesis, University of Tampere.

Karvonen, P. 1995. *Oppikirjateksti toimintana*. (*Textbook as Activity*.) Suomalaisen Kirjallisuuden Seuran Toimituksia 632. Jyväskylä / Helsinki: Suomalaisen Kirjallisuuden Seura.

Kasprzyk, J. 2000. "What is 'Computer Literacy'?" Available in the Internet at: http://kafka.salem.mass.edu/CompLit.htm.

Laitinen, K. and J. Taramaa. 1994. *A Theory to Support the Use of Natural Naming in Software Documentation*. Oulu: Department of Information Processing Science, University of Oulu.

Lammintaus, A. 1999. "Trends In Future Technical Communication" in: *Proceedings: 46th Annual Conference*. Arlington, Virginia: Society for Technical Communication. 239-242.

Leech, G. 1983. *Principles of Pragmatics*. London: Longman.

Leonhard, W., L. Hudspeth and T. Lee. 1997. *Word 97 Annoyances: Taking Charge of Word 97*. Cambridge: O'Reilly & Associates.

Leppänen, S. 1995. "Analysis and Interpretation: Theoretical Positions, Subjective Interpretations and the Persuasion of Language" in: Nyyssönen and Kuure (eds.) 1995.

Levinson, S. 1983. *Pragmatics*. Cambridge: Cambridge University Press.

Longman Dictionary of Contemporary English. 1990. Longman. 2nd ed.

Mac Aogáin, E. and R. Reilly. 1990. "Discourse theory and interface design: The case of pointing with the mouse". *International Journal of Man–Machine Studies* 32, 591-602.

Marcus, A. 1998. "Metaphor Design in User Interfaces". *Journal of Computer Documentation* 22:2, 43-57.

Meriläinen, R. 1993. *From Silicon Valley to Finland: Translation of software user manuals from English into Finnish*. Unpublished Master's thesis, University of Tampere.

Microsoft. *Getting Results with Microsoft Word 97: Real World Solutions for the Work You Do*. Ireland: Microsoft Corporation.

Montgomery, M., A. Durant, N. Fabb, T. Furniss and S. Mills. 1992. *Ways of Reading: Advanced reading skills for students of English literature*. London: Routledge.

Moon, Y. and C. Nass. 1996. "How 'Real' Are Computer Personalities? Psychological Responses to Personality Types in Human-Computer Interaction". *Communication Research* 23:6, 651-674.

Mårdsjö, K. 1994. "Man – Text – Technology: Technical Manuals as Means of Communication" in: Steehouder et al. (eds.).

Nielsen, J. 1993. *Usability Engineering*. Boston: AP Professional.

Nieminen, L. 1994. *Maniement d'un logiciel de traitement de texte et ses deux traductions*. Unpublished Master's thesis, University of Turku.

Niska, H. 1999. *Text Linguistic Models for the Study of Simultaneous Interpreting*, chapter 4: "Text typology." Part of licenciate thesis: *Textual Processes and Strategies in Simultaneous Interpreting*. Stockholm: Stockholm University. Available in the Internet at: http://www.geocities.com/~tolk/lic/LIC990329p4.htm.

Nystrand, M. 1986. *The Structure of Written Communication: Studies in Reciprocity between Writers and Readers*. Orlando: Academic Press.

Nyyssönen, H. 1995. "Exploring the Accessibility of Written Text" in: Nyyssönen and Kuure (eds.) 1995.

Nyyssönen, H. 1997. "Accessibility and Text-Reader Interaction" in: Nyyssönen and Kuure (eds.) 1997.

Nyyssönen, H. and L. Kuure (eds.) 1995. *Principles of Accessibility and Design in English Texts: Research in Progress*. Oulu: Publications of the Department of English, University of Oulu.

Nyyssönen, H. and L. Kuure (eds.) 1997. *Principles of Accessibility and Design in English Texts: Research in Progress 2*. Oulu: Publications of the Department of English, University of Oulu.

Pilto, R. 1995. "The Role of Structure in Text Accessibility – Contrasting Text and Hypertext" in: Nyyssönen and Kuure (eds.) 1995.

Pilto, R. and T. Rapakko. 1995. "Testing Accessibility of Utility Texts – Work in Progress" in: Nyyssönen and Kuure (eds.) 1995.

Ramey, J. 1988. "How People use Computer Documentation: Implications for Book Design" in: Doheny-Farina, S. (ed.) *Effective Documentation: What We Have Learned from Research*. Cambridge, Massachusetts: The MIT Press.

Rantamäki, A. 1993. *Implicature and Inference on Three levels of Drama*. Unpublished Master's thesis, University of Helsinki.

Rasmussen, J. 1988. "Information technology and Work" in: Helander, M. (ed.) *Handbook of Human-Computer Interaction*. North-Holland: Elsevier Science Publishers. 4th ed., 1994.

Reynolds, C. 1998. "As We May Communicate". *SIGCHI Bulletin* 30:3, 40-44.

Ridgway, L., R. Grice and E. Gould. 1992. "I'm OK; You're Only a User: A Transactional Analysis of Computer-Human Dialogs". *Technical Communication* 39:1, 38-49.

Riley, K. and F. Parker. 1998. "Parallels Between Visual and Textual Processing". *IEEE Transactions on Professional Communication* 41:3, 175-185.

Sacks, H., E. Schegloff and G. Jefferson. 1974. "A Simplest Systematics for the Organization of Turn-Taking for Conversation". *Language* 50, 696-735.

Salmi, L. 1998. "Computers, documentation, and localisation: A methodological perspective". A paper presented at the colloquium *Translation and Cognition*, Savonlinna, October 2-3.

Santhanam, R. and S. Wiedenbeck. 1993. "Neither novice nor expert: the discretionary user of software". *International Journal of Man-Machine Studies* 38, 201-229.

Schiffrin, D. 1994. *Approaches to Discourse*. Cambridge, Massachusetts: Blackwell.

Schriver, K. 1997. *Dynamics in Document Design*. New York: John Wiley & Sons.

Schultz, S. and J. Darrow.1993. *The Digital Style Guide*. Digital Press.

Slobin, D. 1979. *Psycholinguistics.* Glenview, IL: Scott, Foresman and Company. 2nd ed.

Smart, K. 1999. "Establishing Dimensions of Quality for Technical Information". *Intercom*, May 1999, 40-41.

Sperber, D. and D. Wilson. 1986. *Relevance: Communication and Cognition*. Cornwall: Basil Blackwell.

Spool, J. 1998. "Communicating the Concepts: Learning Design from Tax Software". Presentation at the *Info Online Conference*, Chicago, November 3.

St. Amant, K. 1999. "When Culture and Rhetoric Contrast: Examining English as the International Language of Technical Communication". *IEEE Transactions on Professional Communication* 42:4, 297-300.

Steehouder, M., C. Jansen, P. van der Poort and R. Verheijen (eds.) *Quality of Technical Documentation*. Utrecht Studies in Language and Communication 3. Amsterdam: Rodopi.

Swann, C. 1991. *Language and Typography*. London: Lund Humphries.

Tompkins, P. 1982. *Communication as Action: An Introduction to Rhetoric and Communication*. Belmont, CA: Wadsworth Publishing Company.

Tsui, A. 1989. "Beyond the Adjacency Pair". *Language in Society* 18, 545-564.

Underwood, M. 2000. "The Lasswell Formula". Available in the Internet at: http://www.cultsock.ndirect.co.uk/MUHome/cshtml/introductory/lasswell.html.

Van der Meij, H. 1999. "Supporting the Reader as User: A commentary on Guzdial's 'Supporting learners as users'". *Journal of Computer Documentation* 23:2, 25-31.

Walters, N. and C. Beck. 1992. "A Discourse Analysis of Software Documentation: Implications for the Profession". *IEEE Transactions on Professional Communication* 35:3, 156-167.

Weaver, C. 1988. *Reading Process and Practice: From Socio-Psycholinguistics to Whole Language*. Portsmouth, New Hampshire: Heinemann Educational Books.

Widdowson, H. 1997. "On the Accessibility Conditions of Textual Meaning" in: Nyyssönen and Kuure (eds.) 1997.

Winograd, T. 1999. "Documentation, Interaction, and Conversation". *Journal of Computer Documentation* 23:4, 3-7.

Wittgenstein, L. 1933. *Logisch-philosophische Abhandlung*. Finnish translation 1984: *Tractatus Logico-Philosophicus eli Loogis-filosofinen tutkielma*. Translated by Heikki Nyman. Juva: WSOY.

Wood, J. 1982. *Human Communication: A Symbolic Interactionist Perspective*. New York: Holt, Rinehart and Winston.