

Developing digital competence - learning, teaching and supporting use of information technology

Jens Kaasbøll, Department of Informatics, University of Oslo



Table of contents

Table of contents	1
Chapter 1. Introduction.....	6
1.1. Why bother?.....	6
1.2. Aims and target groups.....	7
1.3. Related areas	9
1.4. Organisation.....	10
Chapter 2. IT skills.....	14
2.1. Learning IT skills	14
2.2. Instruction sheets – scaffolds for imitation.....	16
2.3. Instruction videos	22
2.4. Training for skills.....	24
2.5. Assessing IT skills.....	25
2.6. Summary.....	26
Chapter 3. Learning business fit	27
3.1. Usefulness	27
3.2. Understanding usefulness of IT in own tasks	29
3.3. Minimal Manuals	30

3.4.	Understanding IT in business.....	31
3.5.	Business oriented models	32
3.6.	Confronting misconceptions	34
3.7.	Summary.....	35
Chapter 4.	Understanding IT	36
4.1.	From skills to understanding.....	37
4.2.	Functional models	39
4.3.	Confronting functional misconceptions.....	43
4.4.	Qualities of functional models	43
4.5.	Structural models.....	44
4.6.	Data structures.....	50
4.7.	Qualities of structural models	54
4.8.	Data types and instances	55
4.9.	Layers	58
4.10.	Semantics	60
4.11.	Structural and functional misconceptions	61
4.12.	Summary	64
Chapter 5.	Learning solving IT problems.....	65
5.1.	Learning oriented users	65
5.2.	Research cycle.....	67
5.3.	Problem solving	69
5.4.	Understanding as a prerequisite for problem solving.....	70
5.5.	Research cycle competence	71
5.6.	Stages of the research cycle	72
5.7.	Strategies for iterations	84
5.8.	Innovative research cycles	86
5.9.	Summary.....	89

Chapter 6. User interface for learning	92
6.1. Learnability	92
6.2. Design for learnability	93
6.3. Inline help	95
6.4. Evaluating learnability.....	100
Chapter 7. Training modules	103
7.1. Training modules for skills and understanding.....	103
7.2. Training modules for improving problem solving competence.....	106
7.3. Teaching a module	108
7.4. Online tutorial for a module.....	110
7.5. Sequence for teaching related topics of IT use	112
7.6. Age levels of abstraction	115
7.7. Instructions, functional and structural models – slide design.....	116
Chapter 8. Training for transfer	121
8.1. Transfer.....	121
8.2. Motivation and objectives.....	122
8.3. Realistic training environment	124
8.4. Summary.....	125
Chapter 9. Evaluation of training	126
9.1. Evaluation of reaction to training.....	127
9.2. Evaluation of learning – assessing competence	128
9.3. Evaluation of behavioural change	132
9.4. Evaluation of impact.....	133
Chapter 10. IT user competence standards	135
10.1. Standards and guidelines.....	135
10.2. Tests	136
10.3. Differences in IT competence levels	140

10.4.	Summary	141
Chapter 11.	Superusers	144
11.1.	Roles	144
11.2.	Community of superuser practice	148
11.3.	Superusers' roles.....	149
11.4.	Organising for competence development.....	154
11.5.	Summary	154
Chapter 12.	IT support	155
12.1.	How IT supporters learn.....	155
12.2.	Support quality	158
12.3.	Improving IT support	160
12.4.	IT support versus superusers	161
12.5.	Summary	162
Chapter 13.	Mutual learning during business fit	163
13.1.	Users learning about IT	164
13.2.	IT personnel learning about business fit	164
13.3.	Joint creation of understanding and skills of new system.....	165
13.4.	User representatives as superusers.....	168
13.5.	Summary	168
References	169
Index	180

Ten golden rules for improving IT users' competence

- 1. Provide users with instruction sheets or videos, also during training.**
- 2. Make sure users understand the usefulness of the IT.**
- 3. Provide functional and structural models and confront misconceptions.**
- 4. Train users so that they can solve problems and learn on their own.**
- 5. Divide training into 30 minutes modules and include problem solving modules**
- 6. Organise training at the same time as the system is installed.**
- 7. Train a local group of users, not only individuals.**
- 8. Identify, organise, authorise and cultivate superusers.**
- 9. Include superusers as trainers and champions for new IT systems.**
- 10. Organise one service desk for all user requests with service minded staff.**

Chapter 1. Introduction

1.1. *Why bother?*

When kids learn information and communication technologies (IT for short) from the kindergarten age and grandma is on Facebook, haven't people become so used to the technology, such that learning is no longer any issue? And isn't the user interface of new apps and gadgets so intuitive that anybody can utilize them without instruction?

Simple applications should be intuitively usable, meaning that no learning is needed, while due to bad design, this is too often not the case (Norman, 1988). However, applications grow with advanced functionality which may be far from intuitive. IT systems are embedded in an interdependent organisational setting, making it difficult for a user to know how others interpret the data entered. IT is pervasive without necessarily appearing as anything like a computer. For instance, a toaster ejects the bread when it becomes dark, because it monitors the colour of the bread, while you thought it had a timer like the old one. Or the toothbrush which does not work properly because you haven't set the time zone. There is no obvious reason why it should require setting the time zone, but sometimes, things are poorly designed. People misunderstand the functioning of gadgets, assuming these work as the user intends, without realising that initial set-up or selection of function is necessary.

Children who are fluent on social media may have little clue about the structures behind the user interface. Even if all IT had interfaces for ease of learning, good design can never compensate totally for a complicated underlying mechanism, such that learning will be needed. Professionals have to deal with 5-20 computer systems from several manufacturers; user interfaces are not consistent, some systems change fast and are used only once in a while, and there are 380 possible routes for transferring data between 20 systems, each which may require specific data formats.

IT is penetrating into most corners of the world. Even if professionals in cities are fluent on computers, the merchant in the village may have a basic mobile phone as her only IT artefact. Thus, there are billions of people worldwide who have not learnt much IT yet.

People learn IT during any activity in life. Learning often follows from of struggling with some task, whether it is entering the cost of the bus ticket in the right place in the corporate accounting system, placing a picture in a document, setting the timer at the oven or copying a text message on the phone.

Being able to operate the technology is a necessary but not sufficient competence for IT users. They also need to understand the purpose of systems to adopt them (Venkatesh et al., 2003), and they need to understand the data. For instance, knowing how to enter a specific term as the index item and why the business needs it is not enough if the user mixes up the keyword and the tag. This book therefore considers IT use competence, concerning both the technology and its use in tasks and the business.

Having a background in computer science, friends and colleagues have often asked me about how to get the computer to do this or that, or sort out things which have gone wrong. Without having had a job in the user support department, I have tried helping out on most types of IT user trouble. I assume that all readers who have come this far in the book share similar experiences as an informal superuser, and that they also have contacted others for help when stuck themselves. I have also experimented with the technology and learnt using it in that way and consulted instruction videos or manuals at times.

My experience is that user learning normally happens informally, and that teaching activities in the form of help and support mostly take place outside formal IT user trainings in classrooms. Research in the area of user learning also points in the direction that informal learning is dominating, such that supporting user learning outside the classroom is essential.

Nevertheless, training of staff in organisations in general improves organisational performance, as shown by a review of 10 years of research (Aguinis and Kraiger, 2009). A summary of 165 studies found a medium to large effect both on individual learning and on organisational performance (Arthur Jr. et al., 2003). These effects were larger than many other interventions in organisation, including feedback on performance or management by objectives.

These results include all training of any subject matter, such that we cannot know whether IT training reaches the same level of positive outcomes. While computer science, pedagogy and psychology are based on millions of research papers, the number of research contributions on learning and training of IT use counts to a few hundreds. These results provide nevertheless a scientific basis for planning ways of improving user learning, and this book aims at organising and summarising the knowledge in the area.

The first chapters therefore focus on what learning IT use *is* and ways of boosting learning, including a trainer, a person, help in the user interface, user documentation or other means. Lessons on user learning, and on how to strengthen it, constitute the background for chapters on designing classroom training and organising support.

1.2. Aims and target groups

This book targets anyone wanting to improve their ability of helping others learn IT use. Three professional groups are considered in particular; IT specialists, school teachers, and lecturers in higher education.

All IT specialists provide informal help, and many start their career as support personnel. Software and hardware vendors develop user interfaces and learning material, they may support their customers, and some IT people also run training courses. Larger organisations may have their own IT department which takes part in developing business systems. Making a large number of users adapt a new system is often initiated through training of superusers who are supposed to support colleagues. IT departments also support their users on standard software and on IT infrastructure, develop user documentation and video tutorials. Specialised IT training and support businesses have the topic of this book as their main activity;

developing and running courses and learning material for other organisations or operating support for software vendors. This book addresses the training and support activities mentioned as well as creating learning material built into software or appearing as independent documents or videos.

While having learnt programming in college, IT specialists have mainly experienced training, support and making user documentation as a learner. When they themselves take on the role of helping others learn, they imitate the teachers, support personnel and user documentation, which they have encountered. Such practical experience is valuable, and it can be improved through coupling with a systematic overview of relevant research. This book provides a comprehensive approach to user learning and can enlighten, challenge or extend the repertoire of practices of self-taught trainers. Pedagogical principles are introduced, such that no prior knowledge of pedagogical theories or related areas is necessary.

While IT specialists are educated in the technology but receive little or no training in educational sciences, school teachers have the opposite background; a solid background in pedagogy, but often little computer science. General pedagogical and psychological principles also apply when learning and teaching IT use, and this book will demonstrate how these principles come into play in IT use learning.

More and more teachers have used IT in their teaching. While this also provides insights into IT and how pupils learn the technology, the main purpose of IT supported learning is learning some other subject matter area, e.g. biology or poetry. Given that teaching depends heavily on the matter taught (Stodolsky, 1988), experience from classroom activities in biology by means of IT does not easily transfer to teaching the technology. This book explains principles of IT which are relevant for user learning and which are independent on specific software, systems or gadgets. This is neither a textbook for Word, Gmail, Facebook, iTunes, Android nor Ubuntu, but a book on how software and IT in general can be taught. While formal education in computer science is no prerequisite for this book, the reader should have experience with commonly used IT, such as office software, the file system, internet software and some gadgets. The book explains IT use from the technology and the business sides, presents learning processes for these areas and suggests how teachers can guide the learners through these processes.

This book is also written as a textbook for lecturers in higher education institutions who will teach the pedagogy of IT use to their students as part of a computer science or information systems curriculum or in a teacher training college. A typical computer science curriculum includes programming, human computer interaction and information systems development. Chances are, the only place user learning was touched upon was in a textbook on information systems development, and it would say that "... before implementing the system, users must be trained." Thereafter nothing more about user learning is mentioned, despite the facts that information systems often fail due to poor user understanding and that user training consumes significant proportions of the project budget. In addition, fresh graduates take up jobs as support personnel or have to develop user documentation, since these tasks are considered being simple enough for new staff by vendors and IT departments. This book is designed to

fill the void in the curriculum, preparing the computer science and information systems students for an essential part of their job.

Unless having done research in user learning, a computer science lecturer should know the constructivist view of learning and one of the areas; information system implementation, human computer interaction or computer science education; in order to use this book in their teaching.

A lecturer in a teacher training college who would like to teach according to this book should know some computer science, including human computer interaction and information systems development. The students would need to be reasonably fluent with computers and IT devices. This book will prepare the teacher students for teaching use of IT to their pupils at any level, including training the pupils to use eLearning tools. It addresses the Technological Pedagogical Knowledge in the TPACK model (Chai et al., 2013).

For teaching teachers of programming, this book should be supplemented with a book on computer science education.

1.3. Related areas

Learning and teaching IT use border several fields of study from different disciplines. From Computer Science and Information Systems side are human computer interaction, information systems implementation and social aspects of computing relevant. Information and library science includes information literacy, which deals with understanding the data. Computer Science Education includes learning IT, Computer Supported Learning deals with students using digital technology, Educational Science concerns learning and teaching in general, and Organisational Learning includes how innovations are adapted and how professionals improve their repertoire of practice.

Human Computer Interaction includes learnability and memorability as qualities of software. Learnability concerns the ease with which a novice user can carry out an operation in the software. Memorability correspondingly deals with re-establishing proficiency of software that is used intermittently. HCI provides guidelines on how user interfaces can be designed to support learnability, for instance by making data and operations more visible and through providing help and guidance in the applications.

Information systems implementation deals with the introduction of new systems in organisations, including the organisation of training and support. Main lessons are that a system is not adopted unless the users understand its purpose and experience improvements in their job performance. User involvement during system development is acknowledged as a way of aligning IT to fit the business, hence providing the necessary basis for adoption.

Information literacy concerns retrieval and analysis of data, which includes issues like search strategies, sources of information, evaluation and use. In addition to such general knowledge, knowledge of the domain is needed to assess information. Also, syntactical skills come in handy when evaluating information, for example, grammatical skills are useful for sorting out

hoax e-mails and statistical competence helps identifying outliers when interpreting numeric data.

Computer Science Education is the study of learning and teaching the technology to IT professionals. Issues on learning programming and programming languages are essential. IT use in this book only touches coding briefly, even if some advanced users use HTML or other languages for extending their control of data. Formulas in spreadsheets resemble programming in the sense that these control automatic calculations. Learning spreadsheets is included in the book, since spreadsheet formulas require basic mathematical and not computer science understanding. However, since both programmers and users have to learn IT concepts, findings from computer science education on conceptual understanding are also relevant for learning and teaching the technology part of IT use.

Computer Supported Learning is the field of study that concerns using IT for learning anything else, for example, learning language through communicating in social media or learning physics through virtual experiments in specifically constructed simulation software. Although IT use is not the subject matter of learning in CSL, findings from this field, e.g., that students who establish abstract understanding of phenomena become better problem solvers are transferable also to the IT domain.

Educational Science consists of a wealth of topics including theories of learning, effects of training, and organising for learning activities. This book draws on the constructivist theory of learning. Central assumptions are that learning builds on what we already know, that people are active and communicative learners, and that learning is triggered through interaction with the environment. Theoretical fundamentalism is discouraged, however. Insights on effects of teaching from a behaviourist view are included, and social learning theory is applied when discussing organisation for learning at the workplace.

Organisational Learning is an area of study that provides insight into how people improve their practices at work, how skills and knowledge are spread amongst colleagues and how newcomers are socialised.

Textbooks in pedagogy, psychology and organisational learning will provide the reader of this book with a deeper understanding of underlying ideas about topics presented here. Such textbooks will also provide more practical guidance into planning, carrying out and evaluating teaching, into the art and science of helping others, and into management of support teams and human resources in companies in general. Higher education lecturers who will use this book for teaching should gain some of this background. However, this book is self-contained and will provide the computer scientist with the necessary, but limited insight into the related sciences.

1.4. Organisation

Part I consists of three chapters addressing learning IT use at three advancing levels; skill, understanding and problem solving. Those who stop reading after Chapter 2 on skills will miss the main points of the book.

Part II builds on the learning model from Part I when considering learnability of user interfaces, user training, evaluation and competence standards in Chapter 6 to Chapter 10.

Part III brings in organisational learning and discusses superusers, IT support and mutual learning during development of IT systems in the last three chapters. It identifies Superusers as brokers between users and IT specialists. This part is less relevant for school teachers.

Part I. Learning IT use

Before embarking on how skills can be learnt, this introduction will present a three level model of learning IT use up to the problem solving competence level, which superusers should master. The three level model will place skills in a larger perspective of user competence.

Categorising increasing levels of competence has been done for the purposes of setting educational goals and for characterising learners' actual performance. Bloom (1956) suggested a general taxonomy for advancing levels of cognitive competence in the 1950s, and it is still used for describing learning objectives. Dreyfus and Dreyfus (1986) suggested a five stage model for skills acquisition. It characterises how practitioners acquire skills over years of collecting experience, modifying the novice's behaviour according to the rules of the textbook to an expert who acts intuitively based on a huge number of experienced examples. While Bloom's taxonomy concerns the learning of theoretical material, expressed in language or some formalism, the Dreyfus and Dreyfus model addresses the refinement of practical skills. Concerning use of computers, an important learning challenge is neither of these, but rather the improvement of competence from skills, through understanding, and into problem solving competence. This improvement is characterised by first being able to do something, and thereafter being able to express it. For example, after having saved files a few times and listened to explanations, the learner may be able to say what it means, where files are stored, and why we do it. Since previous learning models do not address the change from skills to understanding, and further into problem solving competence, this book will bring a specific model of levels of mastery for IT use.

Skills can be applied perfectly without being able to explain how we are doing it, like keeping the balance on a bicycle. This definition of skill does not exclude that one can express the skill, but this expression is only a rehearsal of the physical action carried out. On the other hand, *understanding* requires the ability to express it more abstractly, for example, "File conversion creates a pdf file from the text document." Understanding is complementary to skills, and it includes knowing why mechanisms work the way they do or knowing whom to deal with. Understanding is also called theoretical competence, know-why or textbook knowledge, since parts of it can be learnt from reading books.

For carrying out routine work, users only need the skills required for using the technology to support their business. The reason why users nevertheless may need IT understanding is that understanding will in general ease transfer of skills to new situations, like the introduction of

Learning—Roberto.

Roberto used to carry out complex calculations on his phone, but he often lost track of the results. After having discovered spreadsheets, he uses these instead. Since his change of behaviour is lasting, Roberto has learnt a new IT skill. Roberto also tried setting up a relational database for managing his accounts, but he reverted to spreadsheets. Since he did not stick to this new invention of his, learning might not have taken place.

new software versions, systems, gadgets and IT services (Bransford 2000).

When changing business applications, training superusers to help others is a common strategy (Coulson et al. 2003; McNeive 2009). This implies a two tier need of user competence; ordinary users need the skills for operating the software, while superusers need to be able to solve problems, help others and find out things which they don't know. This means that superusers need competence for problem solving. We will use the term *competence* to denote the ability to do something and *learning* for an increase of competence which lasts at least for a while.

Superusers are expected to fix minor IT problems, find ways of adapting systems to work tasks, and help other users who are stuck, implying that they need an IT use competence above most others. We will say that they need *problem solving competence*. Research has shown that understanding leads to improved abilities for solving problems (Halasz and Moran, 1983, Kiili and Ketamo, 2007, Novick et al., 2009). We therefore state that learning IT use involves the three steps

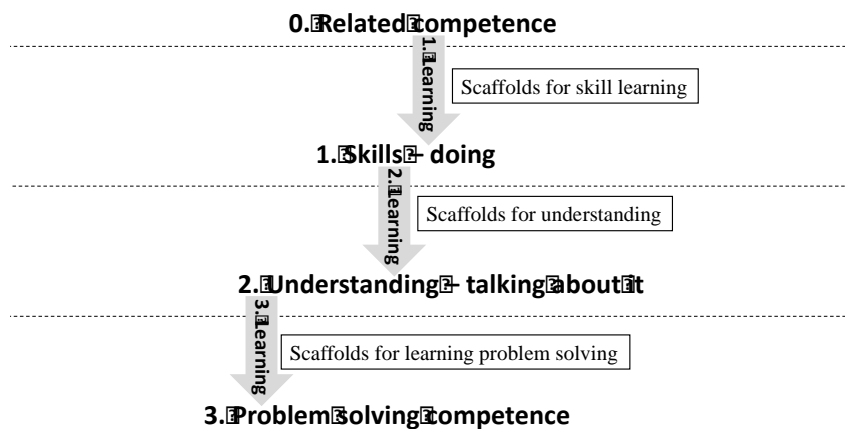


Figure 1. Three steps for learning IT use. Scaffolds are any means supporting the learning.

Often, we learn through using a computer and observing what happens when we carry out operations. Sometimes, we look for help at the Internet, we ask a friend, we attend user training, we look up in user documentation or we read tool-tips at the user interface. Such additional means for learning are called *scaffolds*, since they help us building new competence. Part I will introduce scaffolds for each learning step. Part II will present how we combine scaffolds in user training and user interfaces.

The subsequent chapters in Part I will also divide the learning processes and competence levels into subcategories.

Chapter 2. IT skills

The learning objective of this chapter is to be able to create scaffolds for learning IT skills.

Users of technology need skills for applying IT to meet their needs. A *skill* is a practical competence, indicating that a skilled person knows how to do it. Therefore, skills are also called know-how. IT know-how will normally include some bodily skills like hand-eye coordination for mouse movement or the ability to push the keys on the phone when writing a text message.

2.1. Learning IT skills

Learning IT skills takes place in two different ways; *imitation* for learning new skills through following instructions, and *repetition* for improving performance and remembering the skill.

Repetition

Repetition is a learning process for strengthening an already existing skill. Repeating behaviour many times may lead to the learner being able to do it without conscious awareness. We say that the skill has become automated. Speaking, walking and running are examples of skills that most people would learn to the automated level. We normally don't pay attention to how to move the leg forward when walking. Correspondingly, typing on a keyboard becomes automated after long practice, such that we can write words without considering which fingers to move in order to hit particular buttons. Reaching the automated level has been shown to influence positively the continued use of information systems (Limayem et al., 2007, Polites and Karahanna, 2012).

This book will not address practicing the bodily skills like pushing the buttons on a QWERTY keyboard. The interested reader can find specific textbooks on such skills, e.g. (Barnes, 1890). Rather, we will concentrate on the cognitive component; the know-how of operating software and IT gadgets.

Imitation and instructions

While one can become an efficient user of IT by repeating a sequence of operations, repetition does not necessarily extend our repertoire of skills. A way of learning new skills is

Imitation—Gabriela and Ali.

Assume that Ali is watching Gabriela while she is pressing CTRL/C and CTRL/V for copy and paste. Ali has been doing this operation by means of menu choices quite often, but he has not seen the shortcut before. He observes that it is efficient and therefore starts using the key combination himself. When Ali continues to use the key combination instead of the menus, a relatively stable change of his competence has taken place, meaning that he has learnt this skill.

*imitating*¹ others' behaviour. Those we imitate constitute the scaffold for learning the new skill.

It has for long been recognised that imitating others is an important way of learning IT use (Bannon, 1986). In general, similarity between the settings during imitation and repeating ease learning (Ormrod, 1995). Imitation has therefore a strong advantage as a trigger for learning, since what the learner observes is exactly the behaviour to be repeated.

Ali was motivated to learn the copy and paste keyboard shortcut through observing that it saved time. Motivation is a strong factor for learning. Switching from mouse selection in menus to keyboard shortcuts has long been known as an efficient way of speeding up interaction (Cockburn et al., 2015). But if Ali is happy with his menu choice for copy and paste and does not see any advantage in keystrokes, he will not bother trying, and hence he will not learn it.

User training is often carried out as follows. Each student has a computer, and the trainer uses a video projector. The trainer *instructs* which keys to push on the computer by demonstrating

Psychology – Cognitive load

Human long-term memory has an enormous capacity. Operations which we have repeated a number of times, are learnt in the sense that we do not have to pay attention to them again and these operations are stored in long-term memory. This applies to bodily skills like walking a staircase or swimming as well as cognitive skills like copy-paste and interpreting an e-mail address. There is in practice no limit to how much we can learn and store in long-term memory.

Our ability to process information is severely limited, however. As a rule of thumb, we can process 7 ± 2 items, which require our attention. For example, most people will remember a 6-digit phone number when reading it in the phone book and typing it on their phone, but they will have trouble doing the same with an 8 digit number. Splitting it up into two chunks of 4 digits may help. Short-term memory is limiting our ability to learn in the sense that we can only pay attention to and learn a small number of things at a time.

Learning more complex matters will therefore have to be broken up into smaller pieces. After having learnt one operation such that we do not have to pay attention to it any longer, we can move on to learning the next one. For example, a novice spreadsheet user can learn to set up a formula that has other cells as arguments. Thereafter, the user can learn copying and pasting the formula. After having observed and exploited that cell references have been changed when pasting, this phenomenon will most likely be stored in long-term memory. Thereafter, the user can learn the distinction between absolute and relative referencing.

¹ In behaviourist learning literature, this way of learning is called 'modeling,' but since the word 'model' is used for other purposes within this book, we use the term 'imitation' for this learning process.

it and projecting the screen. The learners try to remember the keys and repeat the operations carried out. Due to the low capacity of our short-term memory, the learners often forget the steps.

A similar situation happens when an IT support person verbally instructs a user about which buttons to push in order to solve a problem. The next time the problem occurs, the user has forgotten the steps. Since a colleague or a trainer normally cannot stay around for repeated demonstrations until every learner has acquired the skill, documents or videos might be helpful.

2.2. *Instruction sheets – scaffolds for imitation*

Scaffolds for imitation are called *instructions*. Instructions consist of guidelines that lead the user step by step through a procedure, demonstrating how to carry it out. The trainer demonstrating software with a projector is providing live instructions. An example of written instructions is provided in Figure 2. A document containing instructions is called an *instruction sheet*.




1. Click on the **Apps** button.
 2. Click on the **Data Visualizer** button.
- 
- The screenshot shows a user interface for 'd/his 2' with a user profile 'Anniken Jansson'. A search bar is present. Below the search bar, there are several buttons: 'Dashboard', 'Print Table', and 'Data Visualizer'. Red arrows labeled '1' and '2' point to the 'Apps' button in the top right corner and the 'Data Visualizer' button respectively.
3. Choose a line chart by clicking the  button.
 4. Mark the wanted indicator or data element.
 5. Use the single arrow  to select. For selecting all indicators in the group, use the double arrow.
 6. ...

Figure 2. The first steps in an instruction sheet for making a line chart in a business system.

Effective instructions should follow some principles that are explained below.

Recognizable

For imitating instruction sheets, users need to recognise specific places in the IT application when reading them, thus including screenshots is often necessary. Screenshots are also needed for pointing to exactly where to push a button or tick a box. In Figure 2, the larger screenshot illustrates steps 1-2, while screenshots of buttons are included in steps 3 and 5.

The learner also needs to recognize that the explanatory text and figures are not parts of the screen. This difference is achieved in Figure 2 by placing the numbers outside the screenshot and drawing arrows at an angle with the screen items.

Proximity

When perceiving the world, people group together stimuli which are located closely together, which are similar, and which constitute shapes which we expect (Ormrod, 2012). This

principle is called *proximity*. For example, in Figure 3, the textual instructions in the left case is close to the buttons to be pushed and linked with an arrow, while in the right case, there is no such proximity. Short distance between button and text means that the learner can keep the attention to the point of action, while in the right case; the attention has to be split between screen shot and textual explanation.

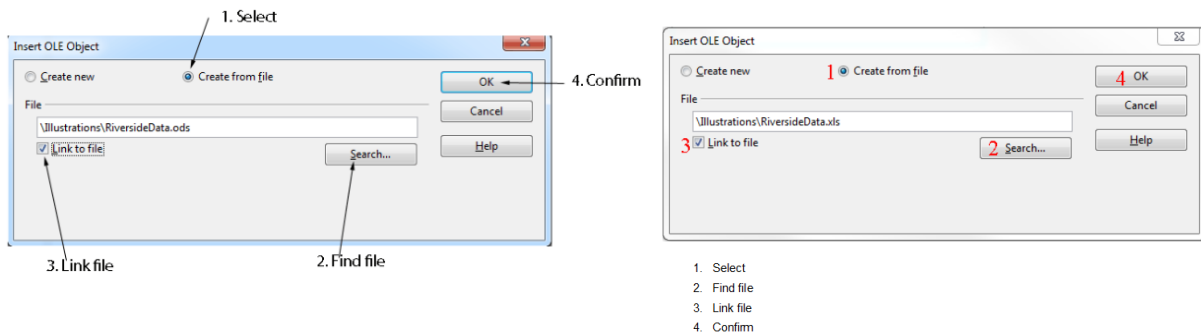


Figure 3. Instructions with text close to action objects (left) and coded with sequence numbers (right).

A summary of many studies shows that proximity gives learning gains, and that when the material to be learnt is complex, the gain is quite substantial (Ginns, 2006).

Sequential

Operating IT involves a sequential series of actions, hence instructions need to specify the sequence. Video, audio and text are sequential modes of expression, while illustrations are not. Illustrations therefore need specific sequencing, for example as in Figure 2.

Direction

Users often believe that the computer can carry out a certain operation, but they do not know where to locate it in the interface. This means that they have an understanding of the functionality, but lack the skill to trigger it. Instructions need to provide directions to the location in the user interface where the functionality is found, we would say that instructions should be *directive*.

Navigation: Defne.

Defne knew that the switch for WiFi reception used to be located in the top level of the settings menu on her phone. After installing a new version of the operating system, she discovers that it is not located there any longer, such that she is in need of navigation.

Defne was able to navigate to the WiFi switch by searching the web, where someone had written:

Where is WiFi service located?

Settings > Connections > Network Services
... listed here

Knowing some facts might ease navigation. For instance, navigating in a new text processor is easier if a user has learnt trivial facts like “Times New Roman and Arial are fonts.” Then the user would recognise these words and deduce that this is the location for changing fonts.

Completeness and Feedback

Reinforcement from the environment that the action carried out is successful is an important factor for learning (Ormrod, 2012). Digital devices often provide immediate feedback on the result of an operation. If the result is what the user wanted, the feedback is reinforcing learning. Sometimes, the system does not provide appropriate feedback, hence learning is not reinforced. The user might therefore have to do additional operations to control the output. Instruction no.8 in Figure 4 is for checking that the operation has yielded the desired result. We will say that instructions which include such check points *promote feedback*.

8. **Click Update.** A chart like this should now appear:



Figure 4. An instruction for checking the result.

Keeping the cognitive load low is also a reason for breaking a long sequence of instructions into shorter sections with an observable end state. If the computer does not produce an observable output after a section, checks like instruction no.8 should be included at least at the end of a section of instructions. Then the user can concentrate on learning one section at a time. The longer sequence can thereafter be learnt by joining the sections. For instance, instructions for uploading a file with meta-data to a cloud service could be divided into the sections shown in Figure 5.

- A. Browse to find and select the file.
The file name should now appear in the Upload field.
- B. Upload.
The file should now be visible in the cloud service.
- C. Add meta-data.
Meta-data should now be visible in the cloud service.

Figure 5. Three sections (A-C) for an instruction sheet for uploading files. The user can check the result after each section. Each section should be expanded with more steps and screenshots.

Instructions should include all necessary steps to reach the result which the user wants. If some steps have been left out, for example *Save* or pushing *OK*, the user may believe that the operation is complete without having reached the final goal. If the instructions promote feedback as the last step, the user might be able to check whether the final output is as wanted.

Short

Few users read manuals (Novick et al., 2007), and long texts are particularly unlikely to be read by most users, since they are more interested in doing than in reading (Carroll, 1990). Instructions should therefore be short. The instructions in Figure 2 are as short as possible.

The example in Figure 6 has short and precise instructions for how to trigger the selection of fields. However, the instructions to the left for how to fill the fields are too wordy and are not broken up into steps.

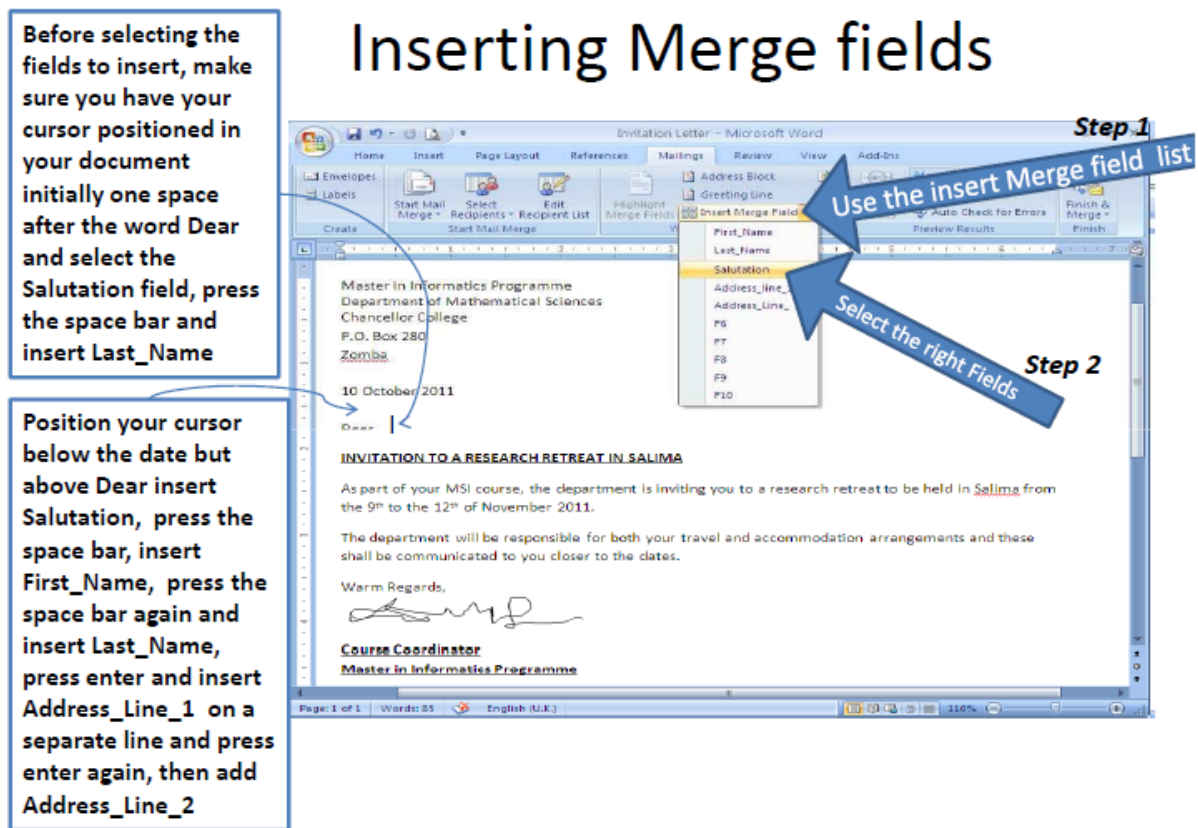


Figure 6. Instructions for mail merge (MS Office 2007)

The instructions in Figure 6 also illustrate another common problem with screenshots. We are often interested in small portions of a large area. If the screenshot includes the whole window, the details in the area of interest become tiny and difficult to see and mark up with arrows. A better solution in Figure 6 could be to extract the name and address area of the letter and blow it up, so that the reader can easily spot the exact position where to type Last_Name.

Choice of example will also influence the complexity of the sheets. The example should illustrate the normal execution of the operation, without including any other, disturbing data.

Level of detail

Studies reveal that users are often dissatisfied with the instructions provided for the software they use. Typical reasons are that IT instructions are too basic, but also too difficult to imitate (Novick and Ward, 2006, Smart et al., 2001). The observation that some documentation is too basic may come from the fact that it is intended for the novice, and then the more proficient user finds it too detailed. The opposite may be the case when the documentation is too difficult. Personally, I have too often experienced that I find some seemingly useful documentation on the web, but after trying to follow it, I realize that my software is not identical to the one in the instructions. It is often a matter of the documentation missing information about the software version.

The instructions in Figure 2 are intended for users who have some skills in navigating in the menus. Novice users might have had problems with finding the indicators and data elements referred to in point 4 in the example, since the instructions do not include a screenshot indicating where these items are located.

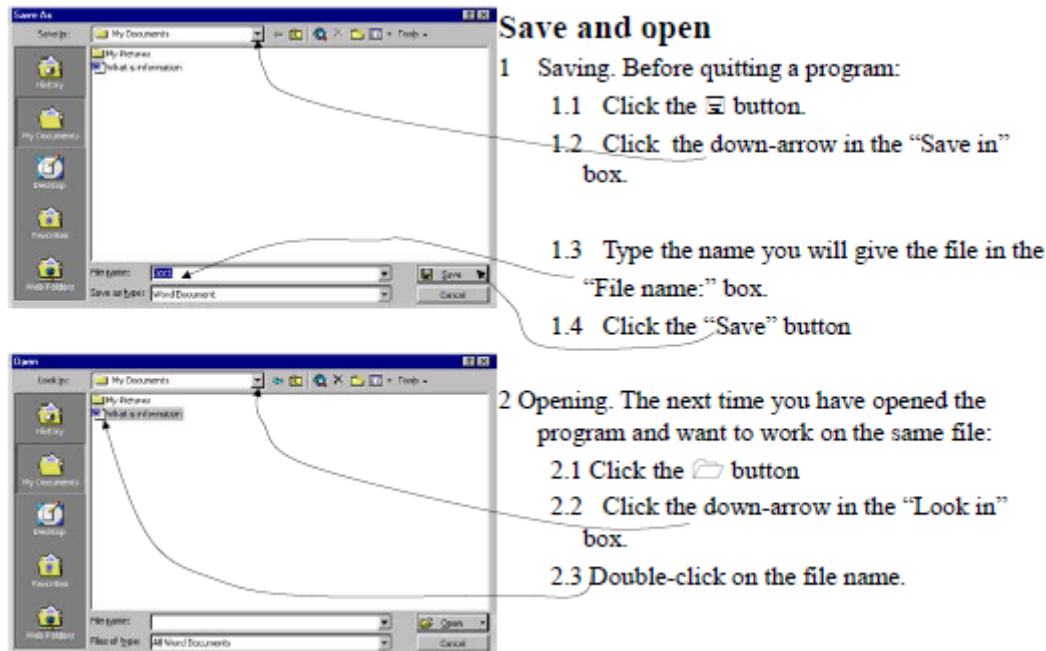




Figure 7. Instruction sheet for novices.

The instructions in Figure 7 are detailed concerning a basic operation, so it fits novices. A user who has acquired some IT skills would find these instructions too basic. One might argue that more advanced users would not look up how to save a file. However, if one wanted to know how to save with a special file format, searched for how to do this and hit the instructions in Figure 7, then the instructions would turn out as too basic. On the other side, if the novice does not know where to find the  and  buttons, the instructions would be too difficult.

In an experiment with elderly novice users, the learning effects of annotated screen shots, screen elements embedded in text and text only instructions were compared (Kehoe et al., 2009). The full screen shots eased the imitation, while the text versions had better effects on the learners' remembrance of the skill. This may be due to that following the screenshots requires less cognitive processing by the learners, while the additional effort needed to follow the text style instructions had positive impact on remembering. The intermediate version, screen elements embedded in text (as in point 1.1 in Figure 7), yielded an intermediate result. Consequently, annotated screen shots are useful for introductory imitation. To support the learners' memory, the users should have continuous access to these instruction sheets.

Hence, instructions have no obvious level of detail, which fits all users. The computer scientist may launch the idea that the software should track the users' skill level and present instructions accordingly. However, even keeping one version of user documentation correct

and up to date seems to be too demanding for many IT vendors and in-house software systems, so managing a set of different levels could easily lead to more chaos than improvement. A more realistic approach is to make some simple assertions about which functionality that will be used at different skill levels, and adjust the instructions accordingly. Table 1 provides recommendations for adjusting the instructions to the skill level.

Table 1. Skill levels and corresponding instruction design.

Skill level	Operations	Presentation
Novice	Any basic	Screenshot and every detail.
Ordinary	Any basic	Brief mention
	Menu selection for new operation	Textual navigation from main window to location. E.g., Insert → Object → OLE Object
	Unknown window Several operations	Screenshot for navigation Sequence
Advanced	Any ordinary or basic	Brief mention
	Menu selection for new operation.	Textual navigation from appropriate point to location.
	Unknown window Several operations	Screenshot for navigation Sequence

Missing the skill level is not the only trouble that users report concerning instruction sheets. Another complaint is that user documentation is out of date (Novick and Ward, 2006). Outdated material was abundant when manuals were printed, and new software versions were distributed. Publishing instruction sheets on the web eases updating.

Terminology

In order to avoid technical jargon, which users don't know, "Speak the users' language" is a slogan often found in in textbooks for human computer interaction. This is easier said than done, because users do not speak with one tongue. When designers invent a term for a functionality, only one or two in ten users would use the same term, while the remaining ones would use 5-8 other words (Furnas et al., 1987). This discrepancy in terminology causes problems when looking for the interface object which will trigger the wanted functionality, and messages popping up on the screen may be written with unintelligible terms for most users. Instruction sheets should therefore use a variety of terminology when referring to the computer operations, such that a user may find some terms that they recognise.

Tools for creating instruction sheets

When creating instructions that include screenshots, one normally needs copying a portion of the screen and thereafter adding some graphics and text. The Print Screen key copies the whole screen, so for selecting an area, the image has to be cropped to the desired size by means of a software tool that can handle raster graphics.

Windows 2007 and later has a program called Snipping Tool that produces a copy of an area which the user can select. Ubuntu Linux has the option Applications → Accessories → Take Screenshot → Grab a selected area. In Mac OS X, the Grab app or Screenshot have corresponding functionality. For finding out how to make screenshots of mobile phones,

search the web with the terms Screenshot and phone name, or use a phone emulator on the computer.

Instruction sheets can be presented in many media, including in-line help that appears in the software, a slide, a web page, a text document. If only one form of publication is relevant, the instructions should be made with appropriate software for the medium, for example, Impress, Keynote, PowerPoint or Prezi for slides. If the instructions are to be published in several media, the professional approach would be to store the instructions in a XML format, from which any type of publication can be extracted. DocBook is such a format, intended for writing technical documentation.

2.3. *Instruction videos*

The previous section outlined qualities of instructions; recognisable, sequential, complete, promote feedback, short, directive and with varied terminology. These properties hold for any medium. The contents and structure of an instruction video should therefore be similar to the sheet.

Generally, the written text in instruction sheets would be presented orally in a video, and the static screenshots would be replaced by a dynamic screen capture, showing mouse movements and characters being typed. Videos may also need some graphics like arrows or highlighting for drawing attention to specific parts of a window.

In videos, proximity is not just a matter of spatial layout, but also of relatedness in time. An event, which is following directly by another one is close and stored in short-term memory, while a couple of events later, it may have been forgotten.

Examples of video instructions are abundant on the web. Using the search terms

“spreadsheet formula video”

produces millions of results.

Some videos have replaced written text with sound, while others have kept the written instructions and have no sound.

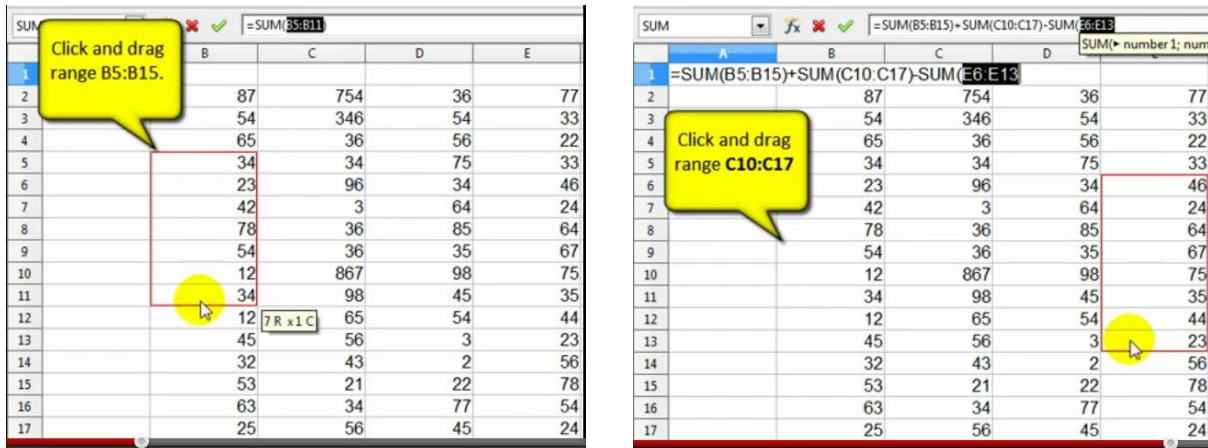


Figure 8. The callout points to the rectangle at left, but not at right (COL CCNC, 2010).

The yellow callout in the silent video (COL CCNC, 2010) points to the rectangle that is dragged, see Figure 8, left part. Also, the yellow colour appears both in the callout and at the cursor position. This supports the association between the callout and the rectangle in both location and similarity, so it supports proximity in two ways. The callout in the right part of Figure 8 is located far from the rectangle, so in this case, the association is only through the similarity in colour. This example is from a video, but proximity can be achieved in the same way also in static illustrations.

The use of screenshots in the video enables recognising the software in these videos. However, there are a large number of cells filled with data in the examples, so the learner needs to be able to disregard the cells that are irrelevant for the insertion of formulas. Excessive amounts of data or of interface details clutters the picture and makes it unnecessary hard to recognise the essentials.

Sequence is guaranteed, since the video is a sequential medium. Feedback is promoted due to the viewer can see the result in the end. In addition, the videos with screenshots direct the user to the appropriate place in the user interface.

Briefness applies to videos as well. Users are more likely to watch a short video (1-3 minutes) to its end than a one lasting for more than 6 minutes (Guo et al., 2014).

Tools for creating instruction videos

Producing a video can be done in three steps:

1. Recording the screen and voice by means of a screen recording and video production software. The recording yields a series of frames, as illustrated in Figure 9. The series of frames is stored in the format of the software package.

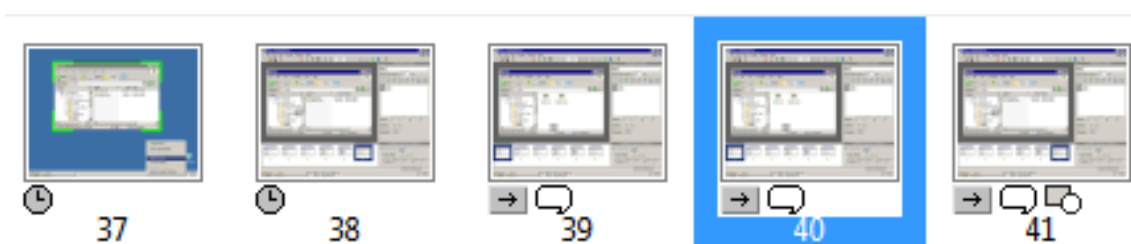


Figure 9. A series of frames for video production. Screen capture from Wink (Kumar, 2010).

2. Editing the frames. Frames can be deleted and added from other recordings. Also graphics can be added in this step.
3. Rendering. The software produces a video file, which could be
 - a. Animated vector graphics – Flash files (.swf) can be played with Adobe Flash Player.
 - b. Compressed video – MPEG-4. Can be viewed with video players.

There are several softwares which can do the whole or parts of this process. Three examples are:

- Adobe Captivate is a commercial product with extensive functionality (Adobe, 2012). It runs on all platforms.
- KRUT is freeware and runs on all operating systems (Östby, 2012). However, it skips the editing step.
- Wink is freeware and can do the steps 1-3 above (Kumar, 2010). It runs on Linux and Windows.

2.4. Training for skills

Training people at work by making them imitate an instructor has led to substantial skill learning for a large variety of trades, as seen in a summary of 117 studies (Taylor et al., 2005). Even so, the effects on job behaviour were moderate, but stable over time. Performance at work was improved when the trainer not only demonstrated how to do things, but also how **not** to behave. Making the learners use some of their own cases during training also helped.

It was noted above that learners quickly forget long series of operations. Written or video instructions may aid users when back at work. While users seldom read manuals, they are twice as likely to look up in training material (Novick et al., 2009). Trainers should therefore hand out instruction sheets or videos to the learners instead of instructing by means of a projector (Herskin, 2006). Then the users will have training material to consult after the training. Following an instruction sheet instead of the trainer at the projector also eases most learners' practice during the course, since they can follow the instructions at their own pace. During trainer instructions, some learners work slower than the trainer, such that they are left behind. One might object to the video on the same grounds; that the learner cannot follow its pace. However, videos can be paused and replayed indefinitely, in contrast to the trainer in front of a class.

Following instruction sheets may be beneficial when introducing a new program or functionality. However, research indicates that after a short time, users prefer working on their own and they also seem to learn more quickly in that way (Carroll et al., 1985).

Training in courses by means of instruction sheets also relieve the trainer from running around in the computer lab to help out those who forgot the instructions (Herskin, 2006).

Nevertheless, some learners with insufficient digital literacy do not imitate the instruction sheets but ask their fellow students for help instead (Hadjerrouit, 2008). This might be a symptom of written instructions being more abstract than live ones, such that novices should also imitate the trainer with projector or possibly view a video, which is more concrete than a written sheet.

Instruction sheets and videos need to be stored where users can find them when needed. Searching Google with “guide windows” yields more than one billion hits, and there are more than a million instruction videos for Linux on the web. The users’ challenge is to find the right one. Research has reported that users have trouble finding instruction sheets and other documentation when needed (Novick and Ward, 2006). This challenge will be addressed in Section 5.4.

When introducing new business specific software in an organisation, people need to learn the skills for using it. How to organise for user learning will be addressed in Chapter 11 and later. In any case, instructions should be produced and distributed. Knowing that users have trouble searching for and finding relevant instructions, the best option is to place the instructions such that no search is necessary. A solution is to include instructions in the user interface of the software, so called context-sensitive or in-line help (Shneiderman and Plaisant, 2010); this topic will be elaborated in Chapter 6.

2.5. Assessing IT skills

Upon completing user training, the trainer may want to know whether the users have learnt the skills aimed at. Since skills are demonstrated by doing and not by saying, tests of skills should be through practical exercises. Exercises like

- Summarise both rows and columns in the spreadsheet, and
- Use styles consistently in the document

are therefore appropriate for testing IT skills. The trainer needs to observe the performance of the learners on the IT device to judge whether they are at the wanted skill level. Alternatively, viewing the result produced by the trainees may be done, but such inspection does not capture the mistakes that the learners might have done on their way. The following question

- How do you summarise both rows and columns in a spreadsheet?

calls for an oral answer and not a demonstration of practical skills. The following question is even further from testing skills:

- What is a style in a text processor?

This question does not concern know-how at all, but rather knowing-that or understanding.

A thorough discussion on testing IT skills and understanding will be provided in Section 10.2.

2.6. Summary

IT skills are strengthened through *repetition*. Learning new skills is eased through *imitating* scaffolds in the form of *instructions* provided by trainers, instruction sheets or videos. Written instructions are more abstract than a trainer demonstrating; hence personal demonstrations may be needed for learners with low general IT competence. The essential processes for learning IT skills are summarised in Figure 10.

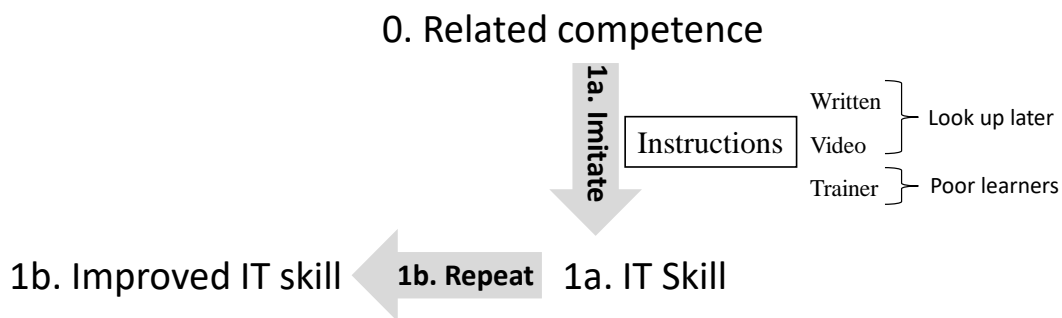


Figure 10. The processes for learning IT skills.

Instructions should have the qualities listed in Table 2.

Table 2. Qualities of instructions

Qualities of instructions	Explanations
Recognisable	Users recognise items in the user interface with those in the instruction.
Proximity	When interpreting instructions, users group together items which are located closely together, which are similar, and which constitute shapes that the users expect.
Sequential	The steps in instructions should follow each other in a sequence.
Direction	Instructions should guide the user to the relevant location in the user interface.
Completeness	All steps need to be included.
Promote feedback	The instructions should enable the user to check whether the wanted result is achieved.
Short	Users want to do, not read. Thus, instructions should be as short as possible.
Level of detail	The amount of detail in instructions should be tailored to the expected group of learners.
Terminology	People use different terms for the same topic. Hence, instructions should include a variety of expressions.

1. Provide users with instruction sheets or videos, also during training.

Chapter 3. Learning business fit

The learning objective of this chapter is to be able to create scaffolds for understanding the usefulness of IT for own tasks and for the business.

Prior to addressing learning, research on what makes people actually use IT for their business will be presented.

3.1. Usefulness

One of the few well-documented connections concerning use of IT in organisations is the Technology Acceptance Model (TAM). In its original form, it says that the perceived usefulness of a technology is the strongest factor concerning whether the technology will be used, while its ease of use and learning is of less importance (Davis, 1989). TAM predicts that if computer software is experienced as useful by the users, they will use it, even if they have to put effort into learning it. On the opposite side, a system which is easy to learn and use will not be used if the users do not experience that it is useful for their tasks.

The model is illustrated in Figure 11, where the bold arrow indicates influence that is stronger than the other one.

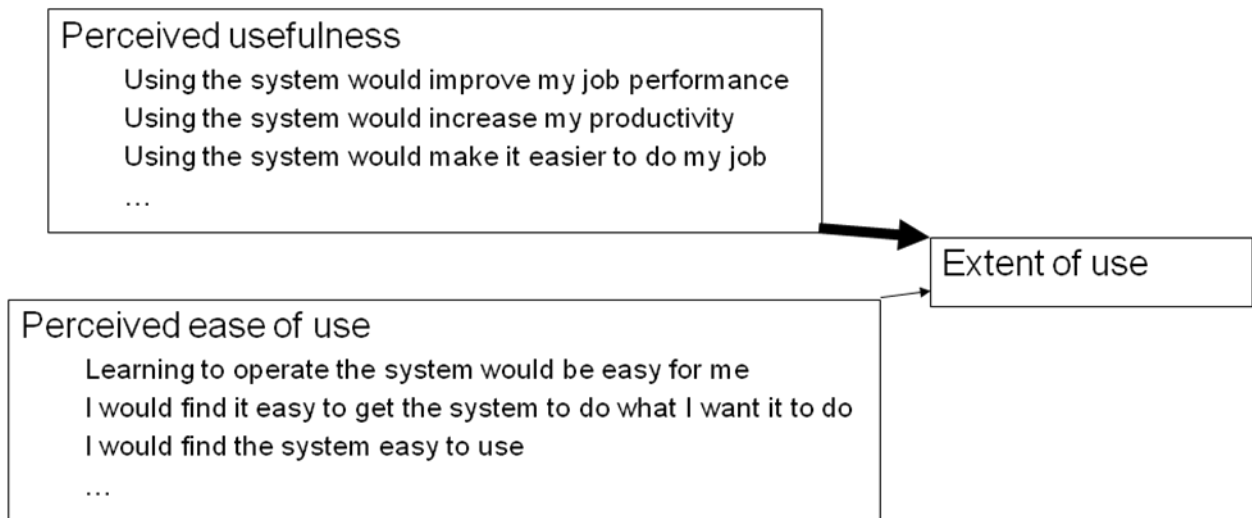


Figure 11. The Technology Acceptance Model, original version, adapted from (Davis, 1989).

An example: At a time when computers were not everywhere, a hospital installed a computerised encyclopaedia for nurses in one ward, where they could find information on care procedures, medical explanations, guidelines, etc. The nurses had previously experienced that their questions were not always answered in a polite way, and that looking ignorant in front of superiors was a bad experience. Therefore, they quickly adopted the system to avoid having to ask doctors or administrators for help. The system only had one terminal, and after a while, this terminal was moved to another ward five minutes walk away. Despite this extra time, they continued using it. In order to reduce disturbances, they organised a buddy system,

so that one nurse collected the questions and walked over to the other ward, while the others took over her tasks during the half hour needed. Thus, they took on additional effort in order to achieve the usefulness that they had experienced.

Measuring the impact of use of IT systems in organisations in general has been impossible. Normally, you cannot isolate the costs of technology implementation, and you cannot isolate their effects. Expenses are interwoven with the costs of learning and changing work processes, and correspondingly, the products and services produced by an organisation depends on a package of factors, including competence, infrastructure, and the market. TAM therefore measures the time that the technology is being used and adapts this as the indicator for the degree of success. This kind of measurement has over the years become a standard for measuring technology acceptance and success.

Later on, TAM has been refined with more factors, and a combined model looks like Figure 12.

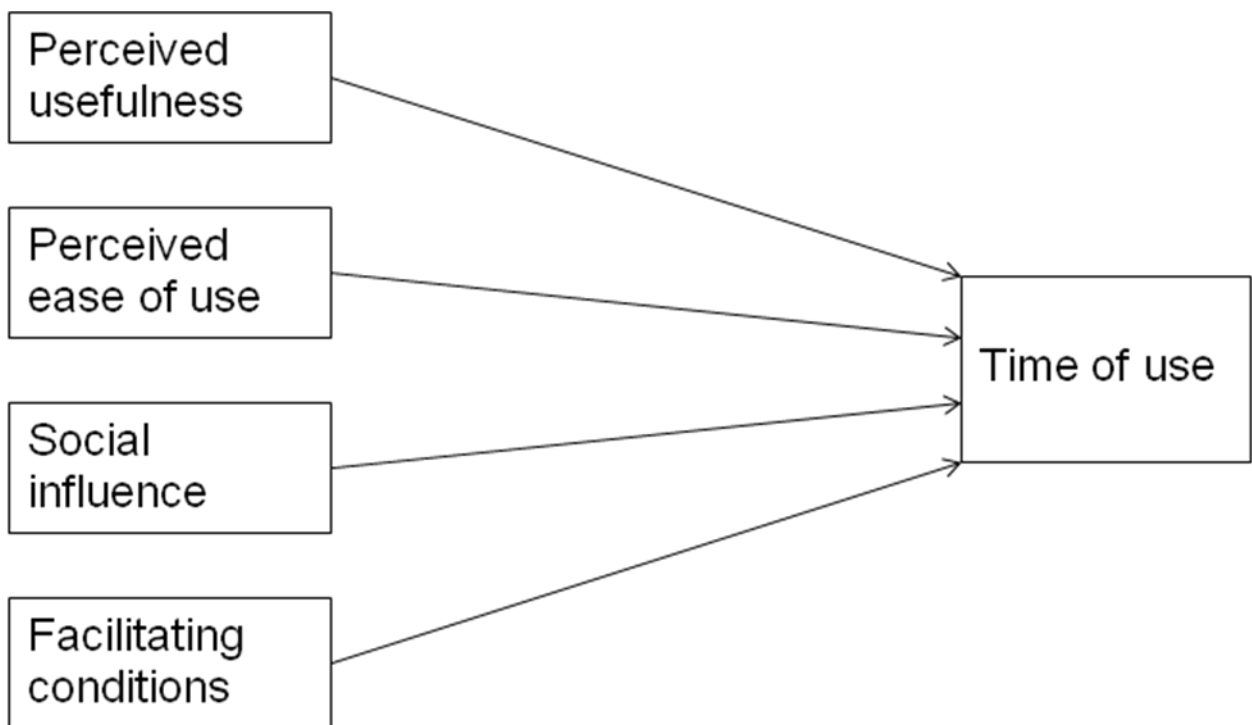


Figure 12. The revised Technology Acceptance Model, adapted from (Venkatesh et al., 2003).

When colleagues use the system or your boss tells you to use it, the social influence is increased. Facilitating conditions concern accessibility, including network connection, electricity, printers, etc.

In the case of teaching IT in schools, the students may find it mandatory to learn the application because of trainer pressure. In TAM terms, the students are under strong social influence, such that usefulness and ease of use are less relevant. For later, voluntary use of the same software, the usefulness which the students have perceived will be prominent for their decision to apply it, according to TAM.

When evaluating proposed IT systems, the model tells us about four factors to consider. Beware that it is the prospective users' opinions which matter. If outside consultants do not see the point in a software package, while the users do so, the system will probably persist. On the other hand, if the consultant thinks that some information produced by the system will be very valuable for the organisation, while those working there do not share that opinion, we cannot expect that they will take the effort of learning and using a new system.

In addition to the four factors, which seem to be relatively stable, other factors will moderate the picture. For a young man, the usefulness will be more important than for others, while for an elderly lady with little IT experience, ease of learning will count more than for others. Such moderating factors may depend on the local culture, and the studies behind TAM have mainly been carried out in North America.

In summary, learners who have not understood the usefulness of a specific IT will be less likely to learn its operation. Teaching usefulness should therefore precede teaching skills.

3.2. Understanding usefulness of IT in own tasks

Sein et. al. (1998) and Coulson et. al. (2003) have proposed levels of knowledge of software use, where an important distinction is made between competence of the software and of its use. The previous literature considers that a basic competence of fitting IT to business is knowing how to use IT in one's work, and a more advanced competence is to see what IT does for the organisation (Sein et al., 1999, Coulson et al., 2003). The Committee on Information Technology Literacy (1999) also considers the societal impact as part of IT competence. One dimension of competence on IT use is therefore its scope, extending from the individual through the organisation and into the societal level.

Let us first consider Kirsten, who is requested to tell about IT impacts in her workplace. Her reply is only about which IT tools she is using for two activities. She is not showing any motivation for using the IT tools in the sense that she can point explicitly to their usefulness in the TAM sense, for example by comparing these tools with other solutions. She is not telling anything about other consequences of IT for her work or the organisation. Although she is able to tell a little bit about how IT fits her activities, she seems to have skills for using IT without being at the Understanding level.

Skills for use in business—Kirsten:

When asked about the impact of IT in her workplace, Kirsten replies:

When I inform important customers about new services and remind them about appointments, I find the customer's details in our computer system and then write letters in Word. Thereafter I send the letters off in Thunderbird.

Understanding IT in own tasks is the ability to explain the relationship between IT and one's own activities. This includes understanding of why an application is useful or not.

Leonard is clearly explaining the usefulness of the computer system for his work, and he relates the data in the system with the status of his customers. He seems to understand the role

of IT in his own activities. Since his own work tasks are closer to his experience of using IT than is the whole business, Leonard would probably understand what the IT means to himself before he understands the larger picture of IT in the whole organisation.

Understanding IT in own task—Leonard:

Since I use the computer system for keeping track of which messages the customers have received, I know that they are up to date on our services. In that sense, the computer system is very useful to me. It is also reminding them about their appointments with me.

In a study of learning a software by means of web-based material (Gravill and Compeau, 2008), the most frequent learning strategy reported by the users was:

I considered whether the material would allow me to accomplish specific work tasks.

Hence, during learning, users seem to aim at understanding how IT can be useful in own activities.

3.3. Minimal Manuals

Scaffolds for learning about IT in own tasks would couple the task with IT functionality, and we will call this a *minimal manuals*. The term was coined by Carroll (1990), who found out that a scaffold which combined tasks with instructions was an effective documentation for users to learn the software.

An example of a minimal manual from a hotel system is shown in Figure 13. The business task of making a reservation differs from the reservation functionality in the system, because the latter assumes that the customer already exists. Thus some instructions were necessary to make a complete task oriented documentation. Recognisability is low in these instructions, assuming that the hotel staff is already reasonably familiar with the system.

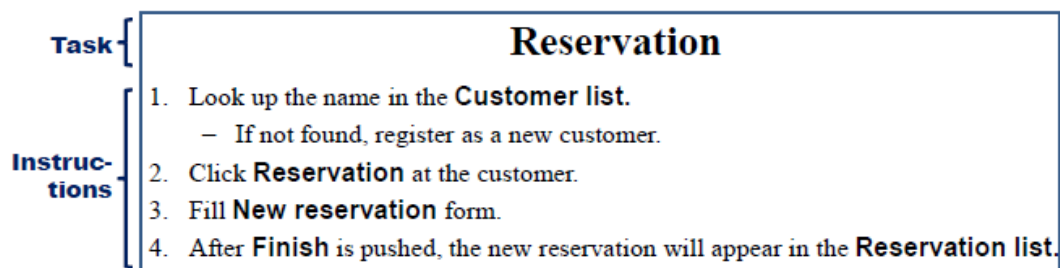


Figure 13. A Minimal Manual consists of two parts; task description and instructions.

Instructions only will not suffice to link work task to computer operations. Hence, to motivate users through understanding the usefulness of IT in their own activities, all scaffolds should at least be minimal manuals coupling task with instructions. The term ‘minimal’ hints to this type of documentation being the smallest sufficient scaffold for learning IT use.

An information system may have several types of users carrying out a wide variety of tasks. For instance, sales staff in an internet shop enters new products and prices in the system, while a customer browses the catalogue, picks goods and manages an account. The sales staff

may need to learn many details on how to describe and categorise a product and how to determine prices. They will need all this competence on the information in the system in addition to the necessary IT competence for operating it. For learning this, some scaffolds may be needed. On the other side, in order to get customers, the system should require as little learning as possible for them. Customers who need to read an additional page of documentation might well give up the shopping.

For understanding IT in own activities, users need documentation according to their tasks. An international standard for user documentation suggests organising users in a hierarchy according to their needs (ISO/IEC, 2008). The web shop could have a hierarchy as shown in Figure 14.

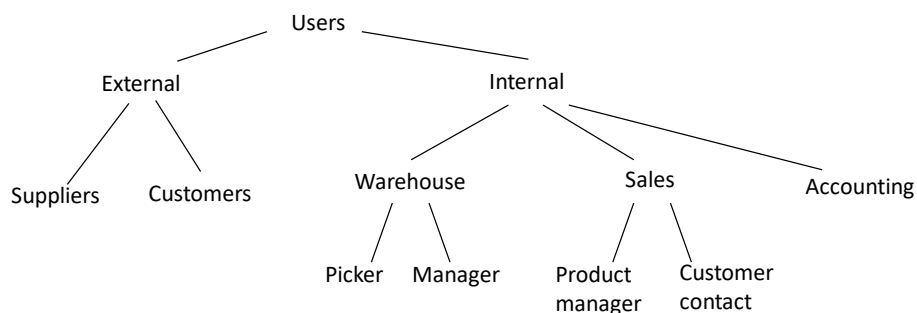


Figure 14. Hierarchy of users according to documentation needs.

3.4. Understanding IT in business

The next type of understanding to be achieved is the ability to explain how IT fits business as well as consequences of IT for own tasks, the organisation and the society. For short, we call this *Understanding IT in business*.

Astrud’s explanation is telling little about her skills, but it demonstrates that she has understood consequences for her own work, for colleagues in other departments, for management, and for customers. She demonstrates an understanding of the current situation and compares it to the previous system. Her enthusiasm of the usefulness of the corporate database demonstrates a clear motivation for using the system.

Understanding IT in business—Astrud:

The corporate database means a lot to my work and to the organisation as a whole. Now I enter all incoming mail in the system, and if it is on paper, I scan it. This means that I can search everything that is there, and also that people in the claims payment department has the information at once. Delays due to waiting for papers to be transferred or finding the case in the archive have been eliminated. Besides, it gives us the up to date information on how we are doing, so that there is no longer any argument with management on productivity measurements. I see this as a win-win-win situation for us, management and customers.

Eddy can express how he uses a messaging service to communicate with his friends, demonstrating an understanding of IT in his own task. Lena, on the other hand, also knows how the communication task is integrated in a larger commercial enterprise and governmental

surveillance system. Hence, she understands how the IT is embedded in business and societal structures. After their conversation, Eddy might have picked up some of Lena's understanding of IT in business too.

The steps to understanding the fit of IT in business are illustrated in Figure 15.

Understanding task versus business—Eddy and Lena:

Eddy: *You should try the newest social media FreshBlog. There you can send messages in full privacy.*

Lena: *You know, everything you write is captured by their marketing business. Haven't you noticed the ads that relate directly to the contents of your messages? Your friends will get them too. And be sure that the National Security Bureau already has picked up who are your friends' friends.*

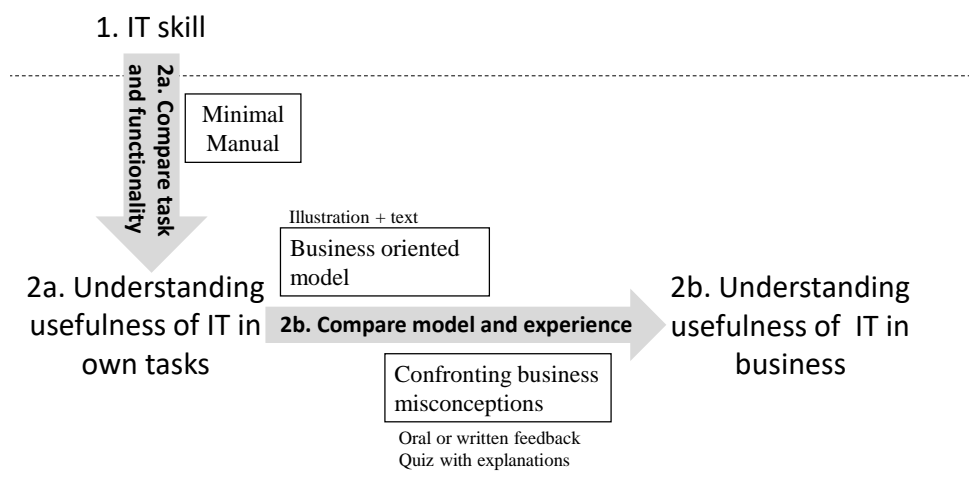


Figure 15. Learning Business fit. Two types of scaffolds help understanding usefulness of IT in business.

3.5. Business oriented models

Research points to that when we work together with tasks which depend on each other, training is more important for our adoption of IT and information systems than when adopting a single user application (Sharma and Yetton, 2007, Reynolds, 2010). Further, users have a harder time learning business specific database systems like enterprise resource planning and supply chain management than communication systems like e-mail and blogs, even if both types concern the whole organisation (Bagayogo et al., 2014). The communication systems do not have the high degree of formalisation as the business systems.

A study of users' perceptions of manuals found that they want documentation to include more than how to carry out a specific task by means of a software package (Scott, 2006). They also want the manual to include how the activities that the software supports relate to other activities carried out by themselves or colleagues.

Supporting understanding of how IT fits business and consequences of IT for own tasks, the organisation and the society may include a large variety of documentation. For example, the Arab spring demonstrated the significance of social media for societal change.

In large organisations, there may be long, interconnected chains of information processing and exchange. In a factory for mass production, many staff groups access the same database, including production workers, engineers, accountants, sales, purchasers, managers and administrators. Purchasers need to understand the accountants' requirements for specific details on invoices. Production workers need to register both use of raw material and products made for a batch, such that both purchasers and sales are kept informed. Figure 16 is a business-oriented model displaying data flow in an organisation.

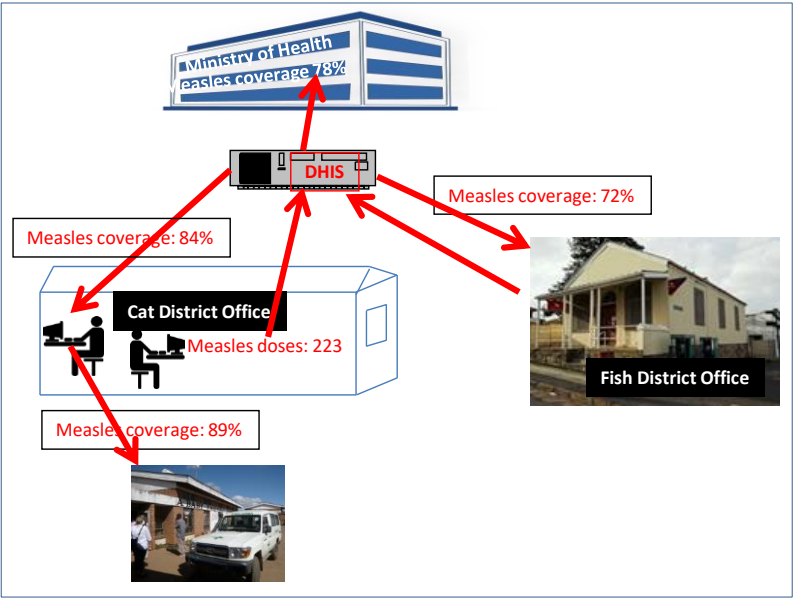


Figure 16. A business oriented model of data flow between organizational levels and units

Drawing maps of information flow on wall charts and gluing paper prints of screen windows which are accessed help providing an overview for the involved participants. The final chart may also help others understand the connections within a large organisation.

Data flow diagrams (de Marco, 1979) like the example in Figure 17 have been used for functional specification of information systems. The rectangle denotes a person, the circle a process, the parallel lines data storage and the arrows data flows. These are drawn in an abstract notation, but they are in general easier to understand than data or object models (Vessey and Conger, 1994). Data flow diagrams provide an overview of which data is going from where to where. In larger cases, processes are broken down in another data flow diagram to show more details. Thus, each diagram is kept reasonably simple.

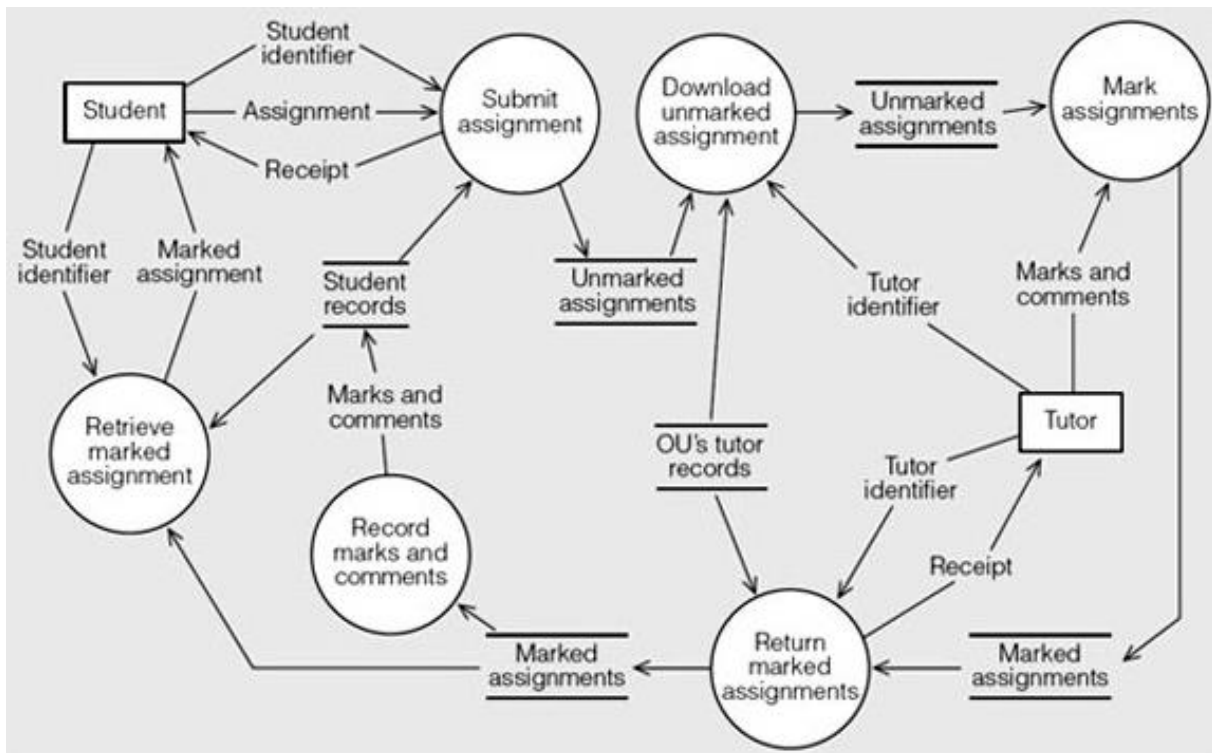


Figure 17. Business oriented documentation: Data flow diagram. (vcilt.uom.ac.mu)

3.6. Confronting misconceptions

Some users develop misconceptions on how the IT is embedded in the organisation. Confronting misconceptions has been found necessary for developing a more adequate understanding of any subject matter (Ramsden, 2003, p. 87).

The simplest way of *confronting business misconceptions* is for a trainer or another person to engage in a conversation with learners, detect when they don't have adequate understanding of the system, explain why, and provide a better conception.

In an organisation, the issue may be to make people aware of the consequences of their work for others. Users may not know how others rely on the information they provide in the systems

Yma and Jussi are colleagues in the hotel. Yma was unaware that her change in the booking system had implications on Jussi's way of organising work. Jussi explained this way to Yma, and she seemed to understand and might not do a late change of booking hereafter.

Confronting misconception—Jussi and Yma:

Receptionist Yma: *The guest complained that the room had not been properly cleaned*

Cleaner Jussi: *You changed the booking just when the guest arrived. That made it impossible to clean the room in time. We prioritize the rooms with bookings.*

Yma: *Oh, I'm sorry, I thought you took them floor by floor.*

While trainers and colleagues may be better suited for confronting misconceptions, Figure 18 illustrates how a multiple choice question with explanations can possibly clarify

misconceptions. After an erroneous answer, the user receives an explanation as to why this response is wrong. While personal explanations might be more effective, automatic response to multiple choice questions does not require another person's presence and may be available anytime and anywhere.

<p>Why is data validation important?</p> <p>a) This avoids recounting the registers. Wrong answer. Data validation might trigger a recount if the number seems erroneous.</p> <p>b) To complete the data entry form, such that superiors can get reports. Wrong answer. Data validation helps improving data quality.</p> <p>c) To get accurate data, so well informed decisions can be made. Correct answer.</p>
--

Figure 18. A multiple choice question which can target possible misconceptions. The explanations in Sans Serif fonts are to be displayed after the learner has responded.

3.7. Summary

Understanding the usefulness of the technology for one's own work is a driver for learning it and is therefore considered the most important topic to learn for IT users. Scaffolds should therefore motivate for usefulness. *Minimal manuals* consisting of *tasks* and *instructions* are the smallest sufficient scaffold for learning IT use.

Understanding the purpose of the IT for the organisation is important for cooperation. *Business oriented models* is a type of scaffold that can help users acquire an adequate understanding of the fit of IT in a business. Users without a proper conception need to be informed about their misunderstandings and be guided to a more appropriate one; we call this type of scaffold *confronting business misconceptions*.

2. Make sure users understand the usefulness of the IT.

Chapter 4. Understanding IT

The learning objective of this chapter is to be able to develop scaffolds that will help users understand IT.

Information technology is characterised by a quick turnover of new software versions, information systems and hardware gadgets. Users therefore constantly need to learn about new technology.

From the educational sciences, we know that understanding ease applying skills to new situations (Bransford, 2000). For example, a computer user who has understood the concept of text flow, and that text flows from one column to another, but not between cells in a table, would be more likely to choose the right kind of text structuring tool in a new word processor.

For the novice user, IT may look confusing, and different devices and software packages may present general principles and concepts in idiosyncratic ways. This chapter will identify some IT concepts and principles which should be recognisable for users and which appear in all software.

Computers and other IT technologies are constructed on the basis of a few principles. Throughout half a century, more principles have been introduced for easing the design of software, and these principles have also appeared in user applications. Some basic concepts for user programs will be introduced.

When using IT, people learn skills, but practice does not necessarily promote understanding.

Pedagogical theory – Constructivism

The constructivist perspective on learning is based on the assumption that new skills and understanding is based on what we already know. For example, if we see a button in a new program that has the same name as in the previous program, we assume that it does the same operation. In general, when we are presented with something new, we always try to associate it with something familiar. Since new understanding is based on the already existing one, we construct our own knowledge; we do not copy the teacher's understanding.

People are active and communicating learners, and learning takes place in interactions with the environment, including fellow learners, teachers, computers and books. A culture which encourages questioning therefore promotes learning.

In order to accommodate for the learners' construction, the teacher needs to know the learners' starting level, such that teaching can start at that level. Learning can go wrong when teachers start from a basis that is not aligned with the learner's background.

While skills are associated with doing, understanding requires the ability to express and communicate in talking or writing about the subject matter areas.

This chapter will first describe the learning trajectory when reaching two levels of IT understanding. Scaffolds for supporting understanding will be presented for each of these levels, and issues of misconceptions will be discussed.

4.1. From skills to understanding

People learn doing first, and second, they may reflect on what they did. The second step triggers understanding of ideas, concepts, principles, relationships, etc. Further, people learn through abstract conceptualisation, which includes relating concepts to each other. Studies of learning of abstract concepts in maths support the direction of learning from concrete experience to abstract concepts (Sfard, 1991) in line with the constructivist learning process of reflecting on experience. Since IT concepts also are formal and abstract, like maths, it is reasonable to believe that IT is learnt in a similar way.

In a study of novice users, who learn searching in a database, the learners became able to describe their searching in terms of the computer's operation, but they were unable to state how the system worked (Borgman, 1986). This means that they could do it, but had not reached an understanding of searching.

Programmers have to learn more IT concepts and principles than users. In a study on the learning of the array concept, it was found that the learners start by coding an array (Aharoni 2000). This means that they carried out an action and gained concrete experience. After having compared the starting point and the effects of their actions, the learners were able to refer to the basis and the outcome of the action without mentioning the steps taken during the action. Their learning process consisted of *comparing input and output*, and its result is called *functional understanding*, see Figure 19. Thereafter, the learners reified the action into a concept and related it to other concepts, e.g., being able to say that that an array consists of indexed variables. Since the learner can express the structure of the technology, she has reached a *structural understanding*, and the learning process of reification and relating is called *conceptualisation*.

Three other studies support the theorized sequence of functional understanding appearing prior to structural understanding. First, novices learnt process modelling with input-process-output units, often combined into sequences and networks, more easily than data and object models capturing data structures (Vessey and Conger, 1994). Second, some year 10 pupils reached a functional understanding of the concept of master slides, but none reached a structural understanding (Stamatova and Kaasbøll, 2007). Third, college students being tested in text processing scored lowest on questions which required an understanding of the document as consisting of more than one text flow, i.e., a more elaborate structural understanding (Grant et al., 2009).

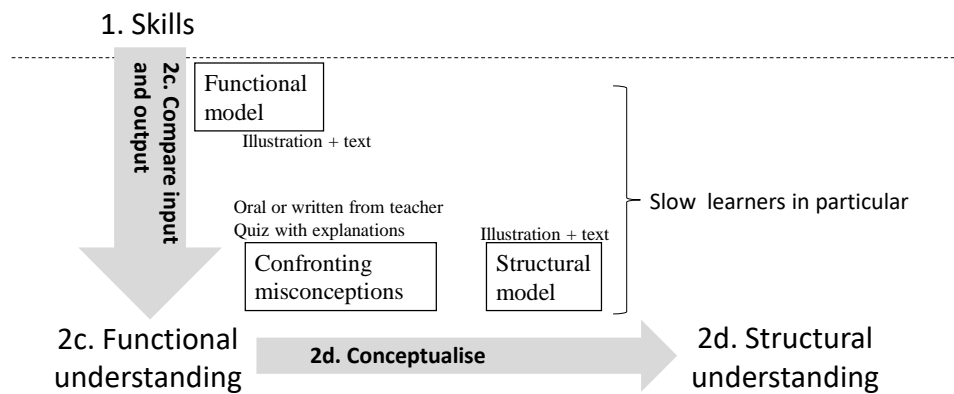


Figure 19. Learning processes and learning outcomes concerning understanding of IT.

Learners' level of mastery can be found by observing what they do and say. Assume that they are supposed to learn the concept of a master slide, and how changes in the master slide can affect all the slides in a presentation file. Learners who have completed the sequence of computer operations that demonstrates the skill can be asked to express the concept.

George is repeating the steps that he carried out without expressing the initial state and the outcome of the operation, hence he has not reached the level of functional understanding yet.

Master slide—George:

I went to view and slide master and then changed the font size, and went back to the normal view.

Mireille has expressed the starting state of a font size, which is different from what she wants. Further, she mentions that the master slide can do all the desirable changes, which is expressing the function of the operation. Mireille is therefore at the functional understanding level. By mentioning font size and slides in her explanation, she demonstrates that for these two concepts, she is at the structural level of understanding.

Master slide—Mireille:

I wanted to change the font size of all the slides, so I changed it at the master slide. Then it will change all the slides.

The learner Domenico demonstrates that he is at the structural level of master slide competence. By comparing master slide with page layout, Domenico refers to master slides as an entity of its own. Since Domenico has grasped the master slide idea in addition to seeing its resemblance with page layout in a text processor, he is ready for learning a more general concept. The ability to compare master slide with other concepts means that Domenico can build a structure of IT concepts; hence, he has reached a structural understanding.

Master slide—Domenico:

Master slides control the appearance of the normal slides. When you change the master, all the others will change too. It's like the page layout in the text processor, which also changes every page.

While some users generate adequate understanding on their own, slow learners are particularly bad at doing this (Furuta, 2000). When trying to manipulate a system with unknown behaviour patterns and no scaffolds, the good learners came close to a functional understanding of its operation (Furuta, 2000). The few accounts that the poor learners came up with indicated skill level rather than understanding, e.g. saying

I am turning the left knob to the right (clockwise).

Learning a new concept builds on those already known. For instance, understanding addition is a prerequisite for learning multiplication, and knowing the difference between document and picture formats is necessary when trying to edit a picture of a text. If you haven't understood the prerequisite, you will have serious trouble understanding the next concept, thus making a poor learner into an even poorer one.

'Mental models' is often used as a common term for functional and structural understanding. In addition, mental models are normally also considered to include the way the interface is operated (Westbrook, 2006).

Since reaching structural understanding of some software may require effort additional effort, many users who are more interested in using the computer for their purpose than for understanding how it is structured may not reach the competence level of structural understanding.

4.2. Functional models

For planning how to train or help others obtain a functional understanding, we need to know what the learning objective should be. An analysis of the function is therefore appropriate.

Any function has at least one purpose, denoting its usefulness for some activity outside the function itself. This aspect was explained in more detail in Chapter 3.

A function has an input state, one or more operations bringing about a change, and an output state which is the result of the operations. For instance, for the function Converting to pdf, the input state is a file of another type, the operation is the conversion process, and the output state consists of the original file plus a pdf-file which looks similar to the original when printed.

A function may have several steps. For instance, printing a report from a database involves selection of data, page set-up, previews and repetition of set-up, and selection of printer. We can imagine that complicated functions have any number of branching and repetition included. While instructions were presenting buttons and menu choices, *functional models* are scaffolds which intend to help building understanding, thus interaction details are omitted. Functional models include the input state, the operations and the output state. The input and output states may be omitted if obvious. A graphical model like Figure 20 can be used to show the series of steps for printing.

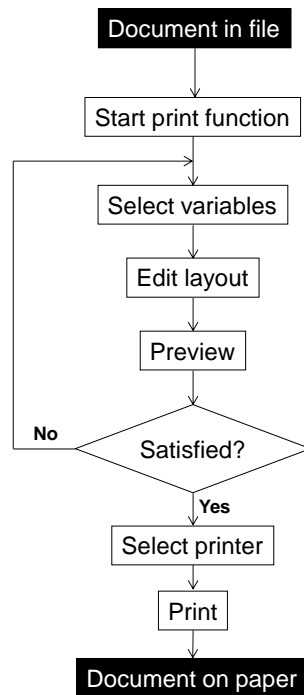


Figure 20. Functional model of a printing procedure.

For the explanation of input and output, a textual model may do, for instance as in Figure 21.

After conversion to pdf, a new, converted file with extension pdf will exist. The original file will be kept.

For example, after converting the file text.doc, you will have the files text.doc and text.pdf.

Figure 21. A functional model in pure text.

In a study of instructions versus functional models, ten novice users of a computer program were given instructions as scaffolds while ten others were provided with functional models (Dutke and Reimer, 2000). They were first given five tasks which corresponded closely to the instructions and functional models. Thereafter, they completed two more tasks which differed somewhat from the scaffolds provided. While there was no significant difference in the first five tasks, those who had received functional models performed better in the modified tasks (Dutke and Reimer, 2000). Despite the small number of learners, the study indicates that the competence acquired through functional models is more robust when transferred to new settings.

Since slow learners are particularly bad at generating functional and structural understanding (Furuta, 2000), they will benefit more than fast learners from being explicitly taught by means of functional or structural models. In a test comparing explanations with and without diagrams, including diagrams improved learning, in particular for learners with low verbal abilities (Cuevas et al., 2002).

Some computer operations have side-effects. Data entered in social media and search engines are routinely used for sending ads which target your area of interest to you. A functional model which captures this effect is provided in Figure 22.

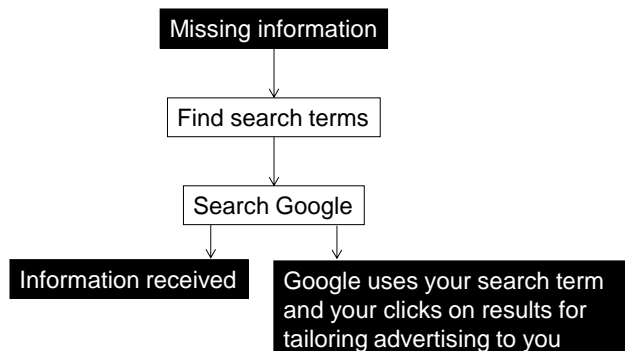


Figure 22. A functional model showing two outcomes of a computer operation.

Security hazards is another frequent type of side-effects of software functionality. These should therefore also be included in the functional models.

Those aspects that appear difficult to understand should be supplied with a functional model. However, no encyclopaedia exists on which aspects people may struggle with when learning IT. We can base our judgements on three factors:

1. Distance between what the user already knows and the new function to be learnt. The more novelty, the more need for scaffolds. If done in a course, we should know what the learner is already familiar with. Otherwise, we have to make assumptions.
2. Interference with similar functions. People may misunderstand a function because they believe that a new function to be learnt has similar effects to a function that they already know, while this belief is wrong.
3. Experience with common learning challenges.

Functions across software, like import or linking, often introduce novel processes. For example, importing records to a database from a spreadsheet may trigger a merging of records that are duplicates; see Figure 23 for an illustration of the learning process and the functional model.

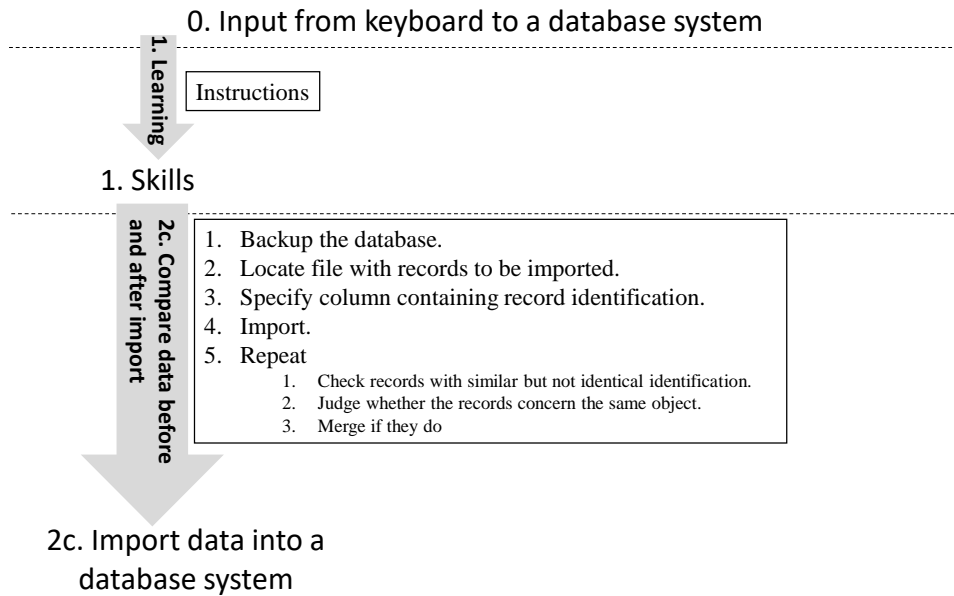


Figure 23. Reaching a functional understanding of data import which may trigger merging of records.

The following is an example of interference. A user is familiar with the functionality of editors, knowing that they can undo if they regret the last edits done. Then they start working with a database system keeping the functional understanding that they can undo their transactions. However, the database system does not allow this. Figure 24 illustrates the learning process and a functional model, however, learning steps 1-2b are omitted. The instructions would in this case have the same sequence as in the functional model and include interface details, while the functional model provides an overview and the motivation for each step. (Heeks, 1998)

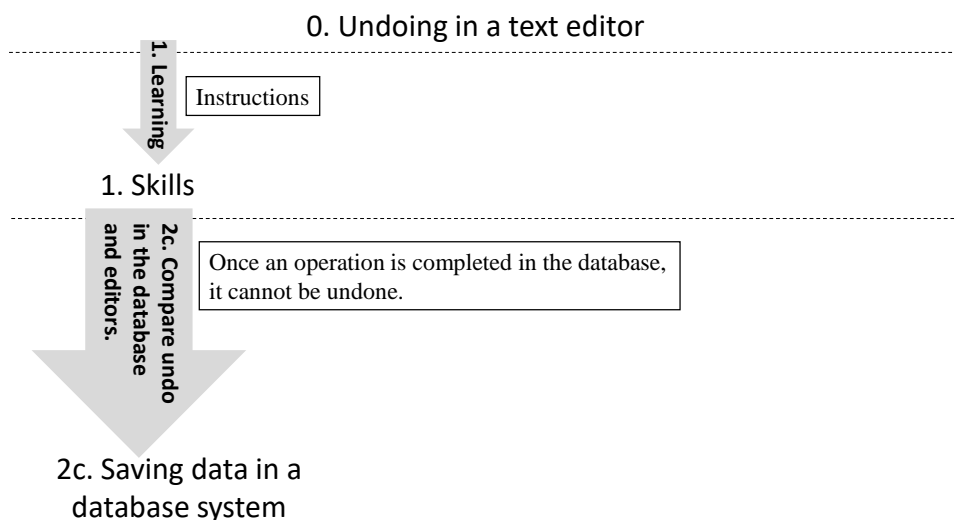


Figure 24. Learning the difference between undo in a text editor and a database.

Error messages from the computer often have the content of a functional model. For example, the message which appears when a web page is not found may tell exactly this output of the

operation (Page not found) and may also suggest reasons like mistakes in input, see the example in Figure 25.

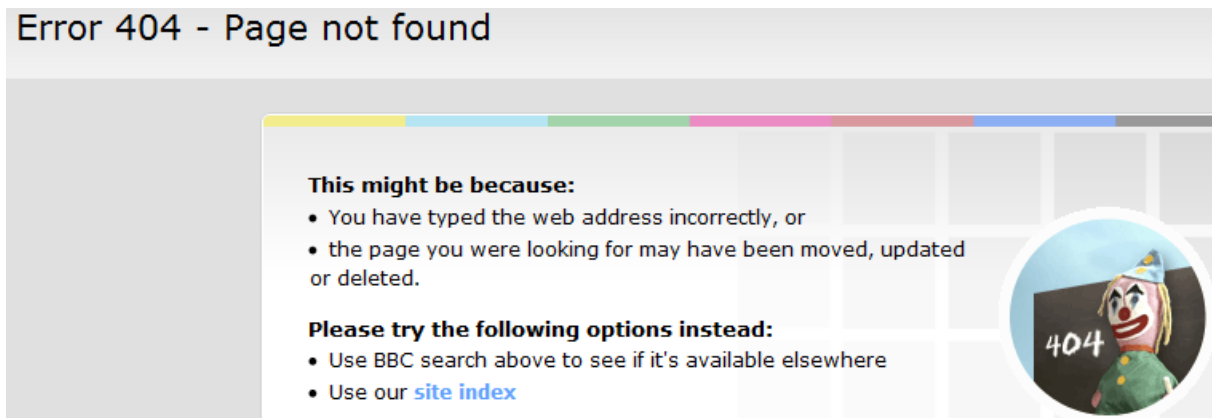


Figure 25. Error message being a functional model.

4.3. Confronting functional misconceptions

While being exposed to a functional model will constitute an effective scaffold for some users, others will nevertheless develop misconceptions. *Confronting functional misconceptions* is important for learning in the same way as confronting business misconceptions, see Section 3.6.

Bobby seems to be of the opinion that when an indicator value is available in

the management information systems, there is no point in him entering his data for the very same indicator. Aziza tries to explain that the indicator value is not fixed, and that Bobby's data entry will cause it to change, thus hopefully correcting Bobby's misunderstanding. Like Aziza's response, the explanation to correct a functional misconception will be a functional model.

Confronting functional misunderstanding—Aziza and Bobby

Bobby: *The measles indicator for the country is already calculated to 78%, so my numbers don't need to be entered.*

Aziza: *No, we need your numbers too. When you enter your immunization data, the country indicator will be updated accordingly.*

4.4. Qualities of functional models

Since instruction sheets also capture processes, we will compare the quality of functional models with those of instructions.

Recognisable. Functional models are supposed to be independent of the user interface, hence they should not include screenshots. The input and output are the data, and learners who recognise the data would more easily understand the input and output. Thus, an example of data as input and output could be advantageous. Examples of data were not provided in Figure 20 - Figure 22, since it was assumed that users who carried out these operations would understand the descriptions of input and output.

Proximity is a quality of any phenomenon, including text, graphics and videos. Thus the principle of what belongs together should be visible close to each other also concerns functional models.

Sequential. Input-process-output is a sequence in time, thus functional modes also need to be sequential. The example in Figure 20 illustrates that selections and iterations may be included.

Direction is denoting where to find the starting operation in the user interface. Again, functional models do not concern user interface, hence direction is irrelevant for functional models.

Completeness and feedback are as relevant for functional models as for instructions. A missing step would be quite confusing. Also, the output should be described such that the user is provided feedback that the function is satisfactorily performed, or that something went wrong.

Short. As with all documentation, users prefer doing to reading. Thus also functional models should not include unnecessary steps.

Level of detail. As for instructions, the functional model has to assume a level of user competence according to the functionality being described by the model.

The example in Figure 22 introduced an element which did not come into play with instructions; **side-effects**. Functionality with side effects, which are not included in the functional model would be of poorer quality than models which included the side effects. Security hazards could be

4.5. Structural models

When requested to explain a word processor, Astor is talking about what he is doing with the program in the sense of its functions. His comprehension of a document may be that it consists of pages filled

Functional understanding—Astor:

You can write and print in a text processor, and change and print again. And then you can change the layout of pages and the fonts.

with letters in a specific font. Astor talks about text processors in the same way as most people, and when asked about the data in the file they have produced, few can tell anything about how a text document is structured. While learning to use a text processor seems simple for a start, documents have complicated structures that many users may not be aware of, something which may hamper what they can achieve. Astor has not conceptualised the document yet which is necessary for a structural understanding, see Figure 19.

This section addresses data structures, and how *structural models* can help achieving a structural understanding. A structural model depicts data structures or structures of IT concepts. Structural models normally consist of graphics and text, and they are not sequential, see examples in Figure 26.

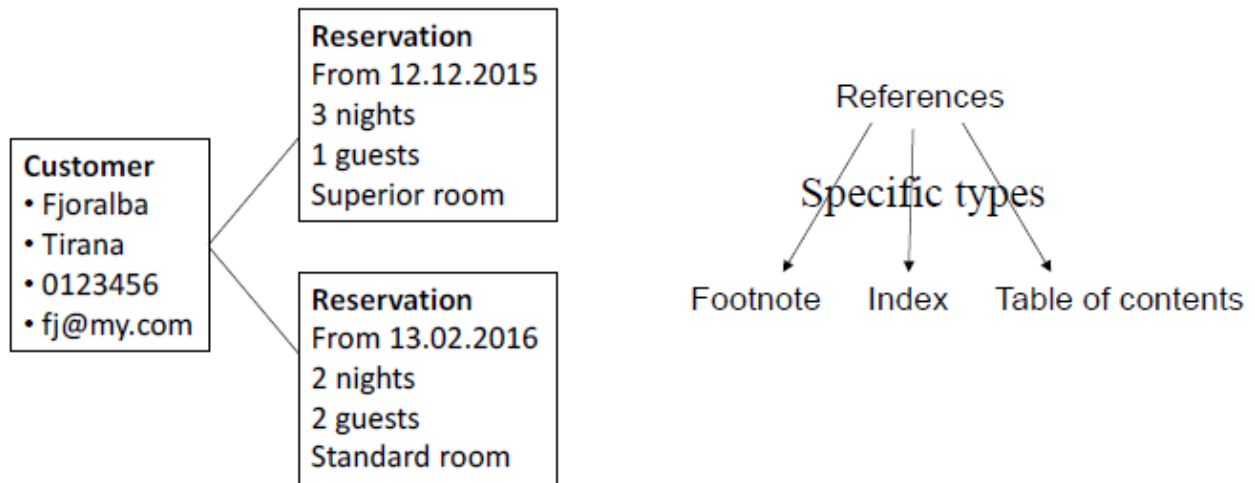


Figure 26. Structural models. Left: model of data. Right: model of relationships between IT concepts.

The file system is an example of a structure, which most users need to understand. Its purpose is to provide access to the files and folders in the computer. It enables functions like creation, deletion, copy and paste of these files and folders, and it is organised in a hierarchical structure. Through a network, it may also provide a relation to other file systems, implying an external structure as well. The file system differs from application software in that it does not consider the interior of files.

While the hierarchical structure may be obvious due to the way it is presented in the user interface, there are two features of the structure, which constitute learning challenges for many users. First, shortcuts or aliases are files, which are links to other files, and such links break the hierarchical structure. Some users get confused when they encounter one, wondering what this is. Second, folders giving access to remote computers appear as a folder on your own computer. People are used to conceiving the files that they see as being stored in their computer. Thus seeing a folder with files being stored in a server can be a novelty.

Figure 27 is NOT a structural model, since it neither portrays data nor relationships between IT concepts. The kind of illustration in Figure 27 could be a part of instructions.

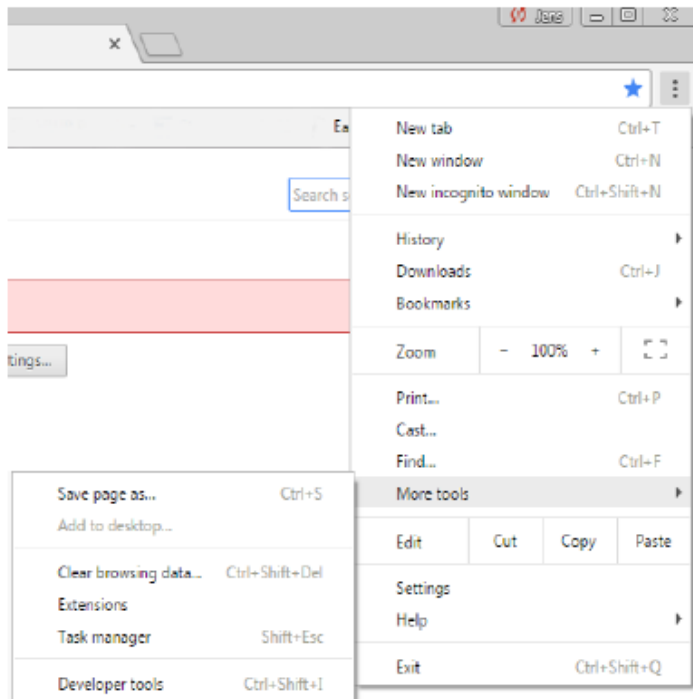


Figure 27. A screenshot of menu choices.

IT structures — Recognisable elements

When using an application, some features and principles are easily recognisable at the interface, for example that the cells in a spreadsheet are organised in a grid, and that the text in a document has a specific layout. The sequence of operations, typically whether to choose data before operation or vice versa, may not be displayed, but the sequence is experienced through actions, so the user obtains an immediate impression.

Other features are less prominent. Examples of hidden features are that the text in table cells in a text document does not belong to the main text flow, and that behind a number in a spreadsheet cell could be a formula, which refers to many other cells. In the word processor, there is no intuitive way to see where one text flow starts and another one ends. It might be possible to view the non-printing characters, but these do not necessarily tell us about the text flows or many other properties of the document, like paragraph and character styles.

When the user interface does not show the hidden features, these should be made explicit through a structural model. The written text is a one-dimensional sequence, while structures in the computer often are of other kinds. Since many hidden aspects are structural, a combination of language and graphics would normally be a better option than just one of them.

Creating useful graphical models is partly arts & crafts, but there are also principles to consider. In the following, specific considerations for visualising the interior functioning and structure of software are presented.

Any presentation of what goes on in the interior of the computer should be based on the current competence of the users, including the users' understanding of concepts, experience with operating the software and their background for understanding the notation used.

In order to aid understanding, and not make learning more difficult, graphical representations need to be

- simple, in the sense that they contain few (7 ± 2) elements
- recognisable, such that each element provides immediate meaning. Including a known example in the model improves recognisability.

A structural model of the internal structure of the file system could be written:

```
Folders can contain files, links and other folders.
```

Figure 28 is an illustration of the same. It is simple, but is made with a notation which is not recognisable by most users. Also, it includes no example. On the other hand, maximum recognisability is sought in Figure 29, and this visualisation of the same data structure also aims at providing a general model of the structure of the file system. Figure 29 also uses examples instead of the general categories in Figure 28, bringing it closer to user experience but making the illustration larger and less simple. There is often a trade-off between simplicity and recognisability.

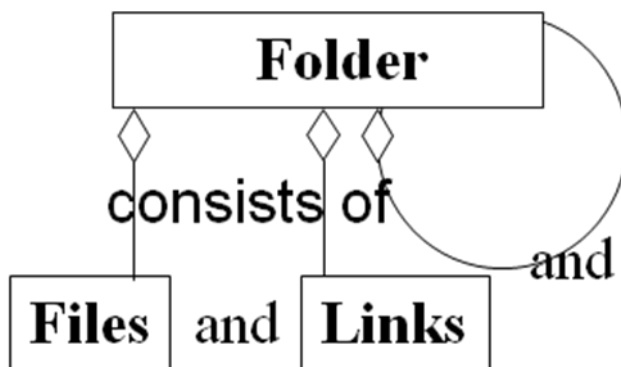


Figure 28. Abstract model of the file system

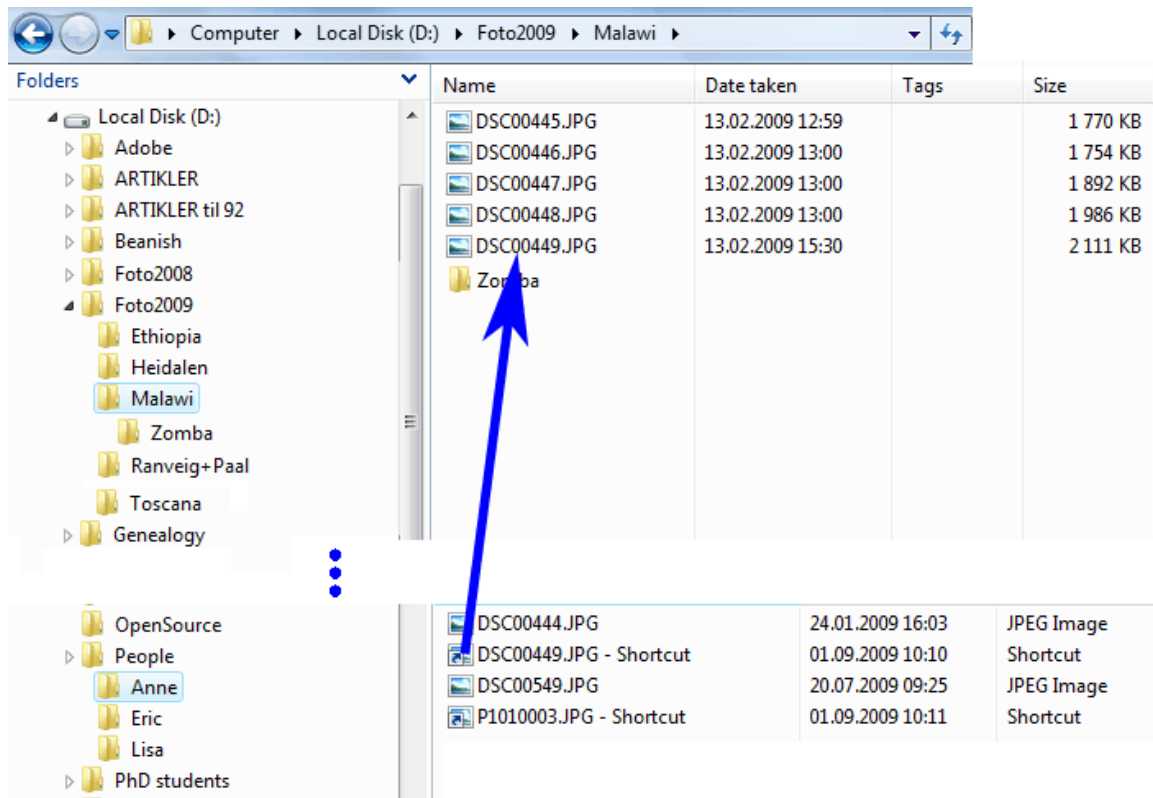


Figure 29. Recognisable model of the file system.

Figure 28 is a model of how the file system can be conceived under the surface, while Figure 29 is mainly a surface model with an additional graphical element for showing the under-the-surface connection. The user interface of the Windows file system provides a reasonably comprehensive view of the data structure when viewed in the Explore mode shown here.

Beware that there are several other aspects of the file system, and these are not included in the structural models presented. Including several aspects would often clutter up a graphical presentation, such that other aspects should be presented separately.

A summary of research on teaching in formal domains shows that textual explanations in combination with diagrams is the best combination for developing structural understanding amongst the learners (Wittwer and Renkl, 2010). Our brains have a very limited short-term memory, making it impossible to make sense of many stimuli occurring concurrently, see Cognitive load in Section 2.1. However, sight and hearing operate in parallel, so we can combine visual and audio impressions and make sense of the combination. Textual explanations accompanying a diagram should therefore preferably be oral.

An experiment demonstrated that learning can be greatly improved by aligning structural models with the learners' visual ability and preferred style of learning. One hundred undergraduate students were divided into high and low visual ability through a paper-folding test. Half were given an abstract structural hierarchical model of folders and messages of an e-mail system, while the other half received a model that demonstrated the analogy between the computer structure and letters in paper-folders. Those of high visual ability learnt better from the abstract model, while the low visual ability students learnt more from the analogy model

(Sein and Bostrom, 1989). The students were also grouped into their preferred learning styles, being abstract reflection versus concrete repetition and imitation. The abstract learners with abstract structural model outperformed other combinations, while the concrete learners with the analogy models were second best. Abstract learners with analogy models and concrete learners with abstract models were the worst combinations (Sein and Bostrom, 1989). This shows that one type of learning material does not fit all.

Since we normally cannot know in advance what learners prefer, the solution is creating a variety of models, some abstract like Figure 28 and other more concrete like Figure 29. While abstract models would be constructed from boxes and arrows, the concrete model should depict a phenomenon which the learner is familiar with. The concrete model in the experiment reflected a domain outside of computers. If the learners have concrete experience with the computer interface, this could also be used in structural and functional models, like Figure 29.

Videos

The contents and illustrations for structural models could be the same in documents and videos. There are a few guidelines particularly for videos, however.

People's ability to receive and combine visual and oral stimuli indicates that verbal explanations should be provided orally. For instance, when presenting a structural model like Figure 29 in a video, the graphics should constitute the visuals, and a voice should explain it (Clark, 2007). There should also be a finger or a colour blob marking the area that is talked about at the moment.

Adding explanatory text in the picture for presenting the graphics would be a mistake. That would overload visual capacity while not utilising our hearing.

It has also been found that video presentations are more effective when the viewer feels like there is a conversation going on. In order to strengthen this impression, there should be a real voice, and not a computer generated one (Clark, 2007). The feeling of being in a conversation is also strengthened if a person is visible on the screen for periods, although this could be a simple, animated figure (Clark, 2007).

Like any other scaffold, videos need to be as short as possible, meaning that also functional and structural models need to be presented without additional disturbance. An example is normally needed, and this should therefore be to the point without additional features.

A simple sequence for a video providing a functional or structural model could be:

1. Picture of a person presenting the concept briefly.
2. Picture of graphics with a finger or colour spot. The voice of the person.
3. Picture of a person repeating the concept.

The remaining sections of this chapter will address specific structures, which users may struggle with learning and thus need a structural model or having their misconceptions

confronted to get it right. When reviewing training material, the lack of structural models seems to indicate that also trainers and documenters find it troublesome to make structural models.

4.6. Data structures

Data units of the same or of different types are organised into larger structures, enabling composite types. There are four basic kinds of structuring:

Sequence. The data units are following each other, and they may be numbered. The letters in a text document are sequentially ordered. A table in a database is also a sequence, although its ordering is irrelevant to the user.

Grid. The units are organised in a grid, which can be accessed through coordinates. A raster graphic image has pixels, which can be numbered horizontally and vertically. There can be more than two dimensions. Grids can be conceived as multi-dimensional sequences.

Hierarchy. The units are organised like containers within larger containers. The files in the operating system are organised in a hierarchy.

Network. The data is linked in a network without any strict topology like the hierarchy. The web is a network of pages.

A web page is a sequentially structured data type, which is then a part of a larger network structure.

Again, there are operations that can manipulate the structures, like inserting a new column in the spreadsheet or moving a folder from one place in the hierarchy of the operating system to another.

Computers allow for creating links, references, shortcuts or whatever they are called in order to achieve two effects:

Functional dependency, meaning that when data is changed where stored, the changes are also accessible from where the link to the data goes. This principle ensures that data is stored and hence updated one place, so that inconsistencies are prevented.

Breaking the structure, in the sense that from one place in a file, there is a reference, which brings the user to anywhere else in the file or to another file, regardless of hierarchical, grid, sequential or other orders.

User training or documentation would normally not present these types of data structures, but rather make structural models that are recognisable for the users.

A spreadsheet consists of a grid of cells with row and column coordinates. When creating formulas referencing other cells, users create an additional network structure on their data. The spreadsheet programs Calc and Excel can display descendants and precedents of formulas, thereby showing the user generated data structures.

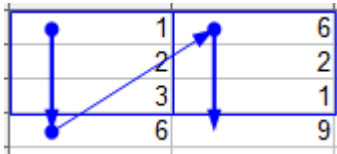


Figure 30. Visualising data structure by tracing descendants in a spreadsheet.

Documents generated with text processors or web page editors have hierarchical structures. Entities like paragraphs are composed from smaller entities like characters in a sequence. Files generated by presentation programs also have a similar structure, as shown in the model in Figure 87 and Figure 88.

In software for editing graphics, it is often possible for users to structure the data in groups, which can be manipulated as whole entities and in a sequence of layers, each of which can be changed separately from the other. While the software may have ways of showing the layer structure, this is often not the case for user-defined groups of graphical elements.

Networks

Seen from a user, relational databases have connected records and a multitude of functions for accessing them. For example, an information system for managing rooms and guests in a hotel may have functions for booking, payments and management. It may contain records for guests, rooms and reservations, and these records may be connected.

Asking a receptionist about the system, he might reply like Theo does. Theo seems to have a sound functional understanding of the system. He might also have understood that “a Reservation requires one main Customer, and a Customer can have several Reservations,” which is a structural

Data structure understanding—Theo:

Theo: I work in the reception. When a person calls and want to make a reception, I have to check whether that person is already in our system, and if not, I have to register her personal details first. Thereafter, I open the New reservation window and enter the dates and room type. We have five room types which I can select.

understanding of the data. Theo does not express the structure in this way, but he may know this connection between customers and reservations. Getting to know users’ understanding would often require a series of questions and answers.

A data model for the hotel reservation system could be used as a structural model for scaffolding user learning. Figure 31 presents the one-to-many relationship between Customer and Reservation in two ways. To the left we see a model used by programmers, including notation that is from the computer realm. The model to the right illustrates that a customer can have more than one reservation by means of an example and by duplicating the Reservations. The latter is thus more recognisable for the user. It does not tell that a customer can have more than two reservations, such that some users may ask about this.

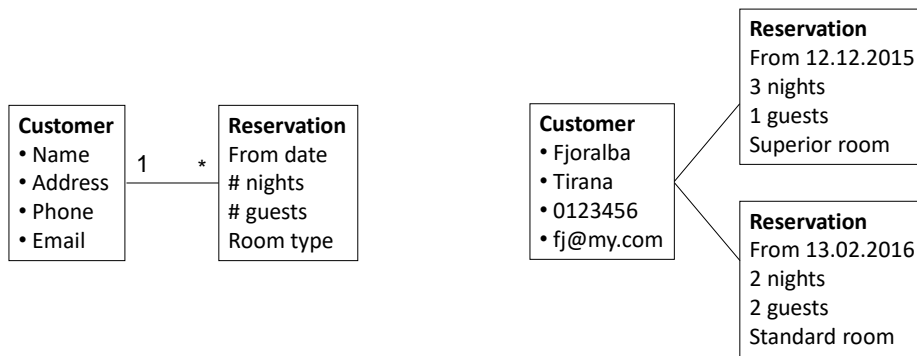


Figure 31. Left: data model for computer scientists. Right: A data model for presentation to users.

Many users might not need diagrams like in Figure 31 for understanding such a simple structure, and a simple sentence like

A Customer can have 0 or more Reservations.

could do. Since text has a sequential structure, a data model can be explained with text as long as there is a sequential path through the structure. If not, graphical models are superior. Figure 32 shows a larger part of the data model, and there is no obvious sequence in this structure.

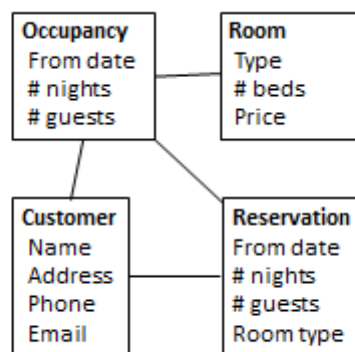


Figure 32. Data model for a larger part of the hotel information system.

In this system, the two entities Customer and Room correspond to physical objects in the domain, Occupancy is a relation between a customer and a room during a period of time, while Reservation is information about a possible, future occupancy. Although hotel staff would know what a reservation means, information about tangible objects like customer and room is more concrete and therefore more intuitive. For information concerning abstract phenomena like reservation, it is, for instance, not obvious whether the reservation should include specific rooms or only room types. Therefore, structural models are more needed for the information entities that have no tangible counterpart in the domain of the information system. Figure 33 shows the starting level 0 being competence about the hotel services and the learning outcome 2d being understanding of the data structure. This illustration does not present the specifics of steps 1-2c.

0. Tangible objects like Guests, Rooms. Abstract concepts like Reservation

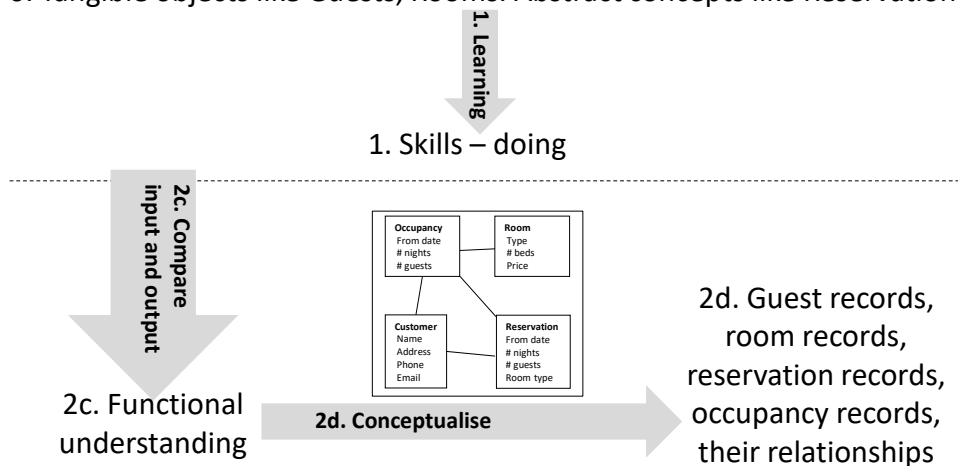


Figure 33. The learning process for understanding the data structure of the hotel information system.

A data model of all entities in a corporate information system may include hundreds of entities spanning many pages, and the structures are often complicated. Hence, users might need simplified models corresponding to their needs for understanding and navigation.

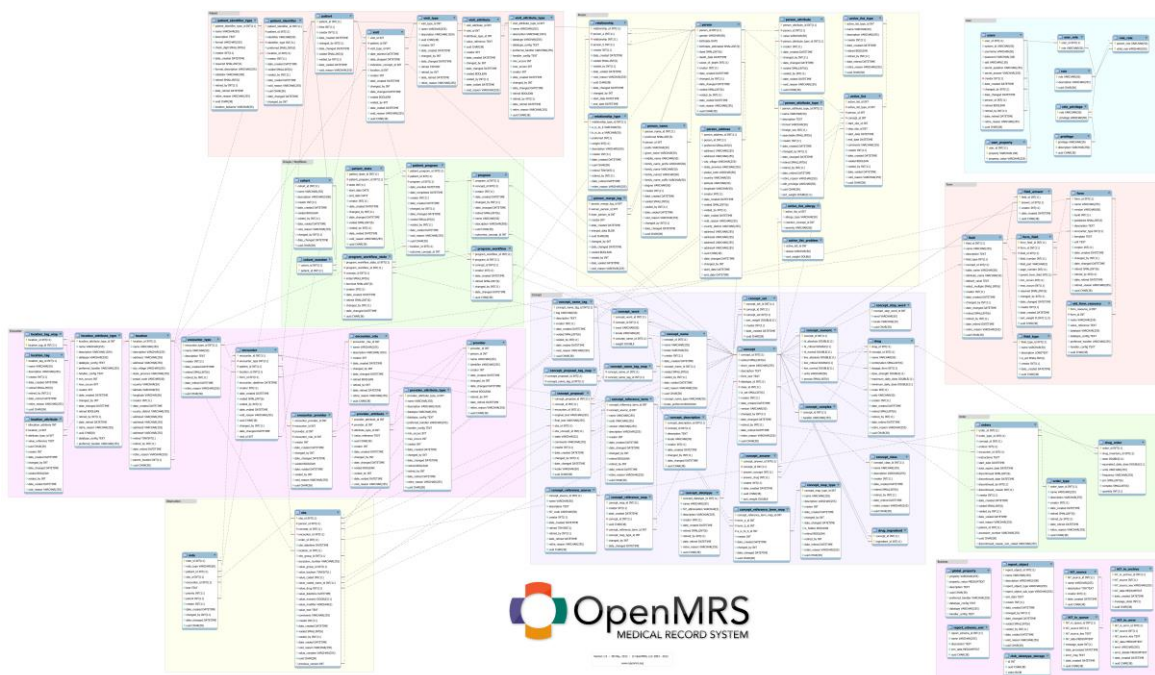


Figure 34. The Open MRS data model for patients in hospitals.

Figure 34 shows the developer version of a data model for a patient information system. A visit is split into four entities in this model, see Figure 35.

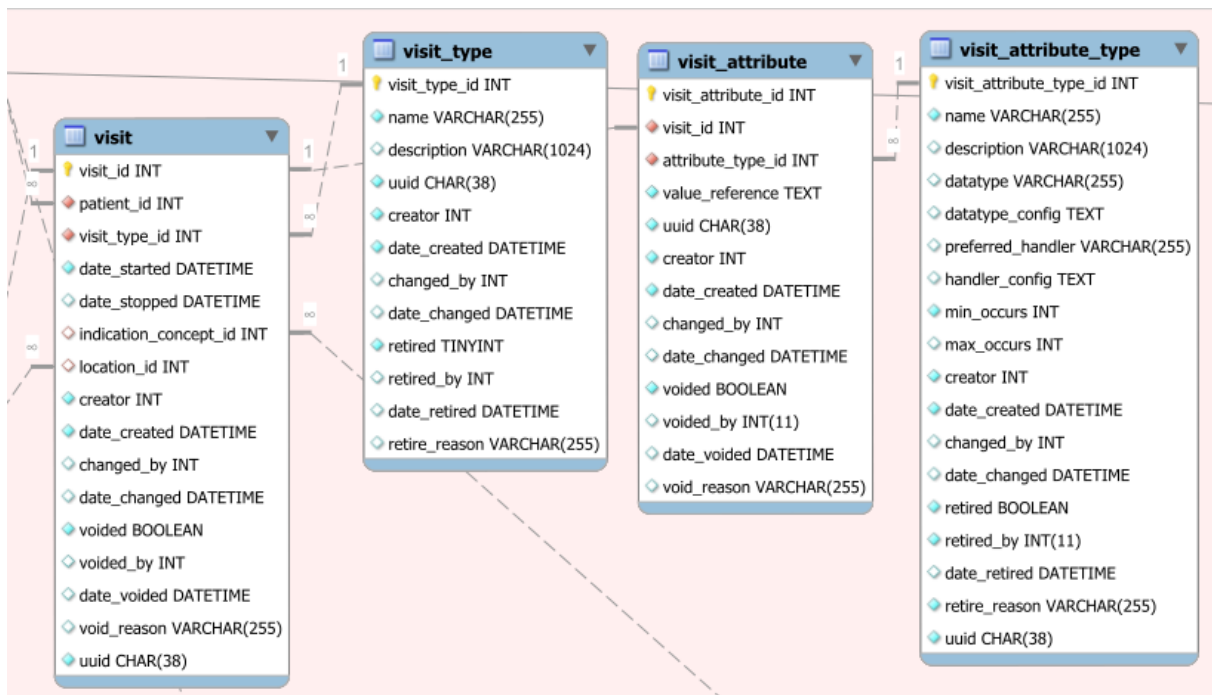


Figure 35. Visit entities in the OpenMRS data model, extracted from Figure 34.

The model includes Visit_types, Visit_attributes and Visit_attribute_types. When registering a new visit, the attendant adds a visit to a patient, and the types and attributes would likely appear as drop-down lists of options and fields to fill. Hence, only the visit and its relation to a patient are relevant for users who register visits. Normally, in corporate information systems, a few people have the authority to alter types and attributes, such that these parts of the model are not relevant for most users. While the model in Figure 35 is relevant for the few people who change the types, a structural model for the majority would only include patient and visit.

4.7. Qualities of structural models

In summary, when designing structural data models for user learning, the following considerations are useful:

1. User group. Normal entering and reporting or setting up data structures? The latter calls for including more entities.
2. Overarching structure. Network for relational databases, hierarchies of sequences for documents, grids for spreadsheets, etc.
3. Abstract entities. While information objects with a physical counterpart may be obvious to novice users, more abstract objects capturing events or relationships may be in more need of a structural model.
4. Examples. Users with poor understanding will learn more easily when the model contains an example, like the model to the right in Figure 31.

4.8. Data types and instances

The `Visit_types` and `Visit_attribute_types` in Figure 35 give rise to *type-instance* relations, such that the system can store many visits of the same type. While this is a case where some users can create types, software is filled with types made by computer scientists, for example the types `Number`, `Text` and `jpeg-format`. Data types may also offer operations on the data, like calculations for numbers and editing for jpeg-files, and types also restricts what can be stored in the data unit. When creating the `Visit_type`, the user cannot create any operation, only a set of data, possibly except automatically generated data entry and reporting fields.

While in the `Visit_type` example, only a few, selected users have the access right to changing the types, any user may have to deal with types in other settings. For instance, when scanning a document, users may choose whether to create a picture file or a format that allows also for storing characters that can be optically recognized by the computer application. The tasks to be carried out later might decide which option to choose; character recognition allows for searching through the text in the document, while a picture can be changed in contrast, colour, etc. Further, the picture format `tiff` is an uncompressed representation, leaving the picture exactly as scanned, while storing in the `jpeg` format compresses the file with some loss of detail, but at 5% of the storage space. This knowledge about two ways of representing may also be useful for working with scanning. Understanding the differences between file types is beneficial when having to select a type.

The relation between the `jpeg` file type and the files `edwin.jpg` and `emi.jpg` can be compared to the relation between human beings in general and the persons `Edwin` and `Emi`. File types correspond to species and files to living beings. Such relationships between categories and instantiations of the categories are well known for most people, hence users also easily understand file type – file relations.

However, most people do not modify genes or change other patterns for generating instances. People build new knowledge on what they already know. With little knowledge of changing types, users who learn to set up `Visit_types` may mix up `Visit_types` with `Visits`.

Recipes for cooking and patterns for knitting are examples of types from which users may have made food or sweaters. Even if they have not altered these types themselves, they may easily imagine doing so and be able to foresee the altered food or sweaters. A scaffold for conceptualising the type-instance relation could therefore compare `Visit_type` and `Visit` with recipe and food, as illustrated in Figure 36.

Recipe

Ingredients

- 4 dl wheat flour
- 8 dl milk
- 3 eggs
- 1 tsp salt

Directions

1. In a bowl, sift together the flour and salt. Pour in the milk and eggs and mix until smooth.
2. Heat a frying pan over medium heat. Scoop the batter into the pan, 1 dl per pancake. Brown on both sides and serve hot.

Food



Visit type

Follow up surgery

Inspect wound. Measure puls and blood pressure. Take standard tests for infection.

Changed 1 Jan 2013 by Emeline

Visits

Pablo
28 Feb 2013
Knee

Qing
1 Mar 2013
Open heart

Figure 36. A scaffold for conceptualising the Visit_type – Visit relation. The example on the left is intended to illustrate an already known relationship between types and instances.

In presentation programs, many users modify master slides, which constitute types for individual slides. Master slides allow for altering slide and text formats, which apply to all slides in a presentation. Master slides look like the slide instances, see Figure 37.

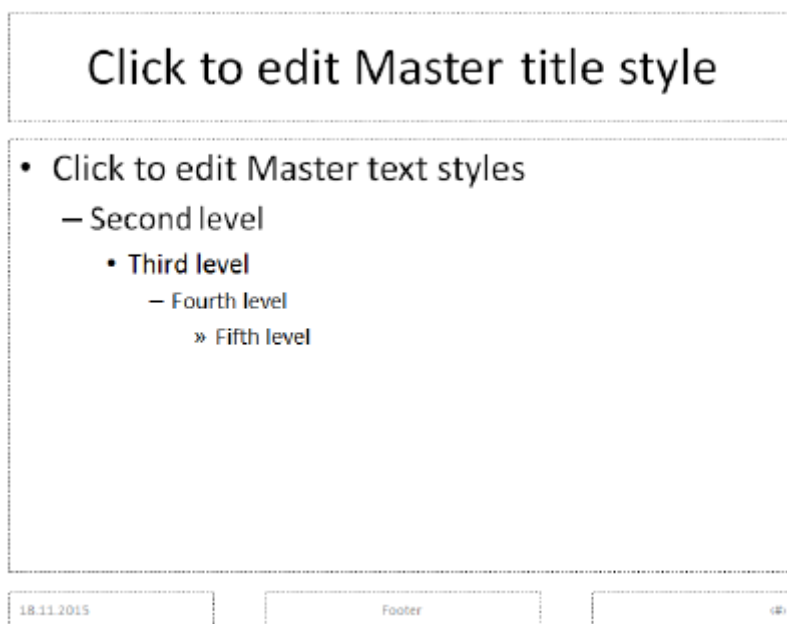


Figure 37. A master slide. From MS PowerPoint.

One issue arises when modifying a master slide; whether the modifications apply to the slides that have already been created with this master slide or only to new slides. In general, the learning challenge is:

1. A functional understanding of whether the modifications of a type apply to existing instances of the type or only to new instances,

The answer will depend of the software being used, which may create confusion amongst users.

Users of text processors may also create and modify types, often called Styles and Templates. While a master slide looks similar to the slides, a style in a text processor is normally presented as a window with many parameters, see illustration in Figure 38.

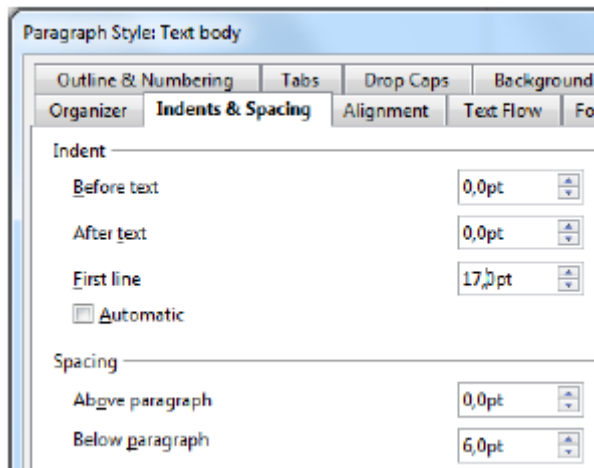


Figure 38. Window for modifying a style. From OpenOffice Writer.

This way of presenting styles triggers two more learning challenges:

2. The user needs a structural understanding of the units of data for which a style applies, which in the case of Figure 38 are paragraphs.
3. The user needs to have a functional understanding of the effect of changing a parameter on the appearance of the paragraph.

While the same master slide is often used for all slides in a presentation, a document would normally have different styles for body text, headings and other elements, bringing up a fourth learning challenge:

4. The user needs a structural understanding of the relation between a style and its associated paragraphs.

The relations between styles and paragraphs are not easily visible at the user interface. When managing paragraph styles, a data structure like shown in Figure 39 is created. The graphics have been extended with an introduction concerning the restrictions on the number of relationships.

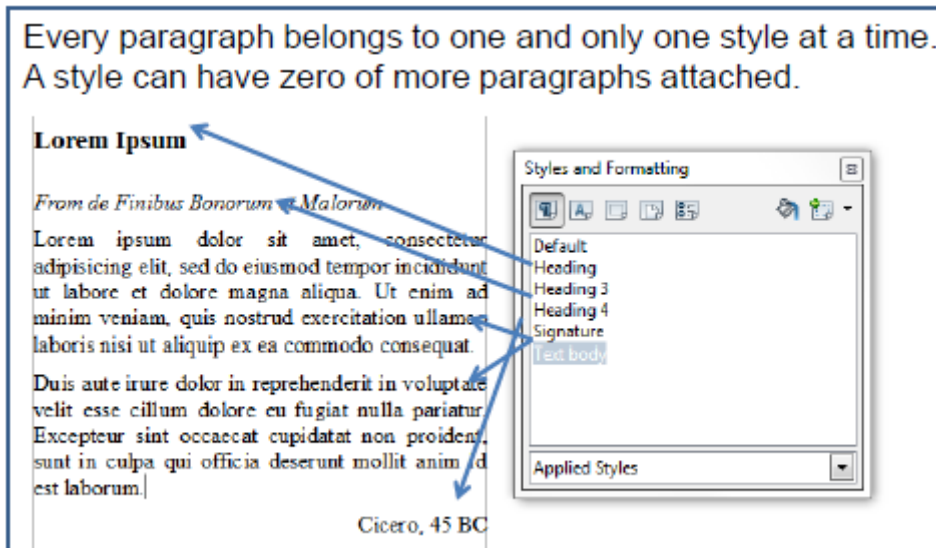


Figure 39. Style-paragraph structure.

The failure of text processors to display underlying structure has been identified as a learning challenge. When entering text in a language written right to left (e.g. Arabic, Farsi, Hebrew and Urdu), one may now and then write a number or an English word left to right inside the text. When shifting direction of writing in a text processor, it also rearranged previously written text in the same paragraph, and users were left in the dark concerning how to fix the situation. In an experiment, one group of users were given a structural model of how the text was organised into blocks, each having its direction of writing (Ben-Ari and Yeshno, 2006). Users having learnt this model thereafter outperformed other users in controlling the direction of writing and the sequence of text fragments. This result points to the effects of structural models for understanding and subsequent problem solving.

4.9. Layers

Information technology processes electrical currents and magnetic charges, but users are not interested in whether a particular circuit inside the box has a 5V charge or not. User interpret the physical signals as symbols, pictures or sounds, which can be information representing something else. Therefore we say that computers are machines manipulating symbols.

A watch is another example of a symbol-representing machine. It may be built from some electro-mechanical parts, but we interpret it as a display of time. While a watch only deals with time, computers can do any type of symbol processing, and for this reason we call them *universal*. The universality is enabled by one of the basic principles of computers, being that data and programs are stored in the same way. This is called the von Neumann architecture (von Neumann, 1945), named after a Hungarian being the scientific leader of a group designing early computers. The von Neumann architecture allows us to insert a new program into a computer in the same way as we enter data. When installing a new program, the computer can do other processing than before, and this is how the universality is realized in practice. Since we can achieve new functionality on smart phones by downloading apps, these

gadgets are also universal symbol processors, hence also computers in the von Neumann sense.

The von Neumann architecture also enables processing the same data with two different programs. This opens for structuring data in layers, where different aspects can be processed in their own ways.

As seen in the previous chapter, the contents of a document constitute one way of regarding word processors, while the formatting enables another view. This can be expressed by saying that the document can be separated in one contents layer and one format layer, and that each of these layers can be changed independently of the other. In web page design, the contents can be structured with html, while the layout can be set by Cascading Style Sheets.

Users who mix up the two layers are likely to do more work when changes have to be made than those who keep format separate from contents. For example, users who add a blank line in order to achieve a format effect, namely larger space between paragraphs with text, will have to change each paragraph. Paragraph formats, including space above and below, can be set by the styles, which is a formatting tool. When changing the style, all paragraphs of that style are updated.

In general, all data can be viewed and manipulated at many layers. For example, if a file is suspected to contain a virus, it can be opened by a text editor, which treats all data as characters, thus internal codes and user data are viewed as being of the same type.

While the deeper layers of the computer software is normally left for the programmers to deal with, having some insight into layers of the internet protocol may, for example, help users understand where connection problems reside.

The hardware layer has some principles which users need to cope with, since they have to grasp the difference between input and output. Some may also have understood that for example, the memory chip in a digital camera is also a general storage for data, so they can use it as a backup for their files while on vacation.

The layered architecture of the computer is a key model to understand the network infrastructure, and how the connection between computers is set up. A *network protocol* is a program with a set of rules for message exchange between digital devices. When the same set of rules exists in two devices that are connected, these devices can exchange data.

In order to transfer anything between diverse devices, there is a layer of protocols. Simple and cheap devices may only be able to receive a stream of electrical pulses and forward these to all other of its connections. They operate on the hardware level only and have the hardware protocol. Everything is electrical pulses for the hardware protocol.

A software protocol recognises internet addresses. When clicking a link in a web page, a transport protocol in the computer is activated, sending a message to the internet address in the link. All network devices which are more advanced than those with only a hardware protocol, distinguish between an internet address (IP, for instance www.google.com) and the

contents of the message. These devices have transport protocols which look up in their list of addresses and forward the message to another network node closer to the destination.

An application protocol will also be able to interpret the contents of a message. For instance, a web browser has Hypertext Transfer Protocol (HTTP) and can therefore display a web page with paragraphs, tables, pictures and links. Electronic mail requires another application protocol.

When requesting a web page through clicking on a link, the request is transported through a variety of connections between the user's browser and the web site. Normally, there is a local area network between the user and an Internet Service Provider (ISP). The local area network may have wireless as well as cabled hardware. The ISP connects its users to the Internet, which again has a connection to the remote web host computer. Figure 40 is a structural model that may guide users to understand the elements to be in place for internet connectivity.

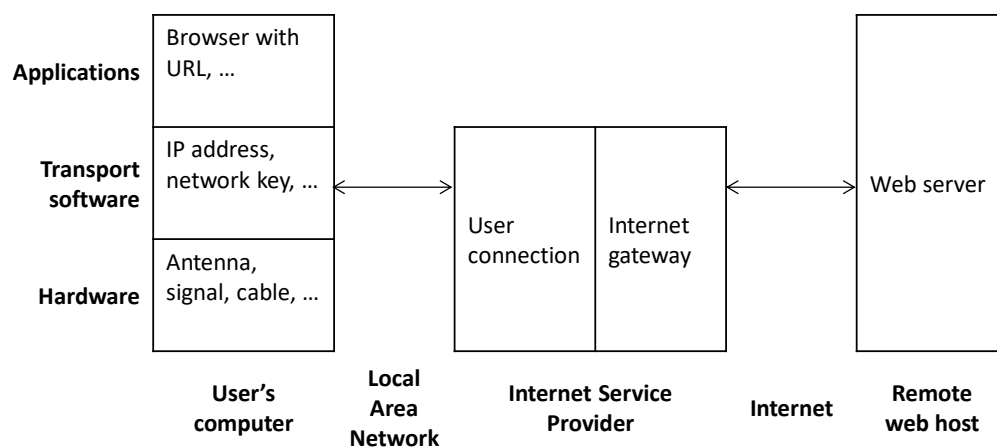


Figure 40. Structural model for understanding the elements of Internet connectivity.

4.10. Semantics

Semantics is the relation between information and the domain, which it represents. When buying clothes in a web-shop, you may be unsure whether their size M fits you. The information is M, and your body size is the domain in this case. Even if M normally fits, you, the sweater in the images in the shop looks small, hence you are uncertain whether you should enter L in the size field this time. A structural semantic model displaying the relation between S, M, L and body measures may have helped the buyer understand the relation between M and body size.

Structural semantic understanding concerns the ability to express the rules and conventions governing the information-domain relation. Janine tells about the reasons for the trend of malaria cases, which is a reason for the information in the figure.

Structural semantic understanding—Janine

Seeing the line graph in Figure 41, she explains:

During the dry season, there are fewer mosquitoes, hence less malaria. That is why the number of malaria cases is low during March to October, and high during the rest of the year.

She therefore has a structural semantic understanding.

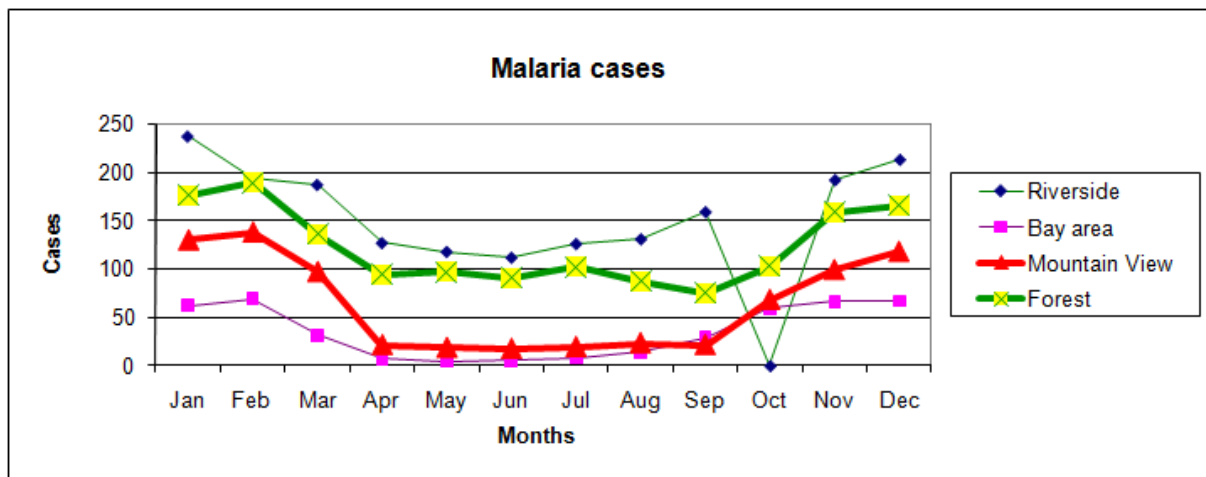


Figure 41. Information representing health issues in four areas in a district.

When learning semantic competence, Janine has to attend to both the information (the numbers in the graph) and the phenomenon being represented by the information (malaria cases over time). The split attention contributes to the cognitive load. If she also were to attend to learning the IT skill of how to correct numbers in a database, the cognitive load would increase even more. Since high cognitive load inhibits learning, separating learning of semantic skills from learning the associated technology skills would ease the learning. The benefits of such a separation is confirmed by research (Chandler and Sweller, 1996). Learners who studied a detailed manual describing the coordinate system (domain) of a CAD/CAM application before practicing it on a computer (IT) performed better on the computer operations than those who followed the instructions in the manual and carried out the computer operations simultaneously. The cognitive load effect also kicked in when learning semantics (Chandler and Sweller, 1996). Those who studied the documentation prior to practicing on the computer performed better during the written test on the coordinate system than those who followed the instructions while practicing.

4.11. Structural and functional misconceptions

As an example, consider the novice learner Herbert, who has just learnt to open programs by clicking at the symbol at the bottom of the screen, for example the Explorer symbol in Windows. Then he learns to close programs by clicking at the × in the upper right corner of the program window, and he observes that the window disappears.

Thereafter, he learns that windows can be minimised by clicking at the underscore character in the upper right corner, and he observes again that the window disappears. So now Herbert is confident that there are two ways of opening and closing programs. He does not recognise the difference between Figure 42a and b, and whenever he pushes one of them, the Explorer window opens with a search engine.



Figure 42. The symbols on the bottom of the screen for a) starting Explorer and b) resuming it after minimising

The lack of ability to discriminate between two different stimuli like these is called a *discrimination error*. Herbert's initial functional understanding of the open-close operations can be illustrated with the diagram in Figure 43.

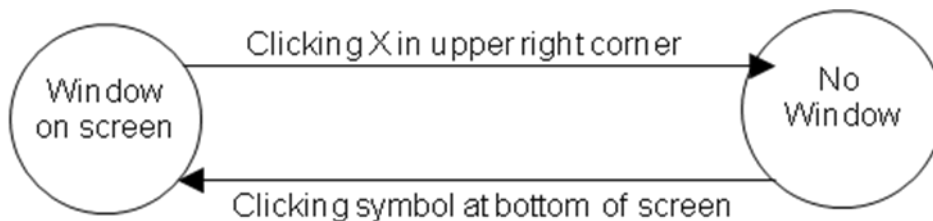


Figure 43. A novice's initial understanding of opening and closing programs.

After learning about minimising, Herbert's functional understanding might have been altered slightly to what see in Figure 44.

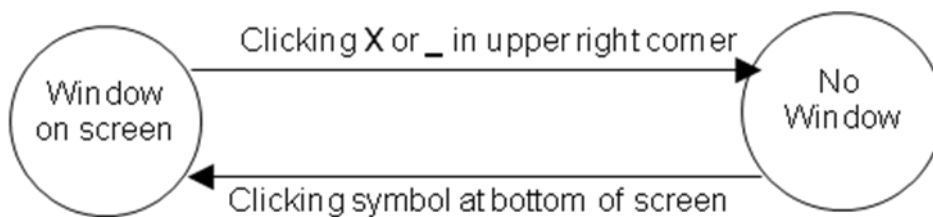


Figure 44. A novice's understanding of opening and closing programs, after having learnt about minimising.

Two factors lead Herbert into this understanding of opening and closing programs. First, the observable difference of the result when minimising or closing is small, and unlike the illustration in Figure 42, these symbols are not displayed at the same time, making comparisons more difficult. Second, Herbert has proceeded from closing to minimising without being aware of a structural distinction, the one between windows and programs. Not knowing that a program can be running even if it has no window open makes it impossible to grasp the idea of minimising. So Herbert has skipped learning one concept which was necessary for understanding the following one.

A functional model that includes the distinction between program and window is shown in Figure 45. In order for Herbert to understand the difference between windows and programs, a trainer or support person may present and explain such a model to Herbert.

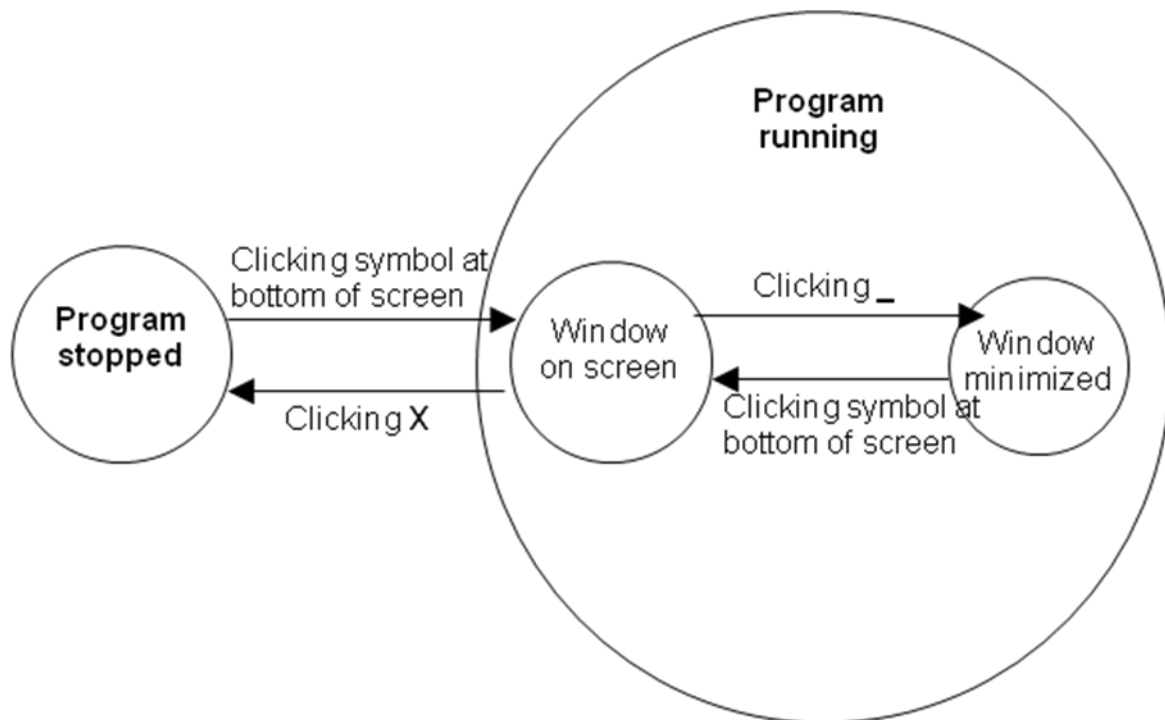


Figure 45. A functional model of opening and minimising which includes the distinction between program and window.

Concepts with related meaning

Consider Fadhili, who might have mixed up two concepts; e-mail address and web page address. A possible way of clarifying the distinction between two concepts is comparing them in a table, for example as in Table 3.

Email address vs. URL—Fadhili:

I typed my email address there.

Why doesn't my mail show up?

Table 3. A model for distinguishing two concepts.

	Web page address – URL – Uniform Resource Locator	e-mail address
Example	www.google.com	fadhili@swamail.com
Purpose	Locate a web page	Identify your inbox as the sender or receiver of an e-mail
Where	Address field of browser	From field or To field in e-mails you send

We also might have to explain to Fadhili that his inbox is not a web page, and that he should rather find out the URL of his e-mail service.

Homonyms

Some terms have two different meanings, for example, ‘well’ can mean a water source or being in good health.

In the case of copying CDs, Oliver’s trouble is based in that the word ‘image’ is a homonym in the digital world. Oliver means a digital photo while Rose talks about a disk image. Again, a table could be used for discriminating between the concepts.

4.12. Summary

Skills are necessary for using IT, but understanding is the basis for learning new skills. Functional understanding means being able to explain that an operation transforms an input state to a result. Structural understanding of what a concept means is necessary for using this concept as a basis for learning new ones.

Functional models provided by people or in documents or videos are scaffolds for achieving a functional understanding. Correspondingly, the conceptualisation which leads to structural understanding is supported by structural models.

Slow learners are in most need of functional and structural models. Such models should therefore be presented in training and be available when users need them at work.

Some learners prefer concrete models with examples and screenshots, while others learn more easily with abstract models. Means for learning should therefore include models with different types of expression.

Misconceptions are often grounded in learners making an analogy to previously known phenomena which do not match the IT to be learnt. Misconceptions can be rectified through someone confronting the misconceptions, and explaining a more adequate functional or structural model.

Copying CDs—Oliver and Rose:

Oliver: Rose, look here on all the photos I have taken. Now I want to burn a CD and send it to relatives, but all this copying is awkward. Do you know any easier way?

Rose: I use Daemon Tool, since it can preserve disk images. You can also make DVDs with it. Let me help you installing.

...

Oliver: Rose, that software didn’t open my images. I had to stick to Photoshop.

3. Provide functional and structural models and confront misconceptions.

Chapter 5. Learning solving IT problems

The learning aim of this chapter is to be able to design activities through which people can become better explorers, problem solvers and learners of IT.

In Chapter 1 the ability to do something was called *competence* and an increase of competence which lasted was called *learning*. It was also noted that problem solving competence was built on understanding, such that the following steps were identified:

1. Skill.
2. Understanding.
3. Problem solving competence.

In the previous chapters, the learning process for IT use competence was presented as a movement from skills to understanding. To complete the learning process, this chapter will present the third level of user competence and the learning involved in reaching it.

Users with IT problems are faced with unknown situations, so they have to learn something new. Hence, the more able they are at learning, the easier they will solve the problem. Since competence is the ability to do something, the ability to learn is a *learning competence*. Getting better at solving IT problems therefore means learning more about how to learn IT. We will therefore call people with good problem solving competence *learning oriented*.

5.1. Learning oriented users

To find out what competence for solving IT problems *is*, we will first look into what good IT problem solvers do and what people say about them.

Kids have fun when exploring new devices and they play together and discover. Nerds do the same with IT, developing skills at a high level, and they are learning oriented in the IT

Psychology – Metacognition

Learning problem solving concerns learning about learning. Since learning is here regarded as a cognitive process (thinking), learning problem solving concerns cognition about cognition, which is also called metacognition (meta = about). Regulating one's learning is an important ingredient in metacognition.

Teachers can influence students' metacognition, for instance by telling students how to take notes, by demonstrating effective strategies through thinking aloud, and by making students work collaboratively (Ormrod, 2012). It has been noted that good problem solvers vary their strategies. If one approach doesn't work, they try another (Schoenfeld, 1992).

For learners with low verbal abilities, also their metacognitive skill of monitoring their own learning was shown to increase when they were explained about structures by means of diagrams (Cuevas et al., 2002). Thus it seems like structural understanding also affects metacognition.

domain. Learning oriented users actively explore the technology, look for better ways of using a program for a certain task, and play around with it in order to see what it can do. The active explorers have a tendency to become local champions, whom others ask for help and who push for new computer applications.

In a study of user competence, Youssou tells about his *learning-oriented* brother, see the text box. The opposite kind of users was also identified in the study, called *performance oriented*. They stick to one way of using a program when they have learnt that way, even though there might be easier ways. They refrain from pushing a button that they have not touched before, due to being anxious for making a mistake or losing data. The anxiety can be regarded as a negative computer learning competence. Phelps et.al. (2001) provide an example of this type of learner too, we call her Ofra.

Learning orientation—Youssou:

My brother is truly amazing. For myself, if something doesn't work I might try it again once but the majority of the time I will just 'give up'. My brother sees these 'failures' as challenges to be met and conquered. He delights in the fact that he never has to stop learning because there will always be a new challenge to conquer. He loves the fact the information technology is such a dynamic field that it is always changing, improving and making new breakthroughs. (Phelps et al., 2001)

A person may be performance oriented in one aspect of life, while learning oriented in another. The stereotypical image of a computer nerd is that he has learnt everything about the computer, but socially, he sticks to what he knows, which is chatting with other nerds. Likewise, the elderly social worker is fabulous in dealing with people, but she has computer paranoia.

Performance orientation—Ofra:

If something goes wrong when I am using the computer I freak out and panic, but when I see these people use the computer they seem to be able to work it out on their own. It is obvious to me that I learn differently to them when it comes to information technology. (Phelps et al., 2001)

The willingness to explore was found to be the most influential characteristic in a study where people were asked to characterise highly competent information systems users (Eschenbrenner, 2010). These users

...try to use IS to its fullest potential ... are not afraid to explore new things.

Learning oriented users seem to explore the technology and solve problems. The ability to explore is therefore a characteristic of competence for solving IT problems. Advanced users who help out others learn from trouble shooting their friends' computers, and they get emotional satisfaction from the experience (Poole et al., 2009):

I just fixed things and learned at the same time....Actually, I remember feeling excited when I first helped someone out.

5.2. Research cycle

In a study of people learning operating a toy car by means of a series of instructions, they had to find out about the syntax and semantics of symbols as well as the structure of the instructions (Shrager and Klahr, 1986). Figure 46 illustrates what the learners had to find out on their own.

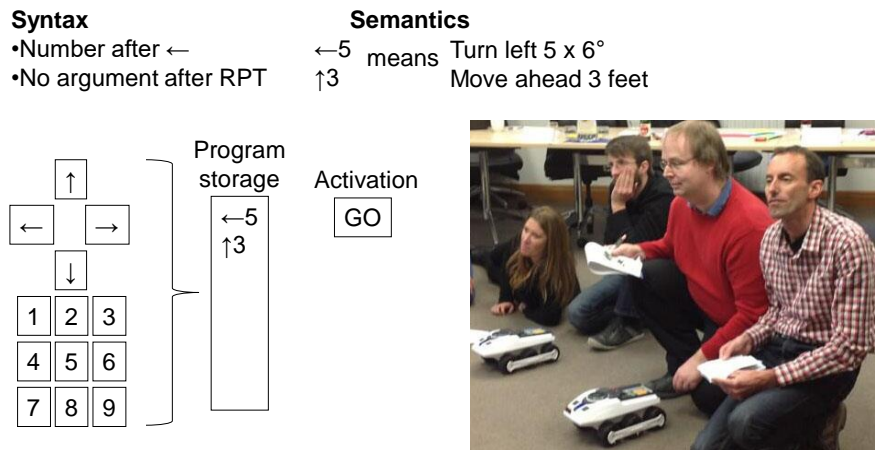


Figure 46. Outline of the instructions of a programmable toy car.

The learners generated the understanding that the car had a memory for storing codes, and that the codes could be activated. To come up with increasingly improved understanding of the car, they followed a research cycle as sketched in Figure 47. Step 1 involves generating a hypothesis, for instance that codes can be stored and plan which input to provide to store code and check whether it has been stored. Step 2 concerns navigating to the right place in the user interface and step 3 entering input. Output is observed and interpreted in step 4 and in step 5 the output is compared to the expected result. The learners completed several rounds of this cycle to gradually build a structural understanding.

1. Generate hypothesis and plan input.
5. Compare output with hypothesis

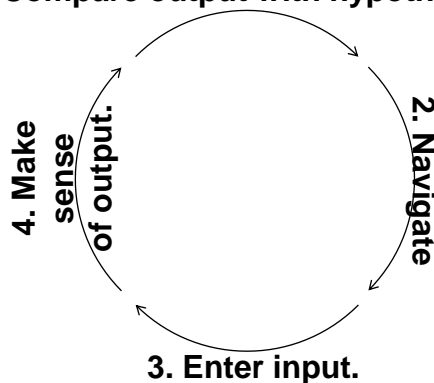


Figure 47. Steps of IT use research cycle. Arrows denote actions to be carried out.

Exploration

Exploration means learning what the IT can do for the enjoyment of finding out something. Exploration involves going through many research cycles and also consult scaffolds for learning. Learning oriented people like Youssou's brother probably searches the web to find answers or send messages to user communities, implying that he draws on scaffolds for learning in his exploration.

In the 'Hole-In-The-Wall' test of exploration (Mitra et al., 2005), computers were set up so that children in poor communities in India could play with the computers without any scaffolds, see Figure 48. Groups of children explored the system. Seen from the individual child, the other children provided scaffolds for learning. Regarding a group of children as an entity, the group was left on their own without any help. In competence tests, the children were asked about the meaning of icons, and a steady progress was demonstrated over nine months. The children had developed both IT skills and exploration skills. Due to the type of tests carried out, we do not know their understanding of the technology.

Even if learning oriented users have the ability to explore, they might choose not to. In a field study of user learning of software, exploring for the sole purpose of learning constituted the exception (Rieman, 1996). This was the case even if some of these users were computer scientists. Reasons why people don't explore is that exploration is unproductive, and that they had too much else to do (Gravill and Compeau, 2008, Rieman, 1996, Bhavnani and John, 2000).



Figure 48. The Hole-In-The-Wall experiment. Computers were installed in poor areas such that kids could explore (Mitra).

5.3. Problem solving

Even if only a minority of users explore, they have to solve IT problems, either through learning their way through it, or getting others to solve it. In the latter case, learning has meagre conditions.

We will distinguish between two kinds of problem solving. Experimentation takes place when we start wondering whether IT can do something that we would like it to do, while we trouble shoot when the technology does not respond as expected. The starting points differ, but in both cases, we may learn how to solve the problem through research cycles.

Experimentation

Experimentation is a planned action of problem solving, starting at the understanding level with a hypothesis of the type “It can do this, but I want it to do that. Can it? How? ” Then the user has to navigate, run the operation, interpret the result and compare with the hypothesis. The experiment thus includes a research process cycle from understanding and back again, as illustrated in Figure 49.

Advanced users experiment a lot. They test the limits of software, for example, “This field is for numbers. Will it take text also?” or “Can the scanner transfer data directly to the phone, or do I have to use the computer as a receiver?”

Not all problems are solved through experimentation. We might nevertheless have learnt something from unsuccessful experiments.

Inadequate functional or structural understanding was found to be a main reason for failure in a study of experimentation (Novick et al., 2009). This was the case regardless of whether the users consulted scaffolds or not. In another study, some users were taught with functional and structural models, while another group was given instructions only. Those who were given models outperformed the instructions group in problem solving tasks (Halasz and Moran, 1983). This again points to the need for understanding in order to experiment.

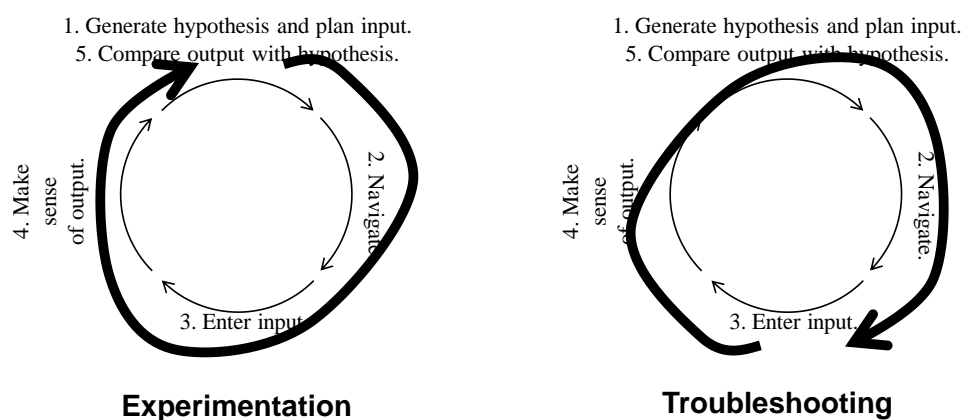


Figure 49. Experimentation and Trouble shooting. Two complete learning cycles with different starting points.

Troubleshooting

Experimentation was triggered from understanding. Troubleshooting is triggered from practice, when noticing that the IT does not do as expected, see Figure 49. We might get help from a colleague to interpret or search the web for an explanation, and in both cases we bring the issue to the understanding level. Through reflection, we might find a possible solution, which has to be tested. Troubleshooting may include the same learning cycle as experimentation, although the purpose differs.

The people learning to operate the programmable car had to trouble shoot when the car did not do what they intended. Their trouble shooting happened through a pattern of first interpreting and thereafter generating hypotheses and testing them (Shrager and Klahr, 1986).

One common way of troubleshooting is to restart. This might do the trick, without the user learning anything about the specific problem. The user might learn that restarting is effective in such cases.

5.4. Understanding as a prerequisite for problem solving

Knowing that kids explore their environment and their toys by playing, and that animals also learn through playing, one could assume that there is no need for teaching people the process of exploring. Sadly, this assumption is wrong, as the case of Ofra illustrates. Users need to learn the competence of exploration, experimentation and troubleshooting, and we will call it *problem-solving competence*. Since this competence is about learning, it is metacognitive competence. We will identify several components of competence for solving IT problems. However, first we will see through an example how understanding ease problem solving.

The findings referred in the previous section emphasized *understanding IT* as an element of problem solving competence. For example, when trying to access a web page, there are many things in different places, which can go wrong.

- Virus or software bug in the web browser
- Wrong web address
- Local antenna or cable from the computer missing.
- Too weak radio signal.
- The user connection of the Internet Service Provider (ISP) broken.
- The external internet gateway of the Internet Service Provider fails.
- The server from which the web page is transmitted is broken.

A structural model like Figure 40 might guide users to an understanding for locating errors.

Antennas, cables, radio signals and transmitters are recognisable hardware devices and properties, so dividing the user's model into hardware and a software layer would be feasible, and it would contribute to understanding at least one useful distinction in the communication infrastructure.

Inadequate functional or structural understanding is a main reason for failure in experimentation (Novick et al., 2009). This was the case regardless of whether the users consulted scaffolds or not. Other studies have confirmed that blind trial-and-error may lead to more errors instead of rectifying what's wrong (Subrahmaniyan et al., 2008, Grigoreanu et al., 2012). Therefore, problem-solving competence requires understanding of the IT.

5.5. *Research cycle competence*

The people finding out how the car was working had the ability to plan trouble shooting by controlling one variable at a time. This is part of the general experimental research competence, which we employ in many aspects of life. Ivo demonstrates how kitchen researchers think:

- He remembers his input of salt last time.
- He has taken notice of and he remembers the output.
- He makes the hypothesis that by changing the input, the output will also change.
- He changes the input of salt the second time.
- He makes sure that all other input, which could affect the output is kept equal.

Research in the kitchen—Ivo:

Ivo is making his mutton stew for the second time, talking to himself:

Last time I added two teaspoons of salt in the pot, and that was obviously too much. Let me reduce to one. Oh, but I also added a stock cube last time, and that is pretty salty. Hm, to find out how much salt is actually needed, I will add the stock cube this time also while reducing to one spoon. If not, I cannot find out whether it is the salt spoon or the stock cube which made the difference.

This research competence is also useful for solving IT problems, and it includes the ability to carry out cycles of IT problem solving phases, we call it *research cycle competence*.

People learn problem solving in the same general way as learning anything else; through doing it and thereafter reflecting on what they did. Problem solving competence also starts at the skill level, implying that users can start learning it by imitating others, whether a trainer or a peer user.

Since blind trial-and-error may increase the number of errors, research cycle competence requires understanding of the IT. Although no research on learning the research cycle in IT problem solving has been found, teaching methods of problem solving in general has a high effect on the learning outcome (Hattie, 2009). Further, teaching which addresses meta-cognitive strategies, such as self-questioning has also yielded large effects (Hattie, 2009). Self-questioning can trigger new hypotheses.

Research has found that pairs or small groups are better problem solvers than individuals, implying that the individuals learn some problem solving skills through cooperation. A laboratory study where learners were given software tasks to be done but no instructions on how to do them, compared individuals and pairs. The results showed that the pairs developed

a better understanding of how the software operated and they performed better in exercises which introduced some novel elements (Lim et al., 1997).

Creating the hypothesis that if a certain change is done, the output will change in a specific way requires the ability to think about future situations, which will be counter to the current experience. Children below 11-12 years normally struggle with the abstract way of thinking when creating hypotheses, see Section 7.6, p.115, for more details.

Figure 50 illustrates the processes of learning the research cycle competence and corresponding scaffolds. Since the research cycle is the general way of generating any type of new understanding, the learning outcome 3b constitutes the insight that this is the way new understanding is also produced for IT users. This places understanding of problem solving of IT use in the meta-cognitive domain.

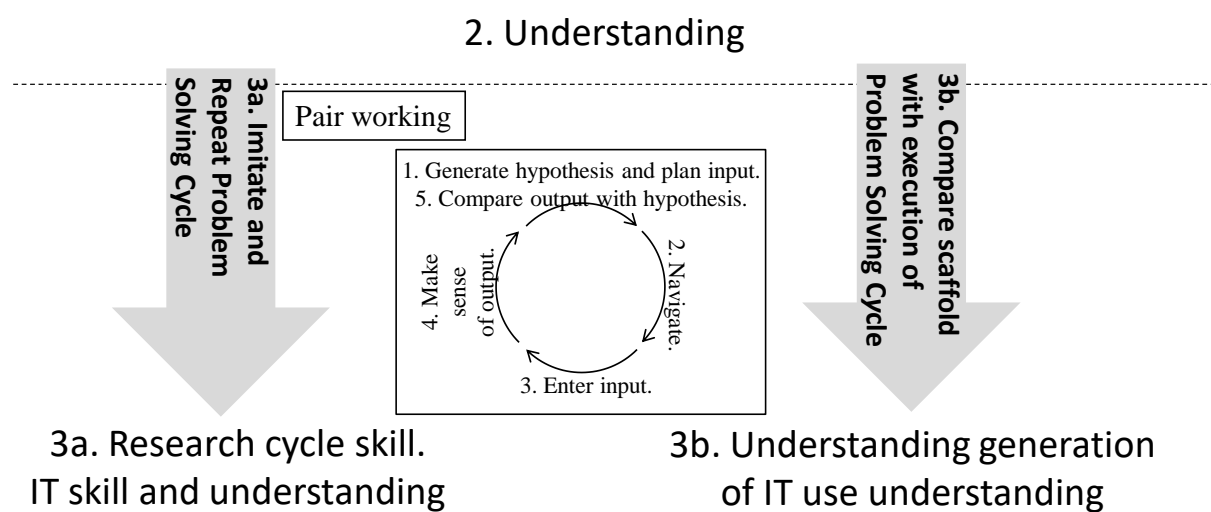


Figure 50. Learning research cycle competence.

Also in line with understanding IT use, users come to an understanding of the research cycle through reflecting on their experience and possibly comparing that experience to scaffolds explaining the cycle. The outcome of that learning process would be that this is the canonical way for developing IT understanding, see right side of Figure 50.

5.6. Stages of the research cycle

Each stage in the research cycle requires specialised skills and understanding. This section will present ways of learning these competences.

Observing repetitive use

IT can offer an easier way for almost any repetitive operation, since a fundamental IT principle is that a series of operations can be repeated automatically. Observing repetitive use is a trigger for generating a hypothesis that additional software, which can ease the current task exists. Research emphasizes that most users keep

Kyung:

Now I have opened each picture in the editor, adjusted its size to the standard and saved it again. This took time. There must be an easier way

operating the computer with inefficient repetition of operations, however (Carroll and Rosson, 1987, Fu and Gray, 2004, Bhavnani and John, 2000), including users who are aware of possibly more efficient procedures. The more efficient process may not provide immediate feedback from the computer and thus may require more cognitive effort for planning than required for the less efficient procedure (Fu and Gray, 2004).

Scaffolds to inform about additional software end convince about its personal benefits are indicated in Figure 51. No scaffold guarantees learning, and the attempts to convince users about more efficient operations may be amongst the less effective ones (Fu and Gray, 2004). If users' previous functional understanding include the series Make details → Aggregate → Manipulate (Bhavnani and John, 2000), cases where this series is applicable may be solved with more ease. Since observing repetitive use concerns stage 1 in the research cycle, this step is indicated amongst the learning outcomes in the figure. The scaffolds are placed in the middle since it can help learning skill as well as understanding.

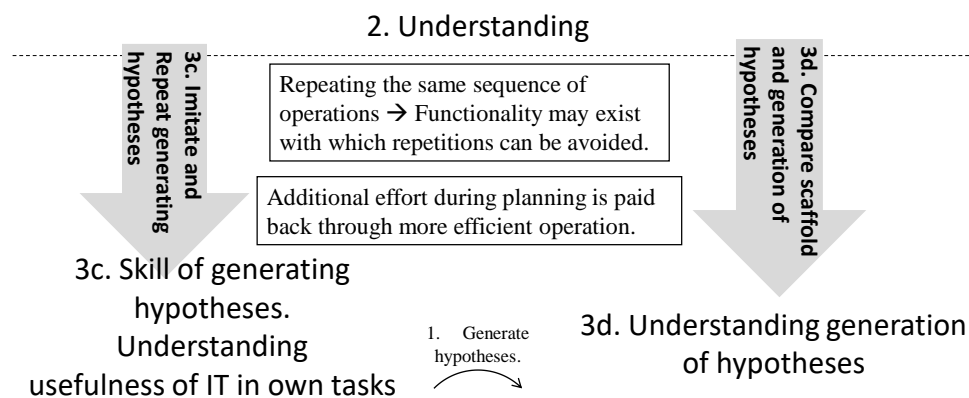


Figure 51. Learning generating hypotheses about the existence of IT features.

Planning testing hypotheses

In the experiment where novice users were learning to operate the toy car, learners planned their testing through making only one change at a time, such that they could observe the effect of the individual change (Shrager and Klahr, 1986).

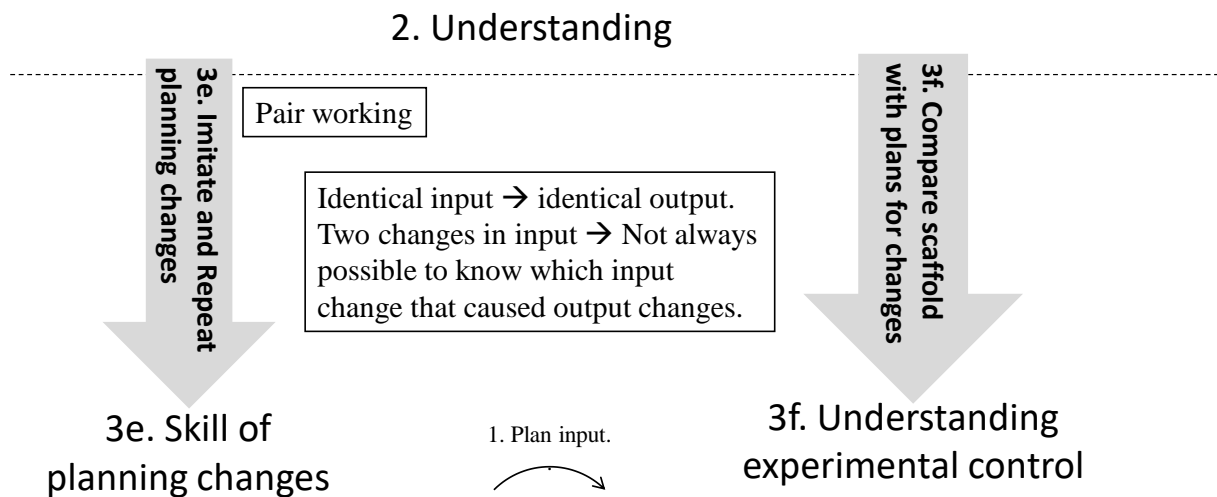


Figure 52. Learning controlling experimentation.

Controlled manipulation of variables is a well-known research principle; its learning is summarised in Figure 52, and a scaffold example is provided in Figure 53.

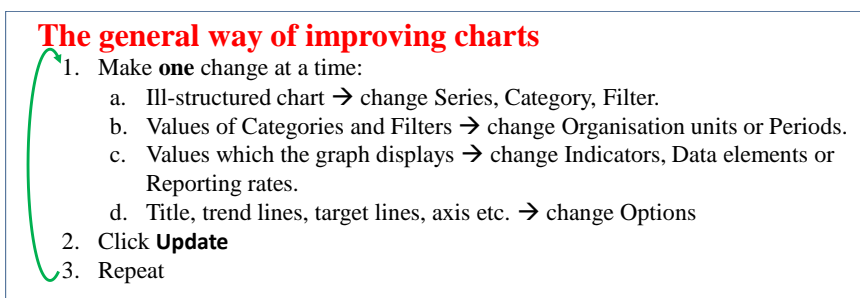


Figure 53. A guideline for experimental control when generating graphs from tables.

Mediating hypotheses

While some users customise software on their own, changes in organisations often require cooperation.

Most business information systems would have access control, meaning that some users have the right to change data, others can read it and some have no access. Access control is a frequent source of issues (Deng and Chi, 2012). Users wanting wider access to reading data will have to understand that access restrictions are the reason why they cannot find the data. Finally, they need the ability to argue for making a superuser or a manager changing it.

Melly demonstrates understanding of how IT systems affect her and the organisation, and how the systems can and should be changed. She is also arguing about a societal issue like the access to data versus privacy.

Melly is also capable of requesting the IT people to change the system. In addition to understanding the misfit, two specific competences are required to make others change the software if you cannot do it yourself.

1. Understanding whom to contact.

2. Having the skill of persuading the IT people to carry out the request.

Even though Melly might have known how to do this, she was unsuccessful due to company regulations. Instead, she developed the workaround for the organisation.

If she had succeeded in convincing the hospital about the changes, she might have cooperated with IT people on redesign of the security function.

An introduction to persuasion principles can be found in (Cialdini, 2001). Learning requesting changes is summarised in Figure 54.

Solving problem of IT fit in business—Melly:

After we got the basic patient information in the computer system, I don't have to waste my time waiting for the patient record any longer. I look forward to the time when the record in the computer is complete with x-rays and attached documents, but we can do most of what we need by the diagnoses and lab info that is there now. When we receive a patient from Jakarta Central, they also send us the patient info, so that before the patient is here, we know what to do.

The main trouble with this system has been the security. You have to log in here and there, and after 20 minutes of idle time, you are logged off. If a nurse has logged on in the meantime, we were stuck, and had to find her to log off before we could access the data. We discussed this with the IT guys several times, but they said it was a policy decision such that medical data should not be spread to those who have no rights to see it. However, in my opinion, it is more important that the medical staff who needs the information gets it than that others are denied access. That means that we give priority to providing the right medical treatment rather than protecting the patient's privacy. The latter will never cure their illnesses. Hence, since the IT people did not help us out, we found out that all staff in the department should have the same password. Thereafter we have had no trouble opening the system when needed.

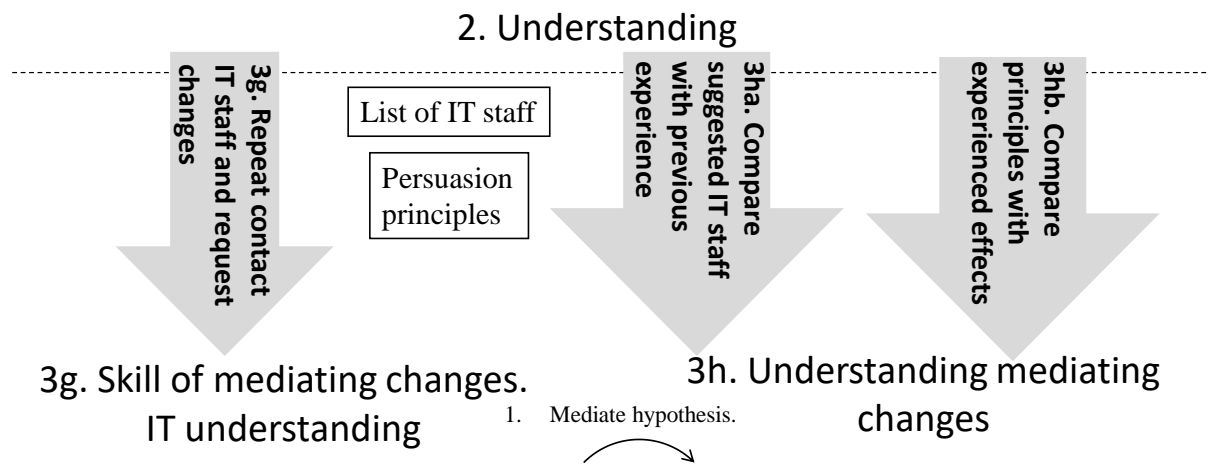


Figure 54. Learning mediating hypotheses.

Changes in software carried out by IT people to cater for a misfit between business and IT requires the IT people to learn from the users and vice versa. Such mutual learning will be presented in Chapter 13, after organisational learning has been introduced.

Systematic interface browsing

When adding IT features to the work, users would at some point have an understanding of the existence of additional functionality but may not know what it is called and where to locate it. The terminology problem is recurring, and the search and help seeking method may be the solution.

Systematic interface browsing is another method, and it can be done in two manners; browsing operations or browsing objects.

User interfaces and web pages are not organised in uniform ways. A window may have tabs and menus for organising buttons and options. Then you would normally select one such operation first, and then select the object that this operation will apply to. We will call this *operation first*. A systematic walk-through of the tabs and menus from left to right could unveil the operation searched for. As previously noted as the terminology issue, the software designers may have used other terms than the user expects, and the user will thus not notice it.

Hovering over a menu item may trigger a tool-tip appearing (see p.95). The description in the tool-tip may be expressed in words that resonate more with the user's terminology, such that the appropriate choice can be made.

The other method for sequencing human computer interaction is the *object first*. This entails first selecting the object to be worked on, for instance a field in a database, some text in a text processor, or an element in a web-page. Hovering over it or clicking on it, right or left, may produce a menu of operations available to manipulate the object.

Figure 55. Learning interface browsing. illustrates the learning processes for understanding and skills for interface browsing. Since interface browsing is a navigation method, this

learning concerns navigation in general, and no particular software. Since navigation in this case is the object of learning, it is placed in the lower middle of the figure..

In order to learn systematic interface browsing, users need to know the difference between operations and objects; hence this is suggested as the scaffold in Figure 55.

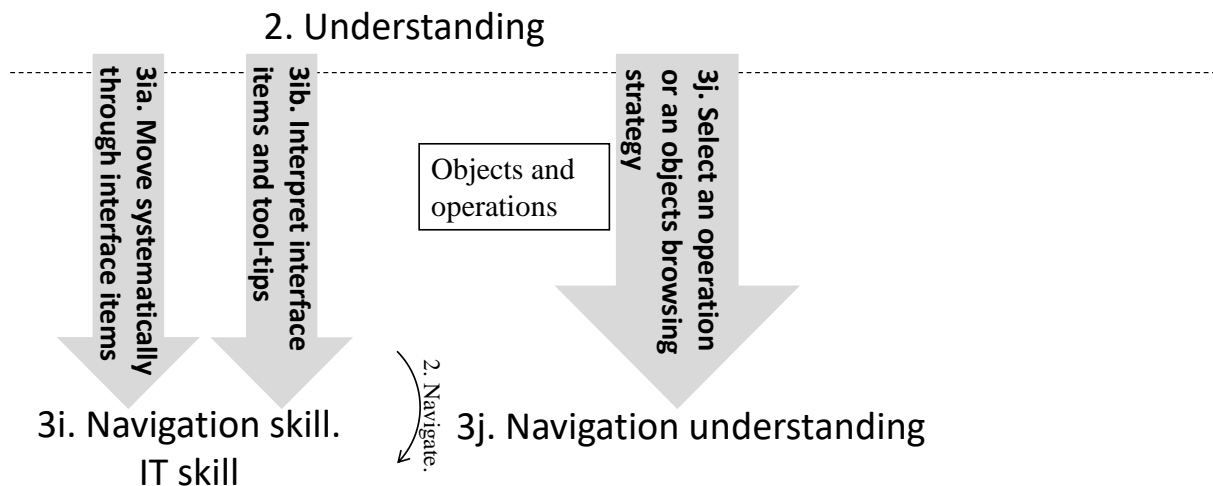


Figure 55. Learning interface browsing.

Self-efficacy

In Section 5.1, fear of the technology was identified as a barrier against problem solving. The opposite of computer anxiety has been identified as *IT self-efficacy* (Compeau et al., 1999), meaning an individual’s belief in her/his ability to perform a specific task using a computer (Compeau and Higgins, 1995). Self-efficacy is therefore another element of problem solving competence, which concerns the user’s ability to carry out the actions, and research has identified it as an important affective competence for IT users.

The most important basis for self-efficacy is one’s own experience (Ormrod, 2012). Previous successes in using IT will boost the self-efficacy, while failures have the opposite effect. A person who has consistently performed poorly with technology is likely to have a low self-efficacy, which will undermine future performance and also increase the person’s anxiety towards IT (Compeau et al., 1999). It may look like Ofra (Section 5.1) is a victim of such a vicious circle.

A trainer could try convincing anxious learners that when things do not work, it is not because they have destroyed the computer. Also, reminding them that there is normally an Undo operation, which can bring them back to where they were could calm their nerves.

Watching peer solving problems—Cheb and Rahel:

Cheb: *I can’t do this.*

Rahel: *Let’s try. I find the menu there and keep the Standard option. Then I paste the image in the right place and ... No. I don’t want all that white space around. Maybe the Standard option was not a good idea. If we click on the image and go back to the menu, will the options show up again? No. Hm. So let’s delete the image and try again from the start. ... I see “in front of” and “narrowly fit.” We want the image above and not in front of the drawing, so let’s try the narrow fit. ... This looks much better!*

One approach towards improving self-efficacy is watching peers. People identify with peers who are believed to have similar abilities as themselves, being colleagues at work or classmates in school. Assume that Chev identify with Rahel in this sense. Watching Rahel's way of working and perseverance as she fixes the problem is likely to raise Chev's self-efficacy, such that he might try himself the next time. Watching a trainer solving the problem is less effective than watching a peer do it (Ormrod, 2012).

Self-efficacy can also be improved by others saying "You can do this. Just try." However, if the learner then fails, self-efficacy would be reduced even further. Therefore, make sure that the task is simple enough.

People collaborating in groups have different competences, and the group as a whole may succeed even though individual members might have failed. Being member of a successful group boosts self-efficacy (Ormrod, 2012). The peer, for example Rahel, and the successful group are included as scaffolds in Figure 56.

Being one of the reasons for people's decision to actually use an application (Compeau et al., 1999), improving low IT self-efficacy is an important part of learning problem solving.

Ways of improving self-efficacy are illustrated in Figure 56. Since low self-efficacy triggers a fear of doing something on the computer, improving the self-efficacy will enable more button pushing and input from the user. Entering input is the main object of learning and is therefore included in the figure.

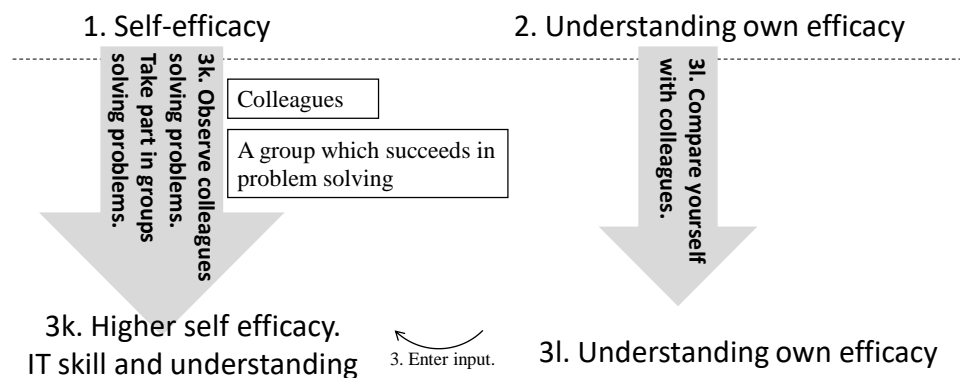


Figure 56. Improving the confidence to operate the computer.

Input checks

Spell checking is a commonly applied method for trouble shooting input in natural language. Correspondingly, a database system may warn when a number is out of the normal range.

While some corrections suggested by the input check are obvious, users also need syntax and semantics competence to allow for non-standard expressions. When troubleshooting input, three choices may be available

- Strict syntax: Adhere to the rule and change input accordingly.
- Semantic correspondence: Make an exception to the rule.

- Semantic correspondence: Change the rule.

The latter may have consequences for future input, while the first choice may imply poor semantic correspondence between the data entered and the phenomenon it is representing. Coming to grips with how to make the right choice implies developing an understanding of troubleshooting input. Manuals and other user documentation may constitute scaffolds for this learning process, see example in Figure 57 and the general learning process in Figure 58.

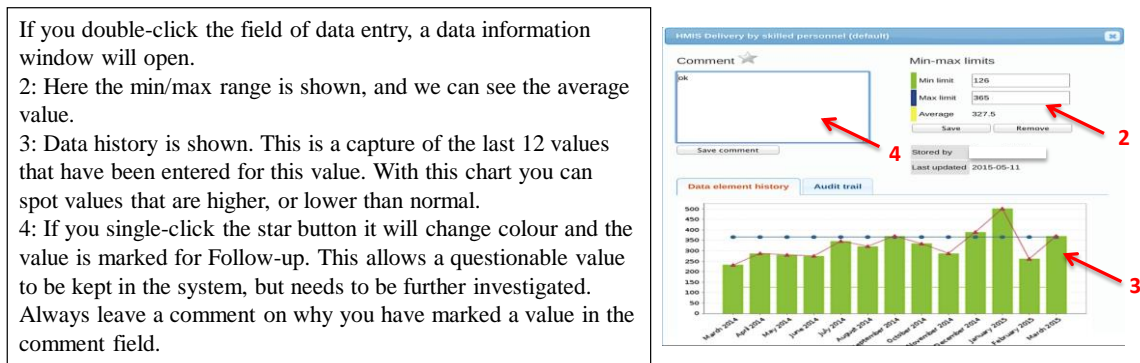


Figure 57. A scaffold for understanding exceptions.

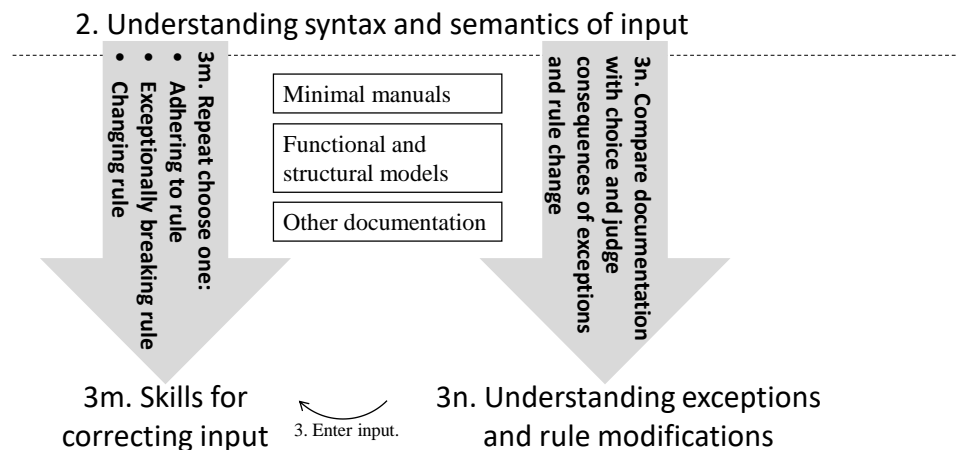


Figure 58. Learning trouble shooting input.

Precise observation

While adding a teaspoon of salt is easy to control, users often do not notice exactly which buttons they push. Fast and erroneous typing may be productive, for instance when writing, since it is often easier to correct a typo than to express an idea in a sentence, and slow typing may inhibit the process of expressing oneself. When experimenting, typos will flaw the logic, however. The ability to watch input precisely is therefore a necessary ingredient in the user research competence.

Imprecise observation—Ksenija and Samira:

In a training course, when trying to repeat an operation which she had done before, Ksenija says:

This worked last time, why did the computer do something else now?

The trainer Samira noticed that this time Ksenija hit F8 instead of F9, while she thinks that she repeated exactly the same typing.

Ksenija would have benefitted from watching her typing more closely. Samira’s response could be asking Ksenija starting over and re-typing. If the computer performs as Ksenija expects this time, Samira could bring up the issue of observing precisely what one is doing before blaming the computer. Also, Samira could also have told Ksenija that in such cases, a useful method is restarting and doing the operation slowly and carefully, such that one makes sure that the input is correct.

Correspondingly, users do not always take notice of the output. When calling support and saying that the computer failed, the supporter will return with the question “Exactly what happened?” If the user has no answer, there is little hope of finding out anything, unless the flaw can be recreated. Therefore, the ability to watch output precisely is also necessary for problem solving competence. We combine the input and output and say that the ability of *precise observations* is needed for solving IT problems.

Precise observation of input implies watching the buttons pushed on the keyboard in addition to what is happening on the screen. Precise observation of output includes the abilities to note down what happened, copy error messages and take screen shots, both of intermediate and end results.

In line with other teaching of meta-cognitive strategies, teaching awareness of inconsistencies is effective in general (Hattie, 2009). Observation ability is an element in becoming aware of inconsistencies.

In the experiment on trouble shooting spread sheets, users who observed precisely were able to generate appropriate hypotheses and interpret the outcome of changes (Grigoreanu et al., 2012).

Figure 59 summarises learning precise observation. Since precise observation concerns steps 3 and 4 in the research cycle, these steps are indicated amongst the learning outcomes in the figure.

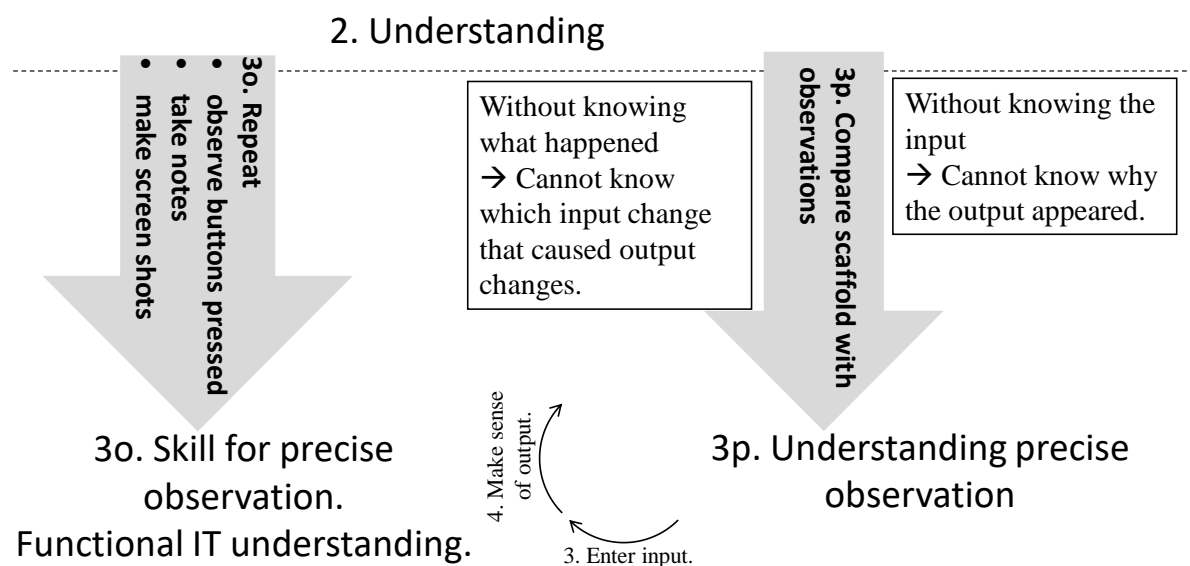


Figure 59. Learning precise observation.

Information search and help seeking

Users' choices when trying to solve problems have been found to be (Novick et al., 2007):

1. Experimentation or trouble shooting without any scaffold
2. Asking others for personal help
3. Giving up
4. Looking up inline help and searching the web
5. Looking up in printed manuals

This is a ranked list based on several studies, where number 1, problem solving without scaffold, counted for approximately half of the cases. Printed manuals were hardly used at all.

A study of effectiveness of problem solving compared three methods (Andrade et al., 2009):

- Experimentation without any scaffold.
- Only consulting an inline help system.
- Switching between experimentation and consulting the inline help.

The combined method was the most effective. Comparing with what users do when solving problems, we see that people in general do not choose the best way of solving problems.

Consulting inline help is one possible way of finding scaffolds. This way has traditionally been regarded as *information search* while *help seeking* denotes asking other people. Both of the abilities to search and to ask have been established as metacognitive skills, which are useful for learning. Asking others has the benefit of getting the response tailored to the needs after a dialogue on what exactly the user wants. However, those being asked may not know the answer and may have to search the web or look up in the IT department's knowledge base of user requests. Asking for help may therefore become a mediated information search. Due to the massive amount of information on the web, searching with the text from an error message or a precise description of the problem may provide the wanted response immediately. In addition, searching the web or inline help in the software may give access to repositories of user support communication or an e-mail discussion in a user group, both of which could be responses from human helpers to a previous user having the same problem. While previously regarded as distinct, the two ways of finding means for learning can therefore be regarded as a continuum from no (encyclopaedia) to complete (human expert) adaption to the learner (Puustinen and Rouet, 2009).

For help seeking, the user needs to know whom to contact. This could be a friend or colleague, an appointed superuser in the work place, an IT department or the vendor. Users approaching a software vendor's support service may receive the message that the error is due to network problems, and they have to contact the network provider instead. They may deny any error and send the user back to the software vendor. Getting a non-commercial advice may be needed in such situations.

Inline help and search engines on the web can possibly deliver useful means of learning. A comprehensive account of people's search behaviour is found in (Case, 2012). Search is often needed, and the problem solving skill of precise observation comes in handy when searching for help for trouble shooting. Typing or pasting the error message as the search term will often

provide an explanation and possibly also a way out. The next challenge for the user is interpreting the explanation.

Figure 60 is an example of a scaffold for interpreting search results. It points to specific traits in the hits hinting at the price to pay (free), what to do (download a program), who supplies the information (Microsoft support), and which software to use (Media player).

For help seeking, the user needs to know whom to contact. This could be a friend or colleague, an appointed superuser in the work place, or an IT department or the vendor.

The challenge when searching for information for experimentation is often the terminology problem (see Section 2.2, p.21). Searching the web is more likely to hit relevant directions and instructions than inline search due to the richness of expression on the web. Also, chances of success when searching documentation decreases. This terminology trouble has been confirmed by other research. Many of the participants in a study of problematic use episodes were not able to find the functionality which they knew existed (Novick and Ward, 2006).

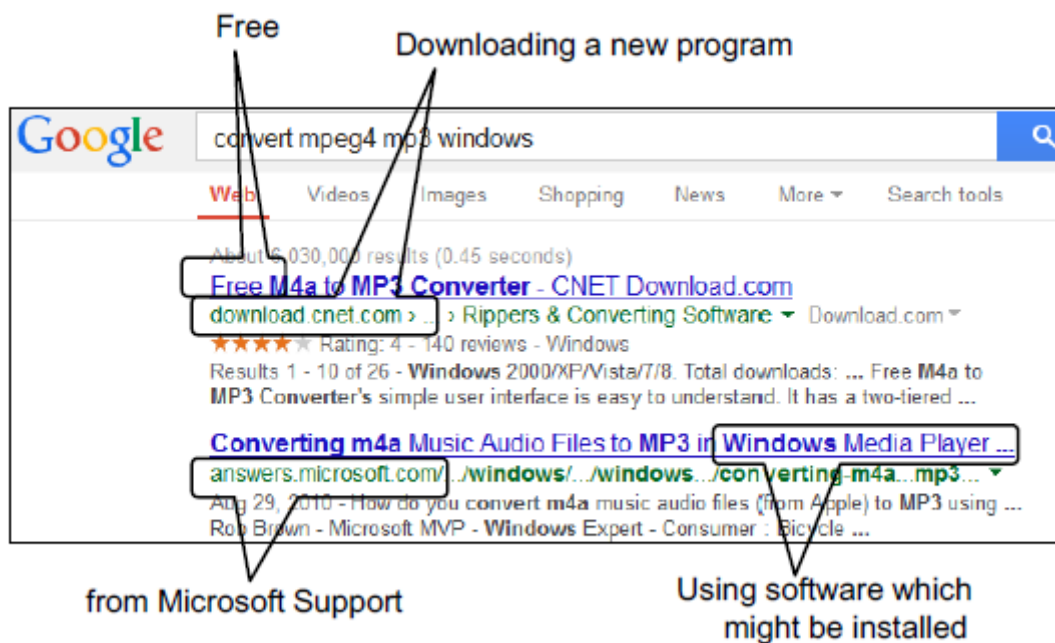


Figure 60. A scaffold for learning to interpret search results.

When turning to the documentation, they reported additional trouble, since they also could not find the right place in the documentation. Often, their search terms did not match the keywords in the documentation. This causes a challenge for writing directions, where the functionality should be described with the terms that users know. For instance, the search terms “reference table document OpenOffice” would not find the directions and instructions in Figure 2. To be more searchable, the page could be equipped with more terms like “inline link, reference, hyperlink, pointer, automatic update, ...” Figure 61 illustrates the general outline of directions. The user interface item could continue with a sequence of instructions.

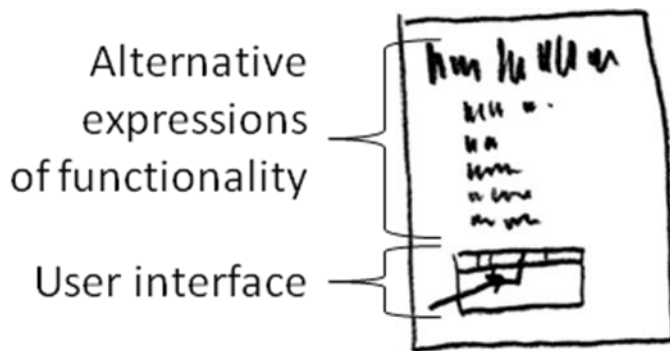


Figure 61. Directions consist of functionality expressed in many ways plus the place in the user interface to trigger the functionality.

In order to overcome the terminology problem, directions generated by a community of many users may increase the likelihood that a user's search term hits one of the other users' help. This is the web solution, which was the most frequently used type of help amongst 107 users (Martin et al., 2005). Half the users had problems with finding what they were looking for, and half of them had trouble interpreting the documentation found.

For business internal systems, the www may have little to offer. Instead, user questions and responses can be made available for searching also inside the interface of the system, so that the threshold for use is as low as possible.

Methodological training of information search contributes to search competence (Walraven et al., 2010). Pairs demonstrate in particular a richer repertoire of search strategies (Lazonder, 2005).

Finding alternative search terms is in general more difficult for younger children. They rarely consider synonyms when failing in their search (Bilal, 2002, de Vries et al., 2008).

The learning processes and scaffolds for search for help and help seeking are illustrated in Figure 62 and Figure 63.

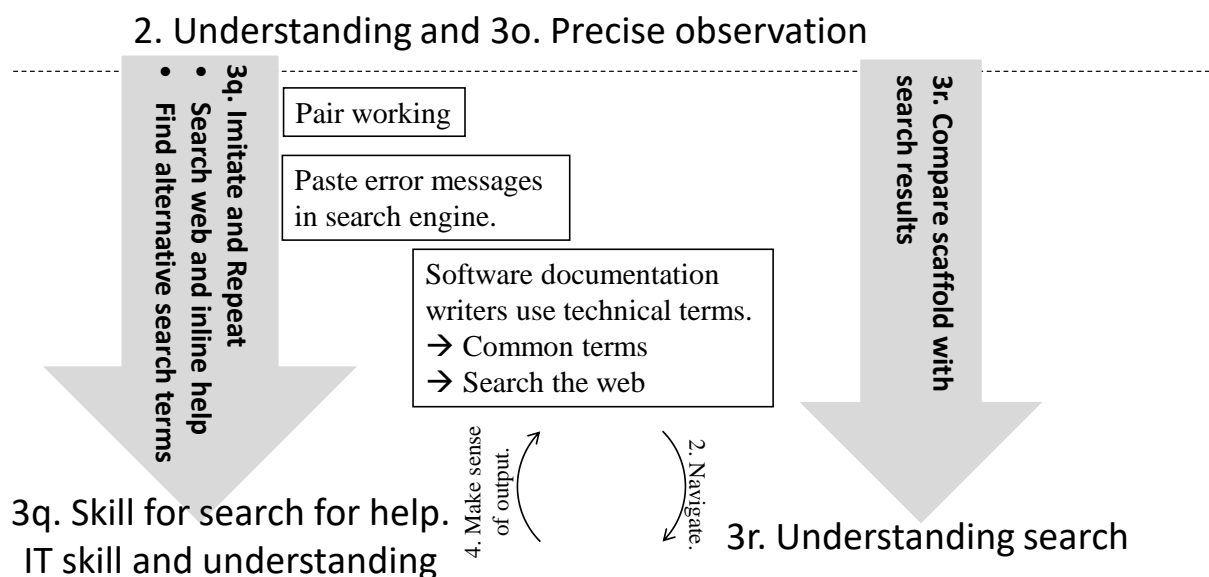


Figure 62. Learning search for help.

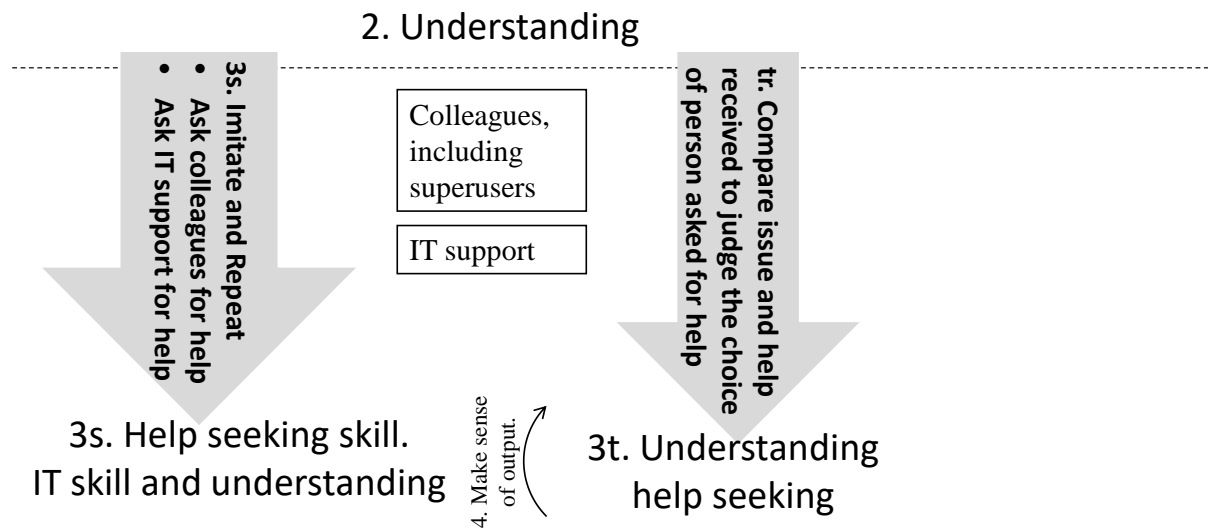


Figure 63. Learning help seeking.

5.7. Strategies for iterations

Several research cycles may be necessary to find a solution, hence competence on organising a sequence of cycles will be presented in this section.

Backtracking

When errors are detected, a chain of events might have occurred. Backtracking is one possible strategy for managing the sequence of research cycles needed for troubleshooting. The troubleshooter identifies the immediate computer operation producing the error, tests this in a research cycle, and if found OK, proceeds to the operation prior to this one, and so on.

Elimination

Elimination is a strategy that speeds up troubleshooting through eliminating correctly operating components. For instance, when the internet browser fails to load a page, first checking the browser, thereafter the network connection on the computer, and so on, is a tedious strategy. Whether these components work might be determined by trying to open another web page. If this succeeds, we have eliminated browser and network faults from further test.

Explicit teaching of the principles of elimination coupled with training for understanding has shown improved elimination skills (Gugerty, 2007). The scaffold should convey the principle that to eliminate searches for errors, the user should check whether possibly faulty components deliver sound output elsewhere.

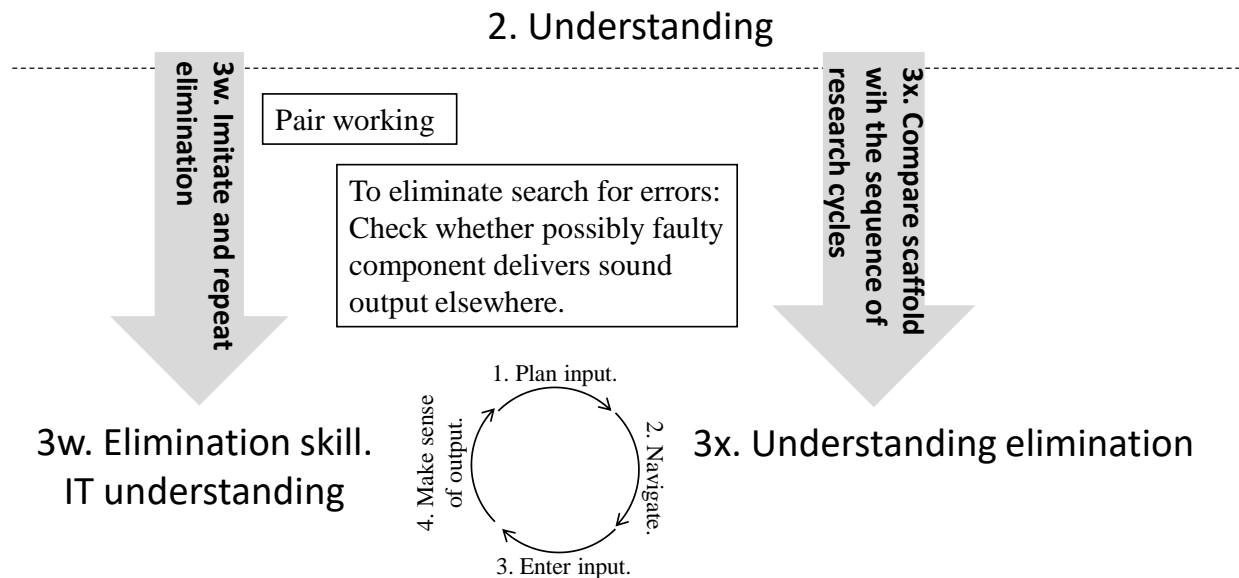


Figure 64. Learning elimination.

Figure 64 illustrates the scaffold for learning elimination. In order to eliminate faults on the internet connection as mentioned above, an understanding of the components and interaction similar to the structural model in Figure 40, p.60, would be needed.

Changing work routines

Often, the introduction of new information systems in organisations also aims at changing work routines, both individual and cooperative. The new work routine would constitute an additional task, which will replace an existing one. *Understanding the purpose* increases the rate of adoption, and adoption would in this case imply changes of work routines.

For example, an engineering business needs traceability of its operations in order to clarify responsibilities. Therefore, a work flow system is installed, which requires that a customer request is first registered at the front office and thereafter forwarded to the engineer. Previously, the customer might have called or sent an e-mail to the engineer directly. This change of work routine requires three people to learn the new way of working; the engineer, the front office clerk, and the customer. While the individual adaptations are simple, such changes of organised work often go wrong due to one of the participants keeps going in the old fashion.

A prerequisite for learning new ways of cooperation is to *understand its purpose*. This means that in the engineering case, both the engineer and the clerk have to understand why traceability is important for the company. The way to learn new routines is to have the

- involved people practicing and discussing the new cooperation,

such that the engineer can tell the clerk about particular customers, and the clerk can inform the engineer about the regulations on who is allowed to do what.

A more difficult learning challenge in the example is to change the customer's behaviour. Although the customer may understand the company's need for traceability, they may be more interested in efficient communication, therefore nevertheless trying to contact the engineer directly.

Repeated, consistent feedback is another mechanism which leads to change of behaviour (Ormrod, 2012).

- Repeated, consistent response according to the new routines

from the company would eventually change the customer's behaviour. If the customer finds the new routines too awkward, there is a risk that the customer will hire another engineering company instead.

Thus, managerial and collegial feedback reinforcing new routines while extinguishing old behaviour, would be ways of changing the routines; summary in Figure 65.

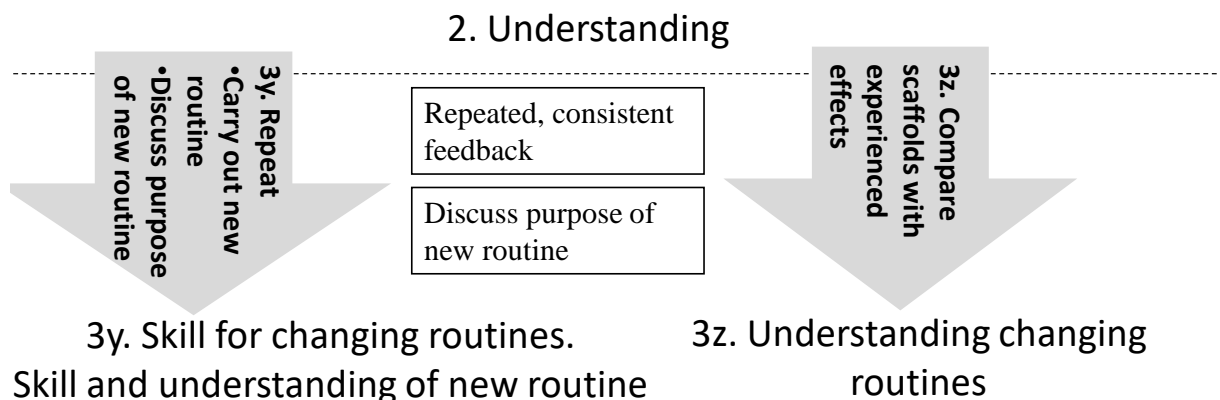


Figure 65. Learning new work routine.

5.8. Innovative research cycles

Misfits between what the IT can offer and user needs can be rectified through changing the IT or using it in innovative ways. Four types of user innovations are following here. The fifth, which requires cooperation between users and developers, will be presented in Section 13.3.

Customizing

Creating reports in a business information system is an area which often requires customization (Deng and Chi, 2012). Most software offers options for customising, which could make the IT address an existing task in a better way or open for using the IT in new tasks. Customising normally requires setting parameters and selecting options, in other words, IT skills and understanding, as also observed by Deng and Chi (2012). The customization process needs to come up with the requirements, and then the research cycle and the navigation method could address the implementation work.

Installing new software

A misfit could be resolved by extending the functionality of a computer system, either stand-alone programs including phone apps or add-ins to increase the functionality of existing software. A user said:

I just use plugins or widgets that are user friendly. I just have to download, put a link and that's it. (Bagayogo et al., 2014)

The IT might be extended by installing new software, either stand-alone programs including apps or add-ins to increase the functionality of existing software. Installing new software utilizes the universality of computers. In addition to understanding this principle, users may need information search competence to find the appropriate software and check its reputation, see Figure 66.

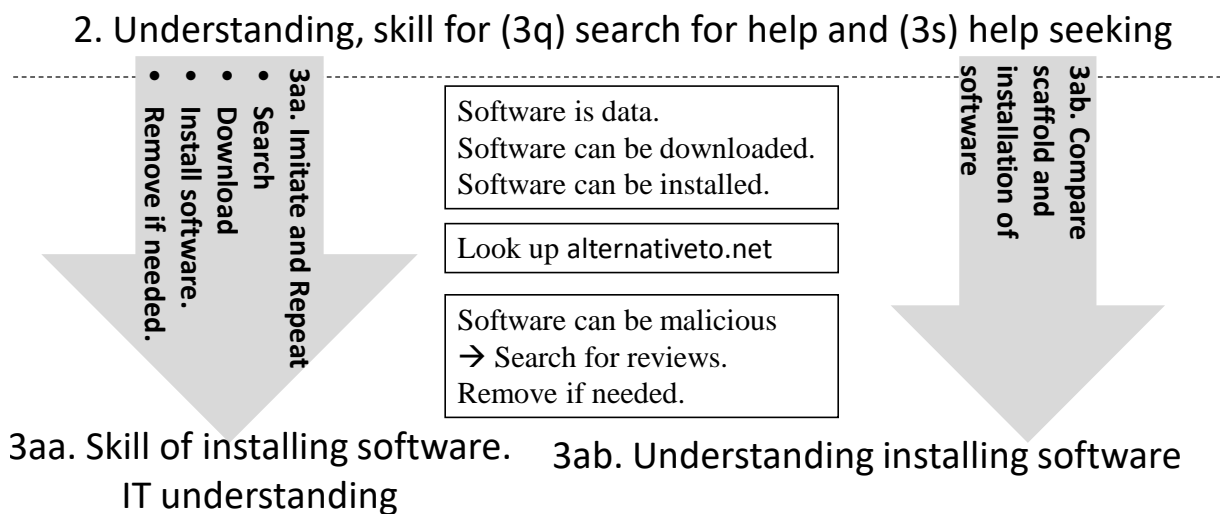


Figure 66. Installing software.

Introducing new devices

New devices like phones, tablets and sensors of various kinds can be beneficial for current work tasks and provide support for new tasks. Software like drivers may have to be installed on the host computer. The new device may also be in need of software to be used as intended. While a smart phone is a universal computer in the sense that it stores programs as data and can process programs, a sensor may have a fixed pattern of behaviour. Users would need to understand this difference, see Figure 67.

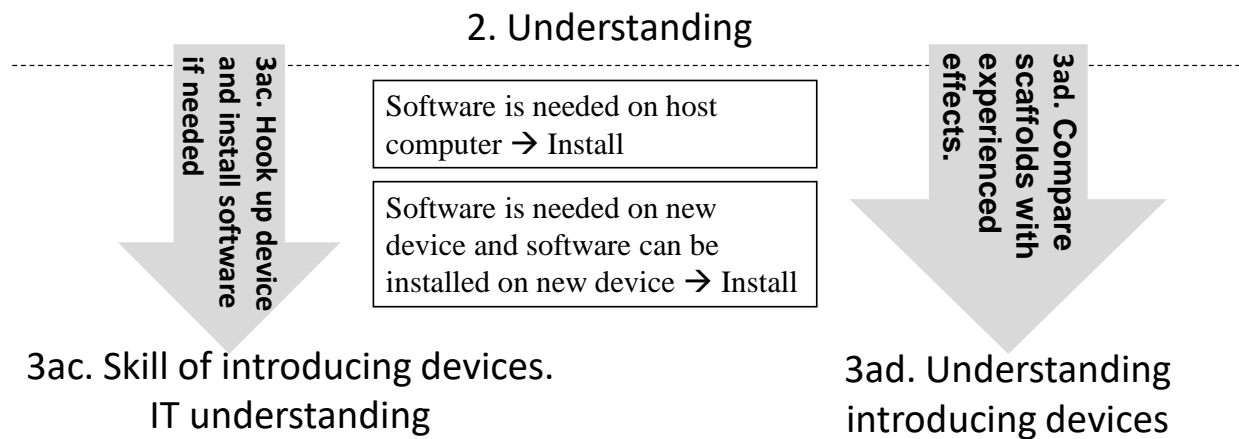


Figure 67. Introducing devices. New devices may or may not be universal computers.

Workaround

Mismatch between the domain and the data fields often occurs in information systems. Angelique is trying to buy something online. Her experience is common, since domestic conventions are not necessarily valid elsewhere.

Solving problem of IT fit in business—Angelique:

That web-shop requires me to fill a field called State, but Benin is not divided into states. Since this address data is probably going to be glued to the package, I'd better not mislead the post office. Just let me repeat the city name Cotonou.

Angelique has a semantic

problem of filling a State field when there is no such thing as a state in her country. The task which the information is used for, distribution by mail, determines her choice of data to be entered. She is using her understanding of the postal services to fit the information in the system to the business. Such problem solving which includes using IT systems in unintended ways is called *workaround* (Gasser, 1986). The semantic mismatch between the data “Cotonou” which is a name of a city, and that it is stored in a field called “State” is accepted by Angelique as well as by the information system.

Solving a misfit with a workaround requires two steps.

1. Based on the business need, can the IT produce an acceptable result? The city name repeated on the package label would constitute an acceptable output according to Angelique’s judgement.
2. Can we make the IT produce this result, possibly by using the technology in ways that were not intended? Entering the name of a city in the field for State made the trick for Angelique.

Step 1 requires an understanding of the fit between the output of the IT and the task requirements. In Angelique’s case, understanding IT in her own task was sufficient. Step 2 requires a functional understanding of the input and output of the computer system

Practicing workarounds requires experimentation skills. Since manuals and inline help tend to be kept aligned with the software and its intended use, chances of finding workarounds in

such documentation are small. Searching the web or help seeking might yield more useful results, since these problem-solving strategies give access to other users' experience.

See summary of learning workarounds in Figure 68.

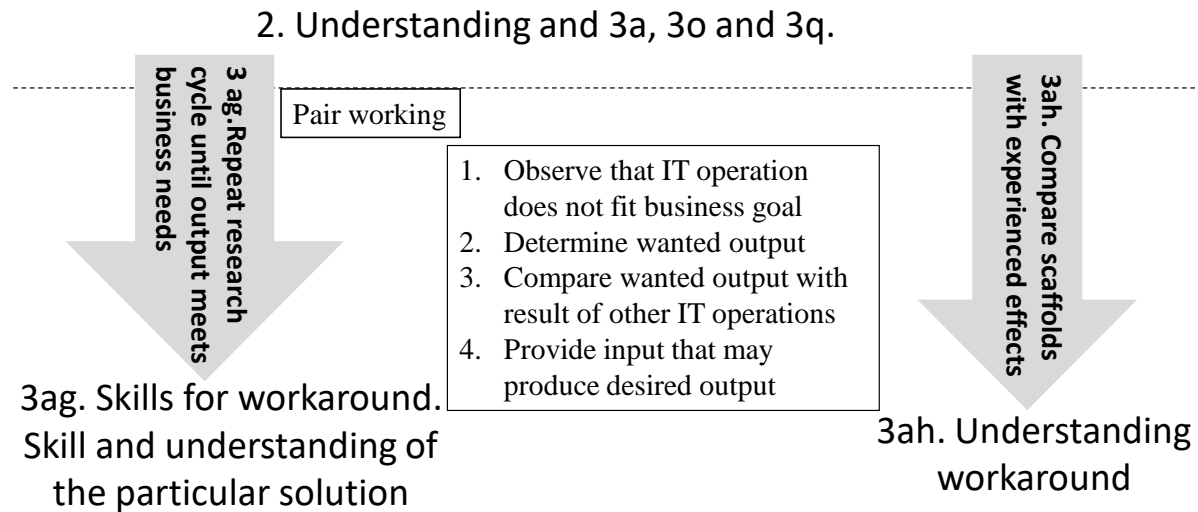


Figure 68. Learning workaround.

5.9. Summary

Users who are better able to learn on their own become more proficient and require less attention by support services. Users who can solve problems on their own are also fitted for becoming superusers of support personnel who can help others.

Table 4 provides an overview of problem solving methods. Research cycle is the general method applicable for all problem solving and exploration activities.

Table 4. Problem solving methods.

Area		Problem solving methods
Research cycle		Exploration
		Experimentation
		Troubleshooting
		Restarting
Stages of the research cycle	Generate hypotheses	Observing repetitive use Mediating hypotheses
	Navigate	Systematic interface browsing
	Enter input	Self-efficacy Input checks
	Enter input and make sense of output	Precise observation
	Navigate and Make sense of output	Information search and help seeking
Strategies for iteration		Backtracking
		Elimination
		Changing work routines
Innovative research cycles		Customizing
		Installing new software
		Introducing new devices

	Workaround
	Mutual learning

Learning these methods improve users' ability to cope on their own. Learning problem solving builds on understanding the topic. Some of the problem solving methods also build on the specific method of searching for information and seeking help from others,

4. Train users so that they can solve problems and learn on their own.

Part II User Interface and Training

The previous part of the book has discussed learning of IT use and presented scaffolds for particular learning processes:

- Instructions for imitation.
- Minimal Manuals for understanding the purpose of the IT and for imitation.
- Business oriented models for understanding usefulness of IT in the business.
- Functional models for comparing input and output.
- Structural models for conceptualisation.

A three level model of competence building was also presented.

Based on these learning processes and associated material, this part will consider how the scaffolds are organised into helpful configurations for user training and design of user interface for ease of learning.

Pedagogical theory – behaviourism

Within a training and transfer view of user competence, the outcome of the learning process, which takes place during training is our focus. This view of learning is in accordance with the behaviourist approach, where learning is considered a relatively stable change of the potential for action. That means, after learning, the learners should be able to do things that they could not do before, and that this ability is not a random change. Being able to do something does not necessarily imply that it is done, since required conditions like time and money might not be present. The behaviourists only consider observable behaviour, meaning that what goes on in people's head is outside the area of interest.

The typical way of regarding learning in the behaviourist perspective is that a person is presented with a *stimulus* from the environment, for example Arja's computer displaying a spreadsheet table and a document. Thereafter Arja responds to the stimulus, for instance by importing by a link. If this *response* was different from the previous ones, and also that Arja continues with this response later when she is presented with the same stimulus, she has learnt a new behaviour.

After a response, the person can receive a new stimulus, which can reinforce the learning, for example that the numbers in the document are updated according to changes in the spreadsheet. If seeing this makes her more inclined to import by link the next time she sees a spreadsheet table and a document, then the updating constitutes a *reinforcement* for her learning.

Chapter 6. User interface for learning

The learning aim of this chapter is to be able to design and evaluate software for learnability, and in particular design in-line help.

6.1. Learnability

Seen from a user point of view, IT applications can be considered having the qualities of learnability, usability and usefulness. This chapter will focus on learnability, but the other qualities will be mentioned briefly first.

Usefulness concerns effectiveness, meaning the quality of the result of the IT operations compared to the business needs. Low effectiveness means that there is a misfit between the IT and the business. Usefulness can also be evaluated by asking users about their satisfaction with the results that the IT is producing. When alternative IT systems are considered for a part of the business, users can be asked to rank the systems on different outputs.

Perceived ease of use, as described in the technology acceptance model in Section 3.1, will be divided into usability and learnability. Usability concerns the efficiency with which skilled users are able to use the IT for tasks in their business. Efficiency can be measured in time to carry out a set of operations, in the number of mouse clicks and keys to be pushed, the number of screens to be opened, etc. Another way of evaluating usability is asking users about how satisfied or comfortable they are with the IT.

Learnability concerns the ease with which people develop user competence for the system. This has traditionally been regarded as bringing the novice up to having the skills for using a system (Nielsen, 1993). Later, also more advanced learning has been taken into the learnability concept, and the ability to improve competence has also been included (Grossman et al., 2009). The latter is what was called metacognition or problem solving competence in Chapter 5. As seen in Part I, user competence can be at the skill, understanding and problem solving levels. The learnability of IT is therefore considered as the ease of climbing up one competence step. A large software package can have many modules and functionalities, some which are easy and other which are difficult to learn. Learnability might therefore address only parts of an application and some business tasks.

Users also differ concerning their ability to learn. First, some learn IT use quicker than others. Second, some users know, for instance, more about the information in the system than others, such that they learn the data structure earlier than others. The learnability of an IT component may be measured by the average time to learn for a group of users.

Deciding what to be learnt is also necessary. In case of a mobile phone app, the majority of users probably only need the skills of using it. For a business information system, some need to develop problem-solving competence, but the majority may also here be satisfied with a skill level. Therefore, measuring time from first encounter to skilled use may be a relevant measure of learnability for many systems.

For network components and printers, which seem to break down more often than other IT, learning trouble shooting may be the learnability aimed at.

If introducing a general communication system for cooperative work, there is a risk that users do not understand the purpose of the system for their work. Without understanding the usefulness, people are less willing to learn, as seen in the technology acceptance model. The effort needed to understanding usefulness is therefore an important aspect of learnability.

For systems, which are used intermittently, maybe once or twice a year, users tend to forget how to operate them. A relevant evaluation of learnability for such systems is time required for recalling the skills.

6.2. Design for learnability

This section introduces user learning issues to consider when developing IT. As previously stated, usefulness is in general a more important quality than learnability in order to for a new system to be accepted, and designing for usefulness is described in textbooks on IT design and information system development. Textbooks on design of human computer interaction may cover usability. When designing data structures, functionality and interface, learnability needs to be considered alongside usefulness and usability.

Reinforcement

The behaviouristic learning theory states that feedback from the IT on user actions may constitute a reinforcement of user learning. Positive reinforcement like appraisals and negative reinforcements like the disappearance of an error message will both strengthen learning.

Immediate reinforcements are better than delayed ones. Long response time when using a web system may leave the user doubting whether the input was correct or the connection is poor. Informative reinforcements are better than uninformative ones, also called *extinctions*. For example, changing directory in the command style user interface might produce a feedback like Figure 69 a. It does not inform the user whether the operation has been successful or not. Consistently repeated extinctions will in general wipe out competence. Users of command interfaces may avoid this by watching the effect of the following command.

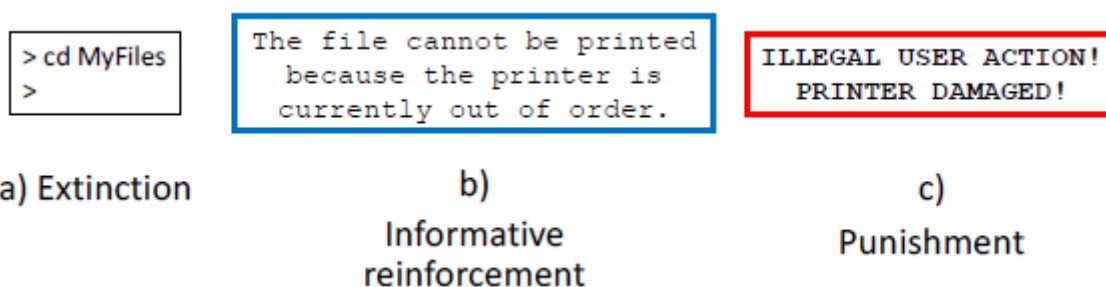


Figure 69. The effect of feedback on learning.

The message in Figure 69 b is informative, since it provides a response about what happened, and it even gives the reason for the outcome. Hence, it strengthens learning. *Punishments* weaken learning and are the opposite of reinforcements. For instance, the message in Figure 69 c would, for most users constitute a punishment, making them less likely to try the print command again.

Consistency

Since learning new things is based on what we already know, learning is eased if the new thing resembles the old ones. Hence, learnability will in general be enhanced if new IT is consistent with existing technology or objects which the user are familiar with.

Users spend most of their time on a few applications, and most users are familiar with the basic functionality and interface of browsers and text processors. Hence, when building a new application, locating operations in menus, ribbons and screens in the same way as in browsers and text processors will ease navigation.

Users can get quite upset, by small changes in user interface when introducing new software versions. Microsoft made a larger change in the organisations of operations in their Office package in 2007, switching from a menu to a ribbon structure. Despite some loud protests, users learnt the new interface without too much trouble (Dostál, 2010).

Using known icons on buttons can ease interpretation and hence bring the user up to a functional understanding quicker than if introducing novel signs and symbols. Icons can also be brought in from other gadgets, for example, the symbols for play, stop, fast forward and backward from audio tape players have penetrated digital players and eased learning of these.

Designers may also exploit consistency with the domain of the system. The hotel information system could use iconic symbols for guests, rooms, travel agents, cleaners, etc. While it is possible to draw intuitive icons for physical objects, icons for more abstract phenomena like an event or a reservation may require explanations. A calendar seems to be a frequently used icon for reservations, see Figure 70 for a small selection of reservation icons without text.



Figure 70. Icons for Reservations.

New applications will obviously introduce novel functionality, which needs a name. The fact that people use many terms for the same concept (Furnas et al., 1987) creates trouble with deciding the name, and the best we can do is asking a number of people and selecting the most frequently used term.

Terminology is also an issue when naming data fields. Established terms from existing systems create less need for learning, and when redesigning business information systems, keeping the terminology is essential, regardless of whether a new system is replacing paper forms or digital databases.

Structures of data entry forms and reports can also be used as templates for new interfaces, reducing the learning of new patterns of data and sequences of operation.

6.3. Inline help

When simple things need instruction, it is a certain sign of poor design (Norman, 1988).

When users do not learn skills easily, improving the functionality and user interface is normally a better option than providing additional functionality for scaffolds for learning. This certainly applies to simple apps, social media, web shops and gadgets, as the quotation from a usability expert says. However, many systems have complex data or a multitude of functions, and everything cannot be depicted in one screen. For example, an amateur video camera may have tens of options, and the cinematographer needs a functional understanding of these before shooting. And a business system with a database as complex as the medical record system in Figure 34 would need a structural understanding which is difficult to obtain when only accessing a small part of it at a time. While the video amateur cannot split her attention by looking for help with camera settings while shooting, users in less intensive activities will have the opportunity to do so.

We saw in Section 5.4 that looking up inline help and searching the web was the fourth most common choice of users when trying to solve a problem (Novick et al., 2007). Here, we will present ways of providing inline help.

Tooltips

My first personal experience with accessing help in an interactive program happened in the days of alphanumeric displays where one application filled the whole screen. I used an editor where help would appear by pushing `Ctrl/H`. One day when looking for some options, I pushed the key combination and quite rightly, help appeared. The annoying thing was that the help filled the whole screen, such that I could not see the file I was working on. What was even worse was that the help screen lacked instructions on how to get back to the file. With nobody around to ask, I had to kill the editor process and lost the work since last saving. It is probably unnecessary to say that I never accessed the help again.

Help windows covering the programs was one shortcoming of early attempts at inline help, and another was the excessive length of explanations presented (Carroll, 1990).

Balloon help is a small box with explanation which pops up when holding the cursor over an interface item, being a button, menu item, data field or other specific places in a window. It was introduced in Apple computers in 1991, and the help appeared immediately when the cursor was located over the item. Most users found it annoying, since it cluttered the screen.

Later, Microsoft created a version with a time delay of about one second between when the cursor is placed over an item and when the scaffold pops up. This delay gives the user the opportunity to push a button before the help appears, and it prevents balloons from popping up all over when moving the cursor. This version was called tooltip or screentip and is now favoured.

Tooltips avoid distraction. When busy working with the spreadsheet, users prefer keeping attention to their tasks, not having to divert into a separate help system. The other way distraction is avoided is through providing help about what users are attending to. No search or lookup is needed.

The example in Figure 71 shows a tooltip from MS Excel. In response to users' negative reactions to long explanations, it consists of only 13 words. The first sentence gives a short functional model of this button. The second sentence explains the purpose of line charts, "display trends over time," thereby also showing which business task this function fits into. This is a quality which has been emphasized in particular in previous research on learnability (Carroll, 1990, Grossman et al., 2009)

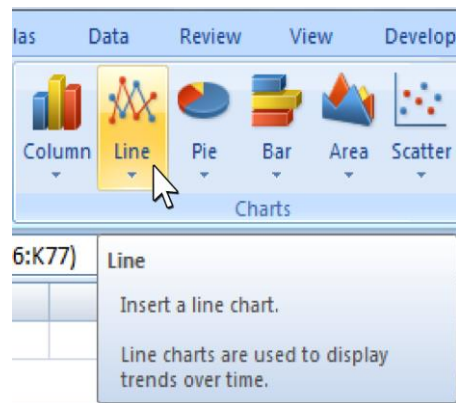


Figure 71. Tooltip help from Microsoft Excel.

The example demonstrates that even short help messages can show the user both a functional model and the business fit. A relevant question is; can tooltips convey instructions, directions or structural models also?

Instructions include a list of steps to be undertaken, often including several buttons to be pushed and choices to be made. Since tooltips are attached to individual buttons and disappear once the cursor is removed or the button is pushed, they do not provide a list of steps, which keeps stable on the screen for several buttons to be pushed.

Directions tell where to find an interface item. Since tooltips tell about the specific item it is attached to, it does not provide guidance for navigation. However, users might browse tabs and buttons in search of a specific function, without knowing its name in the particular application. Tooltips may help present alternative terms for the name on the button or data field, thereby possibly solving the terminology problem. For example, a user looking for the "time trend function" might find it by reading the tooltip in Figure 71.

Structural models display relations between several elements, something which requires space and possible graphics. Again, the tie between the tooltip and one particular interface item leaves little opportunity for relating the item to the rest of the system.

In an experiment where tooltips were attached to data fields, users looking up tooltips made less mistakes, and they said that they learnt correct data entry from the tooltips (Isaksen et al.,

2017). Interestingly, users also opened tooltips after data entry to check whether they had done it correctly. In this case, the tooltips functioned as reinforcement for learning.

Wizards

A wizard is a sequence of smaller windows forcing users through a predefined sequence of steps. In each step, the user may have to choose options and can go back or forward. Figure 72 shows a window from a wizard for creating a line chart.

Wizards are semi-automatic instructions preventing users from doing operations in a wrong sequence. They are the number one choice amongst software producers for installing software. Wizards provide users with low IT self-efficacy with the security of reaching an acceptable result by clicking `Next` repeatedly. At the same time, advanced users can pick and choose some options along the way.

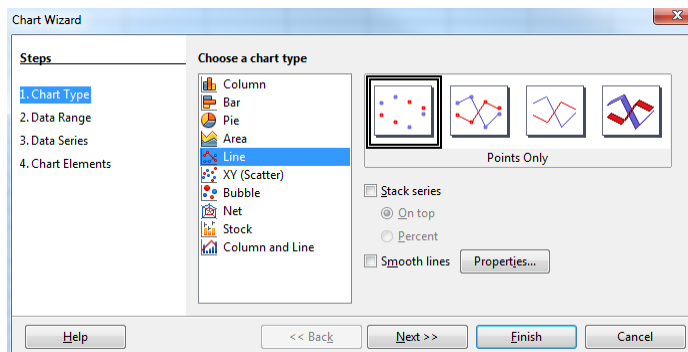


Figure 72. One window from a wizard from OpenOffice Calc.

Successful completion might boost users' self-efficacy. The individual windows have some space for explanations, which can help understanding. However, users might run through a wizard without learning anything.

Users are more likely to learn through repeated runs through a wizard, in the same way as people learn from repeatedly following instructions. Creating graphs with a spread-sheet, as illustrated in Figure 72, creates an opportunity for repeated use.

Context-sensitive document help

Some applications have help buttons that display a help window related to the place where the button is located. The help window, which is smaller than the application window, pops up at a location where it hides as little as possible. As an example, Figure 73 shows a help window that appears in a data entry screen of a business information system.

Data is registered for an organisation unit, a period, and a set of data elements (data set) at a time. A data set often corresponds to a paper-based data collection tool.

Figure 73. A context-sensitive help window from DHIS2.

This explanation provides the part of the structural information model, which is relevant for the data entry screen.

Section 4.4 on structural models provides some ways of visualising hidden structures. Showing the relations between master slides and slides in the presentation might for instance be an option similar to the visualisation of dependants in spreadsheets (Figure 30).

Business information systems like the hotel and the medical record systems in Section 4.6 may have used interfaces where hundreds of operations are organised in menus in tabs in a ribbon. The data visible in a screen may be one record or a table shown in isolation from related data, while the data structure may be at least as complicated as shown in Figure 74.

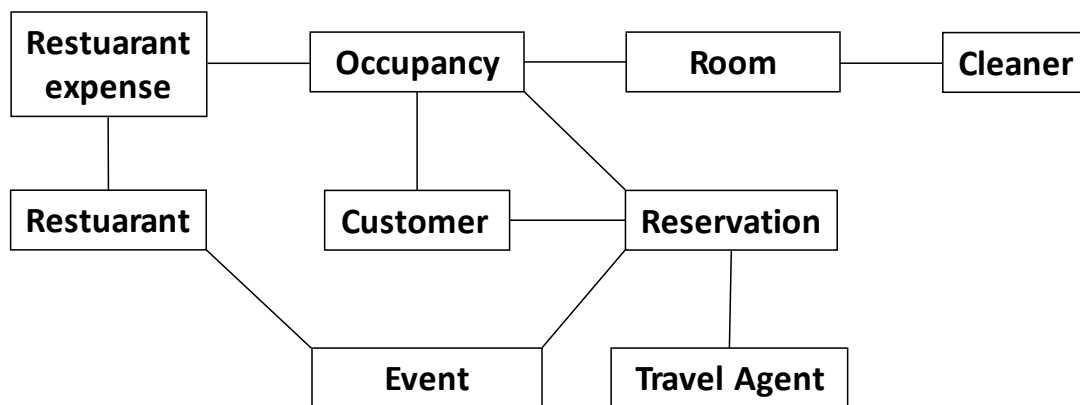


Figure 74. Data model of a larger part of the hotel information system.

Such a model might also be displayed by a context-sensitive help button.

Having such a small model visible in a corner of the screen with a colour indicating which table is active at the moment, may provide the user with a continuous map both reinforcing a structural understanding and easing navigation in the data.

Context-sensitive video help

As explained in Section 2.3, videos are good substitutes for a live trainer and can provide more effective instructions for novices than can documents. Videos triggered from a button in the interface can therefore provide valuable help, for example when opening a screen in a business information system and being told how to operate the functions therein.

Context-free help

A help window with a complete manual for a system is often what pops up when selecting a general help function in a program. The user is then to search or browse to find the topic of interest. Figure 75 is an example that can be searched and browsed.

When trying to navigate to the right spot, the user does not know the location of the relevant item in the application and can therefore neither find context-sensitive help nor tooltips. Context-free help can therefore be a useful alternative. However, these help systems are often

written by the vendor and have therefore the terminology problem (Furnas et al., 1987). If the user cannot locate the item on the screen due to not knowing its name, it may be equally difficult in the context-free documentation. Mixing user written questions and discussions into the help system may alleviate some of the terminology problem.

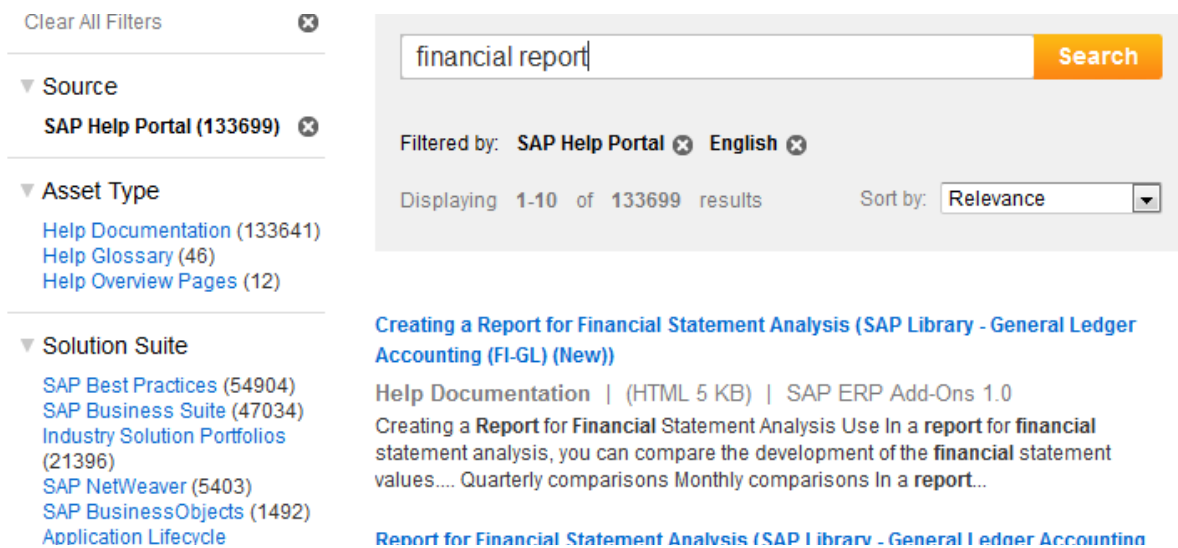


Figure 75. Context-free help system from SAP.

System-initiated help

As a response to user struggling with IT, vendors and researchers have developed automatic help based on logging user actions. The most wide spread was Microsoft's Clippy, a pop-up window which appeared in the lower right corner of the screen, see Figure 76.

This "office assistant" was heavily criticized by users for providing irrelevant and too trivial help (Olsen and Malizia, 2011) and being intrusive,

much like an insensitive colleague who pops in to one's office far too often (Waytz et al., 2010)

and Microsoft dropped it after a few years.

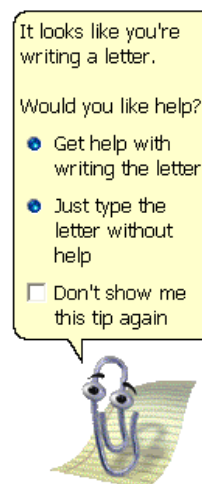


Figure 76. Clippy, a system initiated help from Microsoft.

Research has continued, and it has been found that system-initiated instructions were effective for novice users, while expert users benefitted from functional models (Babin et al., 2009). To be effective, system initiated help thus needs to target the users' level of competence, and the levels of skill, understanding and problem solving may be a basis for characterizing competence levels for such systems.

Summary

The various forms of inline help seem to be fitted to different learning processes, as illustrated in Figure 77.

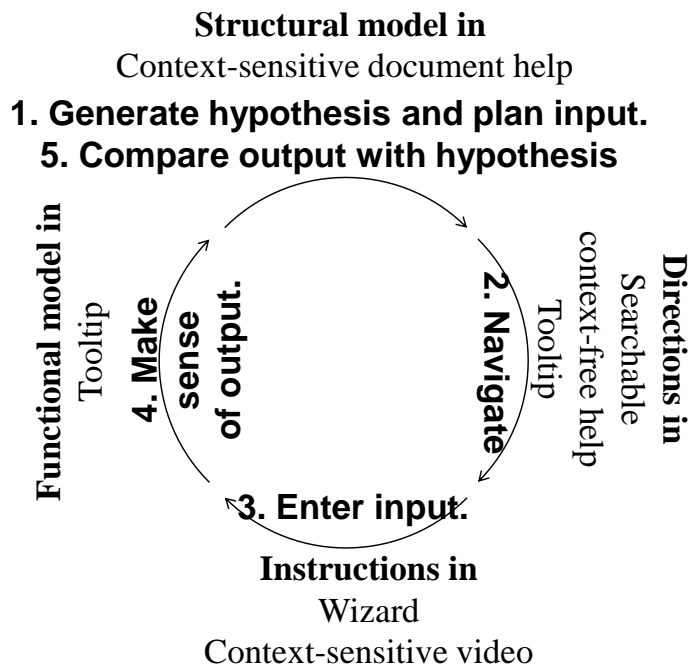


Figure 77. The roles of inline help for supporting steps in the research cycle.

6.4. Evaluating learnability

During design, the learnability of user interfaces and help systems can be evaluated in several ways. Three ways of increasing validity and costs are heuristic evaluation, question-suggestion and measuring learning.

Heuristic evaluation

Heuristic evaluation is carried out by 2-3 experts on IT usability. They inspect every detail of the application and compare it to known guidelines and principles. The sections 6.1 to 6.3 provide several guidelines to which interface and help system can be compared. Concerning inline help, three issues to consider in particular are

- Do tooltips provide both functional model and business fit?
- Are structural models included in any help?
- Does the help include terminology from common users in addition to the vendor's terms?

Heuristic evaluators should note down each time a guideline is broken and note this as a possible learning issue. Even if the design does not match a guideline, this may not pose any learning problem for users. For finding out, real users need to test the system.

Despite this insufficiency, heuristic evaluation is suggested as a first test by human computer interaction specialists, due to that it is cheap and can reveal issues to be solved before more costly user evaluations are carried out.

Question-suggestion

Thinking aloud has become an established method for evaluating ease of use. A think aloud session consist of a user given some tasks to be carried out on an IT component and asked to say what she thinks when trying to carry out the task. Thinking aloud unveils many usability issues.

A related procedure has been demonstrated to yield more results when assessing the learnability of software. In the Question-Suggestion procedure, an expert user with many years of varied experience with the software sits beside the learner. The learner can ask specific questions to the expert, who answers the questions, and this provides for a more natural dialogue than the artificial thinking aloud (Kato, 1986). The expert can also, when judged appropriate suggest alternative and better ways of working to the learner. In a comparison with 10 learners on an architectural design system, the Question-Suggestion procedure unveiled 2-3 times as many learnability issues as thinking aloud (Grossman et al., 2009).

Both thinking aloud and Question-Suggestion requires an experimenter who takes notes on the learnability issues. Also, through video recording of the screen and audio recording of the learner, one can gain more insight into the learning processes.

Bringing in a number of users and experts add to the cost compared to heuristic evaluation. Although no exact estimate exists, rough measures has shown that 5-10 users find the large majority of usability problems in thinking aloud (Nielsen, 1994). This number cannot be automatically transferred to learnability, but the similarity in concepts and procedures suggest that the number is closer to 10 than to 100 for finding the majority of learnability issues.

One hour of video recording may require 10 hours for analysis, also adding to the effort of Question-suggestion evaluation. For systems where learnability is critical, like a web shop, a variety of evaluation methods is needed.

Measuring learning

A simple way of comparing learnability of two systems or two versions of the same system is measuring how long time it takes to learn specific tasks. A group of around 10 people are given a set of tasks they have never done before and told to fulfil these tasks with the IT. The time they spend is measured and errors counted.

Such measurements may show which system is more learnable, and thereby tell us whether a new software version has improved learnability. Learning measurements can also be used when deciding on purchasing one or the other software package.

These pure measurements are less useful for pointing to specific design problems or locate concepts in the software which are difficult to learn and therefore need inline help or training.

Chapter 7. Training modules

We have so far considered the sequence of learning from skill through understanding to problem solving competence, and scaffolds for improving learning have been discussed.

The learning objectives of this chapter are to be able to combine scaffolds to a training sequence and to design a sequence for training related topics within IT use.

7.1. Training modules for skills and understanding

A *training module* is a small sequence of activities with a learning objective. For instance, if the learning objective is the ability to use functions in spreadsheets, the training module could consist of a list of exercises with increasingly complicated functions. A training module could be taught by a trainer in a class, be available as an online tutorial for self-study, or through any combination of media and people.

Studies of training for understanding have shown that presentation of functionality and structure prior to practice improves learning (Mayer, 1989), and that reflecting upon the same functionality/structure after practice also helps understanding. This sequence is also in accordance with a general sequence of teaching for skills and understanding (Gagné and Briggs, 1974). Concerning IT user training, the same sequence of teaching helped understanding for students in high income countries (Hadjerrouit, 2008, Bhavnani et al., 2008) and in a disadvantaged community in a middle income country where few learners were familiar with computers (Marsh, 2007).

Understanding business fit has shown similar advantages. In an experiment, two out of 16 hours of skill training was substituted with a session aimed at understanding the organisational level for half of the participants (Coulson et al., 2003). Those who received the training for understanding business fit developed better mental models than those who only had skills training.

In addition to motivating the learners, presenting functionality, structure and usefulness prior to practice helps trainees combining the new material with what they already know. For instance, if paragraphs in text processors are to be taught, an introduction showing the similarities and differences between the concept of paragraph in natural language and that of text processors might bring the learner's understanding in the right direction and avoid interference.

We have previously stated that people need to practice and develop skills before they can understand. The learning outcome after the Introduction part is therefore characterised as “vague understanding,” since most learners at that stage will only have a hunch about what the models mean. The research on training referred above has nevertheless found that the final learning outcome of a module becomes better with this initial motivation through usefulness and the vague understanding. A training module for learning skills and understanding should

have the parts described in Figure 78. Usefulness should be repeated after the learners have practiced the IT to motivate the learners for using the software after training.

Training activities and scaffolds	Learning outcomes
<ol style="list-style-type: none"> 1. Introductory presentation of <ol style="list-style-type: none"> a. Usefulness b. Business, Functional or Structural model 	Vague understanding
<ol style="list-style-type: none"> 2. Practicals <ol style="list-style-type: none"> a. Provide Minimal Manuals and Exercises b. Help learners who have reached an impasse 	Skills
<ol style="list-style-type: none"> 3. Summary <ol style="list-style-type: none"> a. Review Usefulness (organisational benefits) b. Review Functionality and Structure 	Understanding

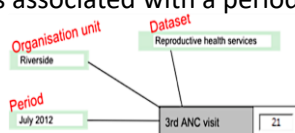
Figure 78. A teaching module aiming at skills and understanding.

A module that follows this scheme is presented in Figure 79.

Motivation

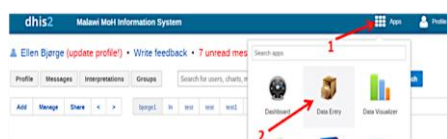
When you enter data, the system can be used to analyse it, make reports and compare of data across time and place.

Data set: The data is associated with a period (when) and an Organisation unit (where).



Practical exercise - Data Entry

- 1: Click on the **Apps** button up at the right on the screen.
- 2: Click the **Data Entry** button.
- 3: Click **+** once to display organisation units.



Questions

- Which three dimensions does a number you have entered belong to?
Why should you enter data in the system?

Figure 79. A training module for skills and understanding. The summary consists of two questions to the learners and subsequent discussion.

During introduction and summary, instructions for particular software installations are avoided in all written materials and presentations for three reasons. First, the introduction and summary should focus on understanding. Second, the strict separation allows learners using different versions of software to be taught, as long as there are instruction sheets for all versions. Third, a modularised design of training material is achieved. When a new software version appears, the material for introduction and summary can be kept untouched.

One module should cover 1-3 new functions or structures, such that the trainees understand these before progressing to the next module. In order to develop skills, the time for practicals should cover the majority of the duration of the module. For instance, a 30 minutes session might be planned for 5 minutes introduction, 20 minutes practical assignments and 5 minutes summary. Overburdening a module with too much material is particularly detrimental for the learners' understanding (Rourke and Kanuka, 2009).

When working on the practicals, many learners get stuck, observe an error or are uncertain about their actions. These learners have reached an *impasse*, which is a good opportunity for learning. Trainers should therefore plan for impasses to take place by providing increasingly difficult exercises. The first assignment in the practicals could be imitating the instructions, while a second could introduce a minimal change, for example asking the trainees to operate on some other data. Larger deviations from the instructions are considered under training for problem solving in the next section.

During the practicals, the trainer should walk around in the classroom and help out. Helping out during impasses has good effect on learning, while trying to help when the learners are coping on their own is wasted effort (VanLehn et al., 2003). The trainer's explanation should be questions or short explanations. These could be parts of functional or structural models triggering the learners to compare with what they do (Roscoe and Chi, 2007). Lengthy explanations beyond what the learners need to continue from the impasse are to be avoided.

During the practical, obviously the learners are busy on their computers. This is necessary for developing the initial skill. Since talking while doing contributes to understanding, it might be better if the learners were discussing while running the computer. This can be achieved through having two learners at each computer. One is operating the keyboard and mouse, while the other is making comments, asking questions, checking the documentation etc. Then the pair swaps roles, such that both do the hands-on exercise. A laboratory study showed that learners operating in pairs outperformed individuals on understanding. The pairs could explain more about how the software operated and they performed better in exercises which introduced some novel elements (Lim et al., 1997).

Operation in pairs is known from computer science education as 'pair programming.' Several studies have been carried out, and pair programming is found more effective for bringing more students through exams than individual programming (McDowell et al., 2006). The lesson from learning of programming addresses learning objectives of understanding and problem solving.

Motor skills were not considered, neither in the user learning, nor the computer science cases. It would be reasonable to believe that learners who still struggle with the mouse and keyboard, should be practicing this as much as possible, while pair learning fits better for those above this level.

When school students or participants in a formal in-service training with a test to be passed know the type of exams or tests, they focus on learning the competence tested. If assessments only address skills, chances of reaching understanding are smaller than when including assessment of learners' understanding as well in the exam (Ramsden, 2003).

7.2. Training modules for improving problem solving competence

Since problem solving competence builds on understanding, trainees should be prepared for a module on problem solving after completing one on skills and understanding.

In a study of adult computer novices, half were trained through a series of exercises for the whole duration of the experiment. The other half was given a few exercises followed by a prompt to explore the operations (Davis and Bostrom, 1993). There was no difference in the learning outcome. This indicates that encouragement to explore is not sufficient for developing problem solving competences.

Encouraging users to make errors during training by emphasizing that errors constitute good opportunities for learning has shown promising effects both for performance and learning oriented users (Keith and Frese, 2008). By coupling error encouragement with emphasizing that the learning objective is improved competence and not achieving a high performance, also performance oriented users above 40 years dared to explore and learnt more than those only receiving instructions (Chillarege et al., 2003).

Problem solving can be learnt more effectively when methods for solving problem are presented explicitly (Hattie, 2009). Inserting a module which teaches a problem solving strategy between modules aiming at skills and understanding is more effective than grouping all problem solving approaches into a specific course (Abrami et al., 2008).

Training which addresses the problem solving methods presented in Chapter 5 requires a similar module as teaching any concept, idea, theory, functionality structure or other abstract thought. This module can be illustrated as in Figure 80, where the vague understanding after the introductory presentation corresponds to that of the module for understanding..

Training activities and scaffolds	Learning outcomes
<ol style="list-style-type: none"> 1. Introductory presentation of <ol style="list-style-type: none"> a. Needs for problem solving b. Problem solving strategy 2. Practicals <ol style="list-style-type: none"> a. Provide problem exercise b. Help learners who have reached an impasse 3. Summary <ol style="list-style-type: none"> c. Discuss the problem solving strategy compared to the practical 	<p>Vague understanding of problem solving strategy</p> <p>Skills in applying the problem solving strategy</p> <p>Understanding of the problem solving strategy</p>

Figure 80. The sequence of teaching a module for learning problem solving.

Corresponding to the training for understanding, pair working during the practicals may help the learners gain more understanding of the problem solving method as well as of the IT (Lim et al., 1997). Having the trainer helping out during impasses in the same manner as mentioned above may help. For improving the learners' problem solving competence, the trainer's comments should focus on the problem solving method more than on the IT.

As mentioned in Section 5.4, self-efficacy can be improved by watching a peer, who is considered at the same level of IT competence, solving a problem. This is an additional argument for pair working during IT use learning.

Training generation and testing of hypotheses could be done by providing the learners with an incomplete functional or structural model of the topic to be learnt (Brown and Newman, 1985). For example, when training layers in a picture editor, the presentation may say that layers are stacked upon each other, and that the image in one layer hides those in layers below. After learners have worked with layers in this sense for a period, a file can be presented where the lower layers are partly visible through layers above. The exercise could be to find out how this came about. This calls for questioning previous understanding and hypothesizing that layers may be transparent in some way.

Learning problem solving is especially important for superusers; since they are supposed to help others solve their problems. The sequence of training for understanding prior to training for problem solving as illustrated in Figure 81 is therefore recommended for superuser training in particular.

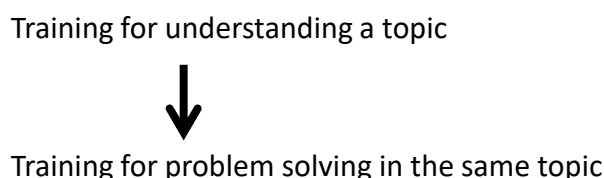


Figure 81. The sequence of training for understanding prior to training a problem solving strategy..

If the learners have been taught the problem solving strategies earlier, there is no need for specific modules presenting the approaches again. The non-trivial exercises which require problem solving can then be added to the exercises in a skills and understanding module.

7.3. *Teaching a module*

Training by a trainer is more efficient than self-exploration (Simon and Werner, 1996). Also when exploring IT systems, trainer intervention through asking questions relevant to the domain to be learnt triggered a similar number of learning events as the students generated on their own (Price and Falcão, 2011).

The main advantage of having a trainer leading the training modules compared to self-studying a tutorial is the interaction between trainees and trainer. A trainer can answer questions, help out when learners are stuck, adjust the explanations to fit the trainees' understanding, discuss usefulness with them and discuss pros and cons of different problem solving methods. These advantages should be exploited when planning and carrying out teaching of modules.

When learners work on their own, the trainer can move around freely in the classroom to help out when needed. Instructing by means of a demo at the projector means that the trainer has to rush between the screen and the learners to help out those who are stuck. Experience shows that working with instruction sheets or videos relieves the trainer from most of the requests, since the learners have material to consult (Herskin, 2006). The trainer is gaining time to respond to the questions from the learners.

To ease learning, users should have both the instructions and the software to be learnt visible on the screen at the same time, not hiding the windows behind each other. The trainer should therefore tell the learners to reserve one part of the screen for the instructions. Reminding the trainees to stop and start the video according to their own speed of working is also useful for easing learning.

When observing learners practicing, the trainer also assesses their skill level. Sufficient skills do not mean that the learners have understood usefulness, functionality and structure, however. The summary should therefore continue the assessment, by asking the students to express their understanding. Kylie has taught styles and has observed that all

participants were able to change and apply styles in their documents. Nevertheless, their understanding is poor. Dmitri does not seem to be aware of that styles apply to units called

Assessing understanding during summary—Trainer Kylie and two learners Dmitri and Julia.

Kylie: *Can you tell us what a style is?*

Dmitri: *It is formatting.*

Kylie: *Yes, and what does it format?*

Dmitri: *The document.*

Kylie: *Does one style format the whole document?*

Julia: *No, each paragraph has a style.*

Kylie: *Yes. Can two paragraphs have the same style?*

Julia: *I am not sure, ... but I think so.*

Kylie: *OK, look at this document. Now, I change ...*

paragraphs. Even if Julia knows this, her last statement indicates a poor understanding of the usefulness of styles, namely that one can keep and change formatting in a consistent way throughout a document by applying the same style to paragraphs that should have the same format. Kylie knows that confronting learners' misconceptions is crucial for making them interpret the computer output again, reflect more and develop a more adequate understanding (Ramsden, 2003). Based on her assessment of the trainees' understanding, Kylie therefore decides repeating the whole explanation by means of demonstrating the function and usefulness of styles. Thereafter, she will check the trainees' understanding once more.

During classroom training, another way of improving trainees' self-efficacy is available. The trainer could assign average performing trainees to demonstrate their skill on the projector, such that the other trainees see that their peers have learnt it.

Developing functional and structural models plus instruction sheets consumes time, which may be one of the reasons why trainers often skip this preparation and rather present interaction with a projector for the users to imitate. Users who were asked to rank the quality of different aspects of IT support gave the lowest score to documentation in training (Shaw et al., 2002). Knowing that reviewing material from training is twice as successful as searching for help other places (Novick et al., 2009) and that users look it up after training (Kaasbøll, 2014), putting more effort into training material could make users become better problem solvers and save the IT supporters from many encounters.

In summary, the trainer could carry out a module for skill and understanding like this:

1. Introduction
 - a. Present learning objective and usefulness and discuss these with the trainees.
 - b. Present functional or structural models for the concepts of the session.
2. Practicals
 - a. Hand out directions and instructions and exercises.
 - b. Observe the trainees' performance.
 - c. Help them if needed, not by doing the exercises, but by explaining functionality and structure.
3. Summary
 - a. Discuss usefulness in the business with the trainees.
 - b. Ask the trainees to explain the new functionality/structure to assess their understanding.
 - c. If the trainees' understanding is poor or they have misunderstood, the trainer needs to confront their misconception, repeat the explanations and redo the assessment.

Problem solving modules can be taught in much the same way as the training for understanding. Usefulness will in this case refer to the applicability of the problem solving strategy in other settings.

When the exercise is difficult, the learners might have to go through many cycles of experimentation. For efficient learning, the trainer needs to provide lots of support during such exercises (Hmelo-Silver et al., 2007). Since the aim is learning problem solving, the

trainer should explain how the trainees should work with the problem solving strategy to find out on their own how to solve the IT use problem. A module for learning problem solving could be taught like this

1. Introduction
 - a. Present the usefulness of the problem solving strategy.
 - b. Present the problem solving strategy.
 - c. Encourage trainees to try on their own and learn from mistakes.
2. Practicals
 - a. Hand out description of problem solving strategy and non-trivial exercises.
 - b. Observe the trainees' performance.
 - c. If needed, help them in their problem solving by addressing their problem solving approach, and not by telling them the solution to the exercise.
3. Summary
 - a. Discuss with the trainees their approach to solving the problem and the applicability of this approach for other problems.

7.4. *Online tutorial for a module*

Training by a trainer is normally not available when wanted, so users need to learn without a trainer in most cases. While the advantages of a trainer was spelled out in the previous section, there is also one main advantage of following an accessible tutorial; you can do it whenever suited and at your own speed. The challenges with designing tutorials for training modules are to compensate for the lack of a trainer.

The introduction lends itself reasonable easy to be recorded or written. As previously stated, users want to act, not read. In an online tutorial the introduction and summary should therefore include only the necessary contents.

Previously the use of video for presenting models and instructions has been described, and usefulness can also be presented in a video. If the usefulness of the IT in the business is presented by a known person, someone preferably working in the same organisation, the message may become more persuasive than if presented by an anonymous trainer.

The practical is mainly the learner's activity, but the trainer's observation and guidance will be missing in an online tutorial. Assuming the first practical exercise is following instructions, the software could be programmed to monitor the user's interaction and alert if the user pushes the wrong buttons. However, this is a very expensive means with an unknown outcome and would probably only be implemented in flight simulators or other high tech learning environment for safety critical tasks.

Following the general advice of two learners working together may be a simpler and more effective way of compensating for a trainer who observes and helps out during practicals. Asking a colleague or a friend for help may also do the trick.

The summary has the highest demand for interactivity. A few written sentences or a talking head in a video repeating the usefulness and the novel functionality or data structure is a way

of training that leaves the learner completely passive, hence the learner might not reflect at all. Also, such a presentation provides no mechanism which checks whether the learner has reached the understanding of the IT or of the problem solving strategy. The learner may therefore proceed to the next module without the prerequisite competence.

Multiple-choice questions is a simple, interactive way of testing competence. Questions addressing understanding as well as business fit can be made. Software exists that can provide feedback on right and wrong answers and guide the learner to the next module or repeating the current.

The tutorial module in Figure 82 is constructed according to the module for skill and understanding:

1. Introduction
 - a. Motivation. Users know that they need to print and export.
 - b. Structure and functions. Learners of this software are familiar with spreadsheets, where the input is done in the same sheets that are printed for output. A frequent misunderstanding amongst these learners has therefore been that the input form should also be printed as output.
2. Practical
 - a. Instructions to follow
3. Summary, multiple choice questions on
 - a. functional understanding
 - b. structural understanding
 - c. skill

(from (Bjørge et al., 2015))

The first multiple-choice question forces the learner to reflect on the difference between how this database system works and a spreadsheet, and the second addresses the contents of the report. In the IT version, if the learner does not pick a correct answer, an explanation of why it is wrong pops up. The learner gets three attempts at answering correctly before being allowed to proceed to the next module. In this way some interactivity is achieved as well as a control of competence prior to continuing the training.

5. Data Set report

Motivation

Data set reports enable print and export to Excel of Data sets. Both facility data and data aggregated at district and higher levels are available.

Structure and functions

In Excel, data input and output takes place in the same window. In DHIS2, *input* is done in the Data Entry form, while *output* is done through Reports; standard reports, data set reports, reporting rate summary, organisation unit report, data approval, report table, pivot tables, graphs and maps.

Another difference from Excel is that data entered in DHIS2 needs to be processed by the server before it can be retrieved in reports. This processing happens at night. The data you entered will therefore only be available in reports the day after data entry.

Practical exercise - Data set reports

1. Click on the Apps button.
2. Click the Reports button.



(... instructions continue)

Assignment

1. What is the difference between data input and output in DHIS2?
 - a. Output is typing the numbers in the Data entry form.
 - b. Output is retrieved from reports, while input is through Data entry.
 - c. Input is done through Data entry, and output consists of exporting Data entry to Excel
2. What is a data set report?
 - a. The same as the data entry form.
 - b. Facility data and aggregated data at district and higher levels.
 - c. Facility data and patient data at district and higher levels.
3. How do you choose report period?
 - a. Click on get report
 - b. Choose year and then frequency.
 - c. Choose frequency and then year.

Figure 82. A tutorial module with multiple choice assignments (1 and 2) which test understanding.

7.5. Sequence for teaching related topics of IT use

The sequence of teaching understanding before problem solving concerned the teaching of one topic within IT use. If learning two distinct applications, like a spread-sheet and a text processor, there is no obvious advantage of starting with one or the other. Normally, training includes several related topics, for instance, cell references, functions and graphs in a spread-sheet course. When topics are related, and possibly build on each other, the teaching sequence should be designed accordingly.

Concepts which build on each other

Assume that you are going to learn making Table of Contents in a text document. Generating the table presupposes that the items to be included have been styled with the appropriate heading styles, so you have to learn about styles before the table of contents. Teaching of

Styles is therefore scheduled prior to teaching Table of Contents, as illustrated in the upper row in Figure 83.

Scheduling can be done during planning of teaching. During the actual teaching of styles, the trainer also has to ensure that the learners have grasped the Style concept. Structural understanding means that the learner can use the concept while talking about other concepts. This implies that heading styles should be understood at the structural level before learning table of contents. The trainer Kylie in the textbox at p.108 is just checking that her trainees has a structural understanding of styles. She should only proceed to Table of Contents after the learners demonstrate that they are at this level.

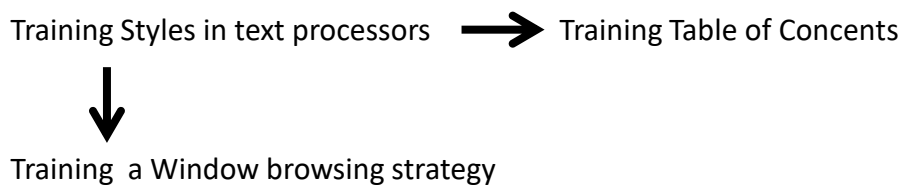


Figure 83. Top row: Teaching the concept Styles before Table of Contents since the latter builds on the former. Left column: Teaching a problem solving method after teaching Styles.

If Kylie wanted her trainees to find out more about styles on their own, she could also have started a session on problem solving methods aimed at styles. She could for instance select the interface browsing strategy (p.76) and make the learners carry it out for the main window for modifying a Style, see left column of Figure 83.

Concepts which are specialisations of other concepts

Table of Contents creates a list of references to the sections in the document. There are other ways of making references inside a document also, for instance footnotes.

Normally, people learn in a process from the specific to the abstract. Users who have learnt table of contents and footnotes would have seen two types of references and can understand the similarities and differences between these two types. This will enable learning the more abstract concepts of references or links, enabling a teaching sequence as illustrated in Figure 84, left side.

After having learnt the more abstract concept of reference, learning more specific types of references like index, cross reference and bookmarks will be easier. These teaching sequences are illustrated in the right part of Figure 84.

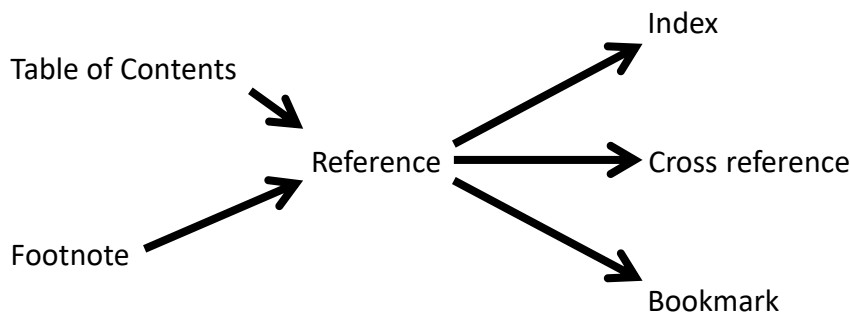


Figure 84. Sequence of teaching from two specific (left) to an abstract concepts (middle) and further to more concrete concepts (right).

Victor is a learner in these teaching modules. After training on Table of Contents and Footnotes, he explains these as shown in the textbox. He partly refers to how these are created, which points to the skill level. In addition, he mentions the inputs needed for table of contents, and he compares footnotes with those in books, which indicates a functional understanding. In summary, Victor is somewhat above the skills level, but not quite at the functional understanding. Hence, he is below where we would like him to be when proceeding to the Reference session.

Table of Contents and Footnotes 1—Victor:

We can make a table of contents by first styling the headlines that we want to include, and then do the Insert table of contents where we want the table.

Footnotes are like we see them in books. We make them by Insert footnote.

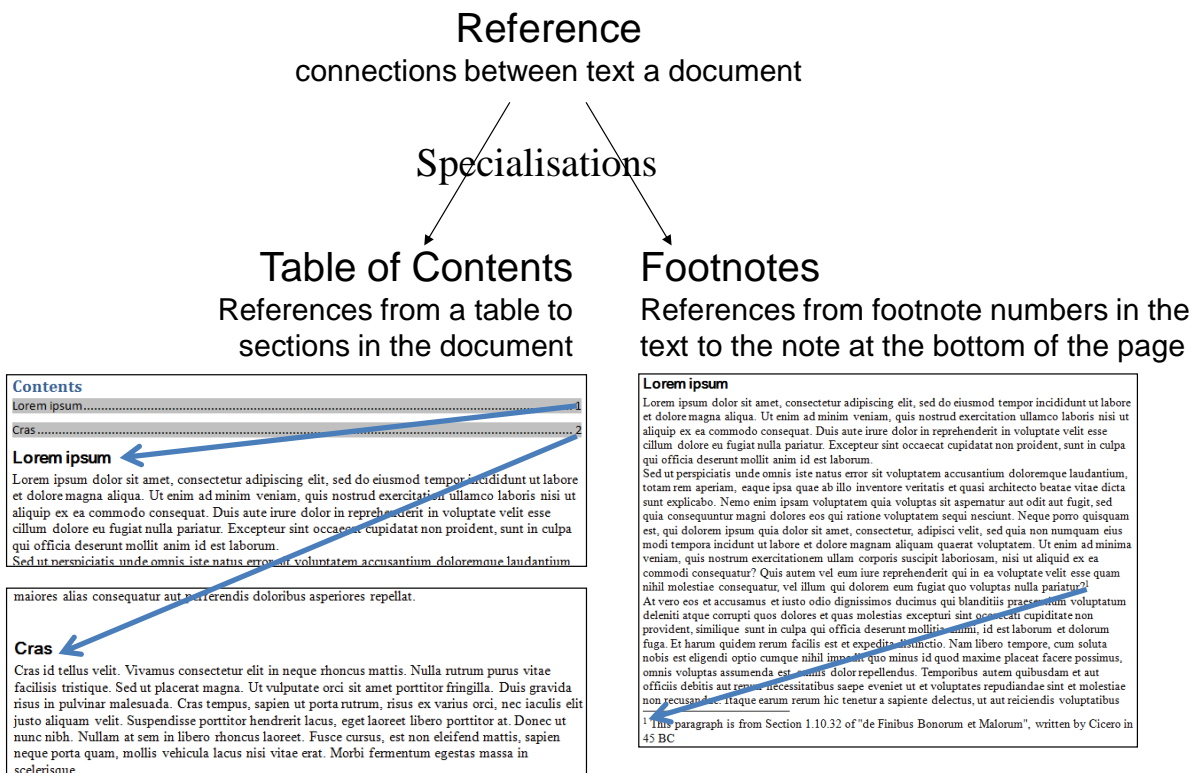


Figure 85. A structural models of relationships between Reference concepts.

However, the trainer proceeds nevertheless. She presents the more general idea of references, and how footnotes and table of contents are two different types of references, and she uses the structural model in

Table of Contents and Footnotes 2—Victor:
Footnotes and table of contents are references. That means that they point to where the text is in the document.

Figure 85 as an illustration in her presentation. Thereafter, Victor has to explain again. He no longer explains the concepts of footnotes and table of contents with mentioning how they are made. His last sentence points to a functional understanding. Also, he uses the two concepts when explaining references in general, which is a hallmark of IT structural understanding, but since he does not mention what a reference is, we cannot be sure that he has reached this level.

When designing a sequence of teaching for IT concepts, a directed graph of how they relate as shown in Figure 84 would be useful. However, there is no definite point where the more abstract concepts should be introduced, and there is no unique level of abstraction that is feasible at any point. Figure 86 provides a brief summary of IT concepts that fall in under the functional dependency category. Considering people in general learn from the specific to the abstract, the bottom level in the figure might be appropriate as starting points for user learning.

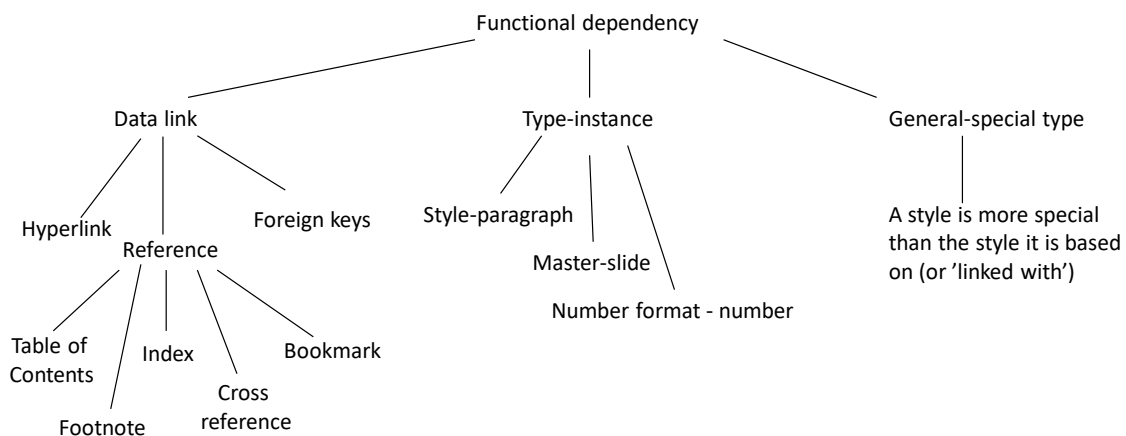


Figure 86. IT concepts from abstract at the top to more specific at the bottom.

7.6. Age levels of abstraction

While computer scientists may find Figure 85 intuitive, they may struggle with Figure 86. Correspondingly, users with less insight into computers and lower abilities for abstraction will also have trouble understanding both the model in Figure 85 and the idea it presents, regardless of the way of presentation.

Children become more able to deal with abstraction with age. Teaching at the right level of abstraction is the single most important consideration when trying to make children understand concepts, principles and general ideas (Hattie, 2009).

The thinking of preschool children is strongly influenced by their perception (Ormrod, 2012) and they conceive of objects as belonging to one category only. This might, for example, imply that the child regards the functionality of an Android device as separate from that of an iPad, and the idea of a file system is beyond their ability of abstracting. Structural understanding is not achievable unless the structures can be perceived directly.

Children up to age 11-12 are characterised as at a “concrete operational stage.” While still having trouble thinking about hypothetical or counterfactual situations, children at this age can sort objects according to various aspects such as colour and size and deal with subclasses (Ormrod, 2012). This implies that these children can understand that table of contents and footnotes have similarities and differences, like Figure 85 illustrates. The figure is not designed for this age group, however.

From the early teens, there is no marked difference between adolescents and adults. However, children as well as adults vary a lot in their abilities to handle abstract concepts and ideas (Ormrod, 2012). Some may therefore grasp many structural models at an early school age, while others will have poor structural understanding of IT throughout their lives.

7.7. Instructions, functional and structural models – slide design

Designing functional and structural models of information is covered in the area of information visualisation. The books by Edward Tufte constitute a comprehensive introduction to the area (Tufte, 1990, Tufte, 2011). Marti Hearst (2003) has made a tutorial on graphical elements and how people experience them, while Rosling (2006) provides a video of visualisation of numbers and statistics. This section presents some functional and structural models for a representation system, which is also useful in user training, namely slides.

Figure 87 presents the information structure of a file of slides generated with a presentation program. The model might contain the most frequently used data elements and data structure for a user. It shows that the file content is broken down into smaller and smaller parts in a hierarchy. In addition, we see that there are master slides which determine the layout and design of the slides and their contents, and that the master slides also can be parts of larger structures, called templates. The model is an example of how type-instance and data structures are mixed in a file.

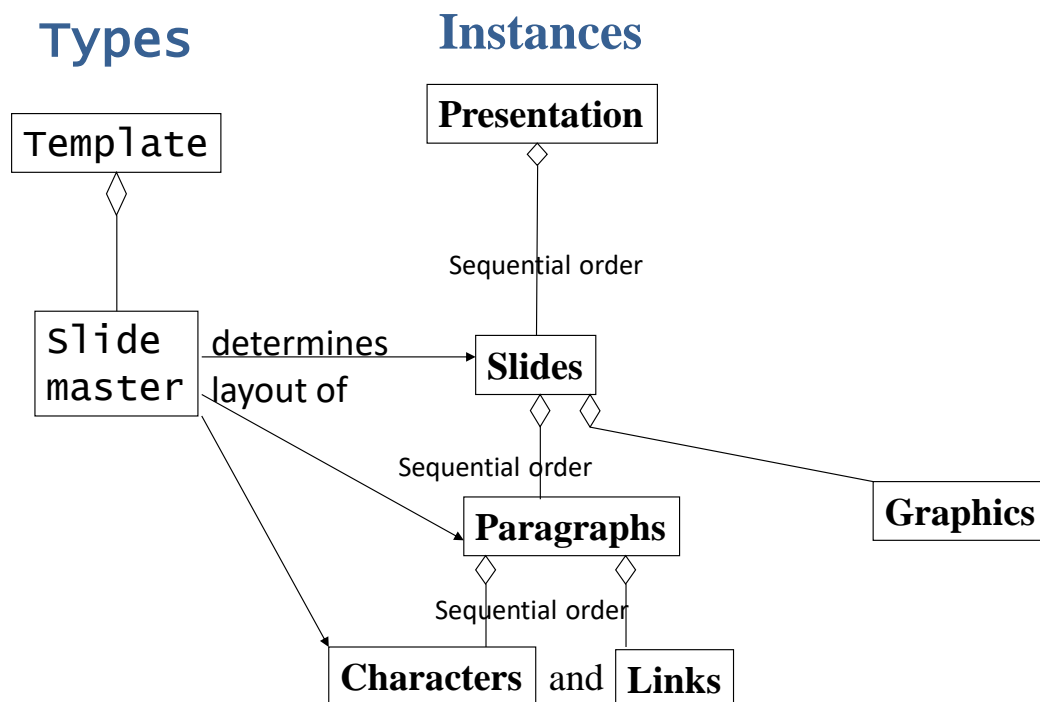


Figure 87. The structure of a file generated by means of a presentation program. The notation is derived from UML class models.

The model in Figure 87 is presented with a notation taken from computer science. It is therefore inappropriate as a means of communication with users, since they are not acquainted with the abstract notation utilized here, even though it is intended to capture the syntax concepts from a user point of view. Figure 88 shows a smaller part of the data structure of a presentation file with recognisable screen shots and an example. This structural model may help users understand relations between master slides and slides.

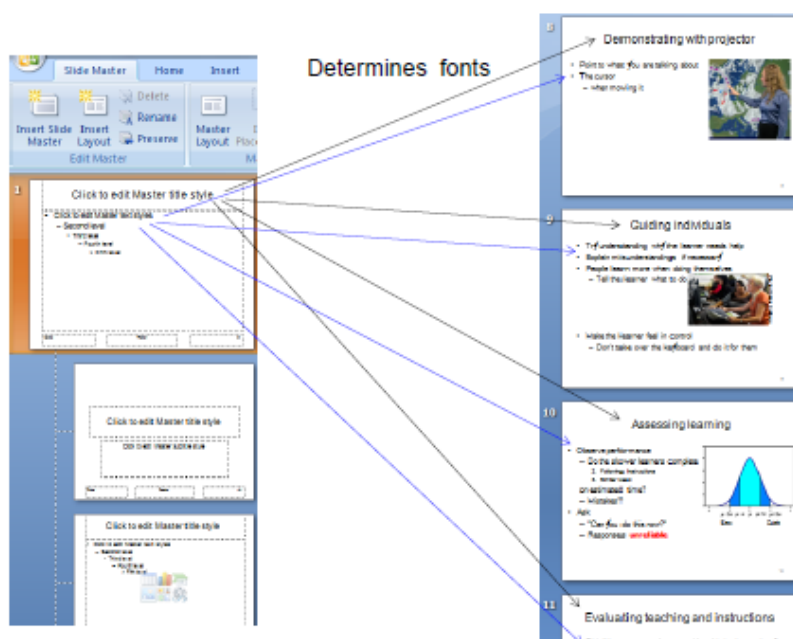


Figure 88. A structural model of master slides and slides.

While the structural models might help users understand how the data in a spreadsheet file is combined, slide design skills are also useful for creating useful slides.

Slide design depends on several ways of representing the world; it brings in the written language and all possible kinds of illustrations. Using already known representation systems makes slide design appealing, since people do not consider that designing slides require any additional competence. However, this is also the pitfall of slide design; people use Impress, Keynote, PowerPoint or Prezi as if it were a word processor with page organisation of the text and easy manipulation of the format and figures. The outputs of presentation programs are used for accompanying a presentation or for displaying a slide show on its own, and both of these differ from the purpose of a written text, which is meant to be read at the readers' speed. Also, each of the two purposes of a slide show has its own design rules, and we will in this section address the design of slides which accompany an oral presentation. The principles from at p. 49 on videos are also valid for an automatic slide presentation with recorded audio.

A main reason for using slides in a presentation is that it allows for several ways of presenting material simultaneously. Since some people learn better by hearing, others through reading and still others through seeing a figure, all three groups in the audience can be satisfied at the same time. Moreover, most people learn even better through a combination of media, so that presenting in oral, written text and figures at the same time is advantageous for all the audience. This brings us to the first instruction for slide design:

1. Combine text and illustrations.

When we are listening to a presenter at the same time as reading the text on the slide, we would easily lose out on one or the other. In order to minimize this fall out, we would normally write text, which reads very fast, and this could be seven words in one line. This line should present the essence of what the presenter is saying in a few sentences, which could constitute a paragraph if written. Copying full sentences from a textbook or a web page onto a slide, which some presenters are doing, is therefore a dysfunctional way of using slides. The slide in Figure 89 illustrates the result of copying full sentences into a slide. In order for the audience to grasp the message, the presenter has to read the full text aloud. The main message of this slide is rewritten in Figure 90. The text on the slide can be read in five seconds, and the design illustrates the process of producing the letters. After pointing to the four elements in this slide for explaining the essence of mail merge, the presenter can subsequently tell about, how the list of recipients is stored and other details from the full text.

Mail merge

- **Mail merge** is a software function describing the production of multiple (and potentially large numbers of) documents from a single template form and a structured data source. This helps to create personalized letters and pre-addressed envelopes or mailing labels for mass mailings from a word processing document which contains fixed text, which will be the same in each output document, and variables, which act as placeholders that are replaced by text from the data source. The data source is typically a spreadsheet or a database which has a field or column matching each variable in the template. When the mail merge is run, the word processing system creates an output document for each row in the database, using the fixed text exactly as it appears in the template, but substituting the data variables in the template with the values from the matching columns

Figure 89. An inappropriate slide design. Text from Babylon Online Dictionary.

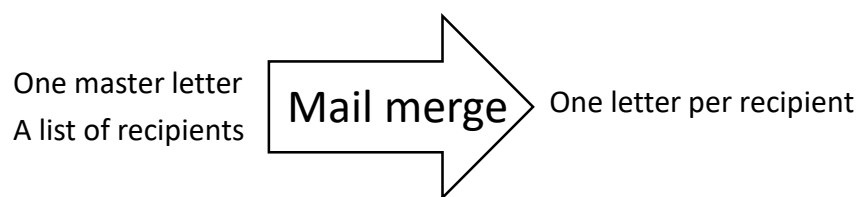


Figure 90. One point per line. A functional model.

In some cases, for instance when presenting a quote of a couple of sentences, we need to display the full text on the slide. To avoid fall out in such a case, the presenter should read the quote in full, so that the oral and visual impressions are synchronised. In general the instruction is:

2. Write each point on one line.

Simplicity is also an advantage concerning illustrations. These should display the essence of the point and avoid disturbing details. If the point of the illustration is to show the reality, a photo is appropriate, but unnecessary surroundings should be cut to avoid distracting details. If the point is of a more abstract character, a drawing is better suited for communicating the essentials and avoiding the disturbances. In summary:

3. Keep illustrations as simple as possible

Text and figures displayed on a screen may look large for the presenter, but the audience in the back of the class-room may have trouble reading the text. To ensure legibility, use minimum 18 points font size and sans-serif typeface (Figure 91), since these are clearer when displayed on projectors than the serif fonts.

Use legible fonts

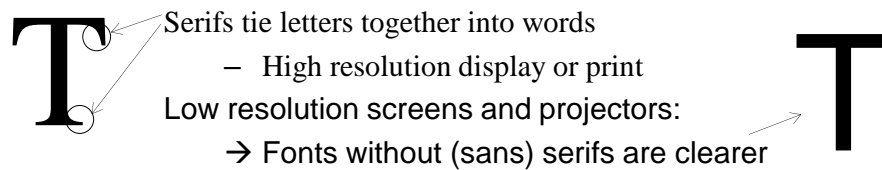


Figure 91. Typefaces with and without serifs. A structural model.

Slides are also often printed as handouts and reduced in size to accommodate more slides on one page. Font size 14 on the original will then become tiny and difficult to read for the long-sighted, while the near sighted have trouble reading 14 point size on the screen. The conclusion is:

4. Minimum 18 point font size with a sans serif font.

The other factor which affects legibility is the contrast between the text and the background. Black on white or white on black are safe, but nearly all other combinations are reducing legibility. Backgrounds that can be chosen in a commercial presentation program may also hamper legibility. The yellow marker colour is the only background colour that actually improves legibility, and should therefore be used for emphasizing. Black letters on a light blue background may help dyslectics, and this combination is in general fine too. Visibility of figures also require sufficient contrast, and even if the contrast looks good on a computer screen, a projector might require a larger difference between light and dark in order to deliver easy to see pictures. Hence,

5. Keep contrast between text/graphics and background close to black versus white.

More thorough introductions to slide design can be found in (Duarte, 2008) and (Reynolds, 2010). Many instructions can also be found on the web, for example at SlideShare.

5. Divide training into 30 minutes modules and include problem solving modules

Chapter 8. Training for transfer

The learning objective of this chapter is to be able to design training that maximises the possibility for using learnt competence in business tasks after the training.

8.1. Transfer

Seen from the training perspective, learners have some competence when starting up, and hopefully a bit more when completing. This new competence is what the learners will bring back to the tasks, and the effect of the training on the changes in business tasks is called *transfer*. We can illustrate the process as in Figure 92.

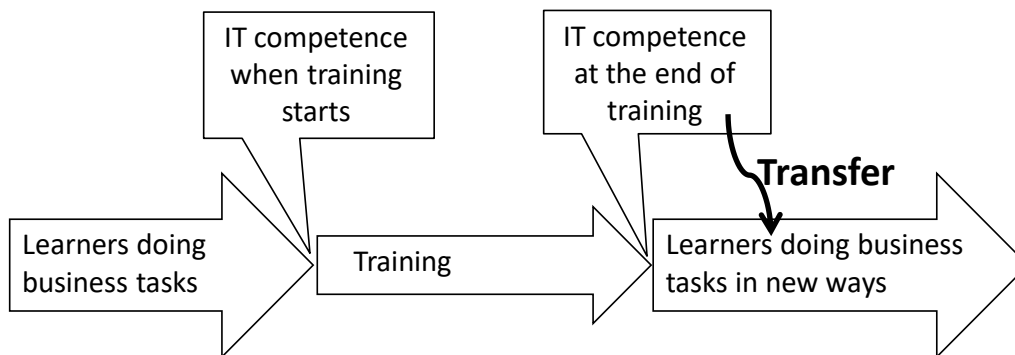


Figure 92. A transfer model of training. Competence is learnt in a course and transferred to tasks.

Some project managers may believe that after users are trained, they will master the application and use it. This assumption is in general false, and the problems of transfer of competence from courses to the activity where the competence is to be used have long been acknowledged. Factors influencing transfer can be divided into training design and the business where the transfer is to take place. Training design will be covered in the next section.

Business factors which are important after training include the opportunity to perform, social influence and support (Grossman and Salas, 2011). The two first ones will be considered here, while support will be considered in Chapter 11 and Chapter 12.

The opportunities to perform include the obvious condition that the IT system is up and running when the trainees return to work. If training is provided weeks before the IT system is put into operation, the users will forget much of their new competence, so the possibility for transfer has decreased seriously (Finnegan, 1996, Karuppan and Karuppan, 2008). On the other hand, if training is provided long after a new system is installed, the users may react like Edita. When coming to training, she brings a negative attitude and may blame the trainer for the delay. In a Dutch study, a

Timing—Edita:

Why training this late? The system has been up for ages, but nobody knows how to use it.

positive attitude towards the information system was found to have a greater impact on use than satisfaction with the training had (De Waal, 2012).

Opportunities to perform include also the facilities for adoption of technology, as presented in the revised technology acceptance model in Section 3.1.

Another factor of the technology acceptance model is social influence, and this is also an important factor for transfer. Peers and managers using a system and encouraging those who have attended training to use it are favourable conditions, as well as incentives for use and remediation for lack of use.

Chances of transfer are increased if the training motivates, includes practicals with imitation of instructions and problem solving, improves self-efficacy and provides an environment that is similar to the business. Figure 93 summarises beneficial factors for transfer during and after training.

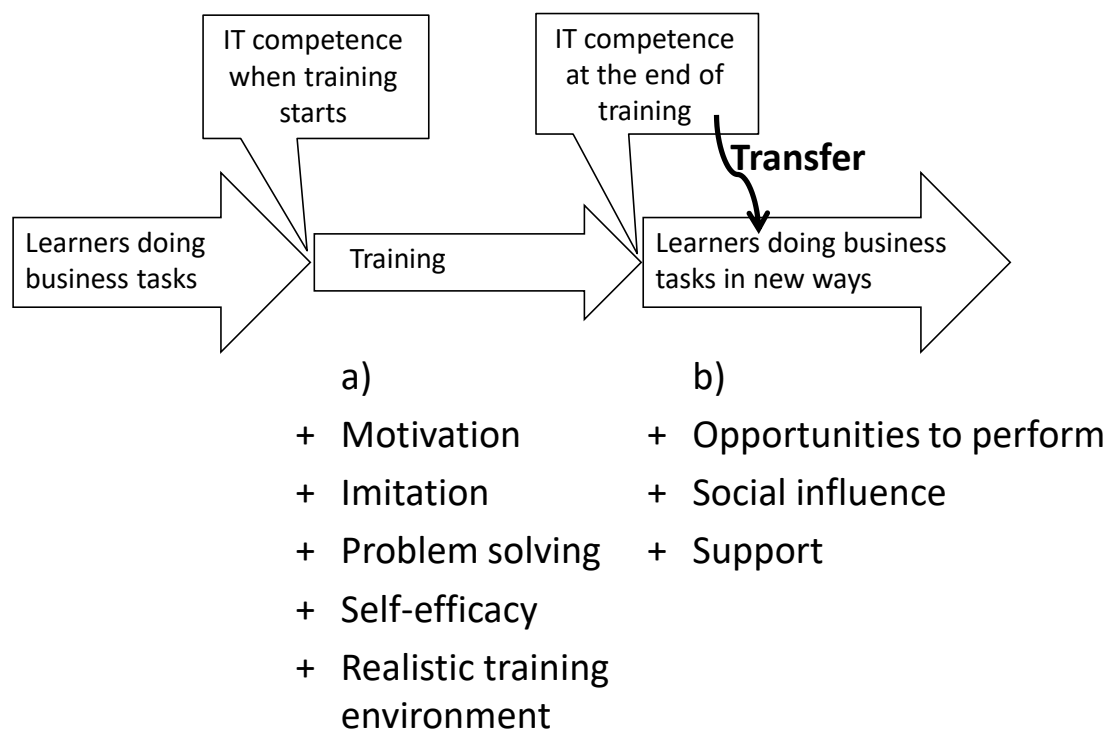


Figure 93. Factors improving transfer a) during and b) after training.

8.2. Motivation and objectives

Learners' motivation improves when clear objectives are set and the learners accept these as their own goals. Trainers therefore need to specify training objectives concerning

- **Outcome.** The change in the business tasks as a result of transfer.
- **Learning objective.** The IT competence to be gained at the end of the training. The competence after training is a means for achieving the outcome.

Objectives become clearer when expressed as what the trainee should be able to do after the training, instead of “knowing,” or “having received an introduction to.” It is reasonably easy to check what people can do, while “knowing” is vague, such that neither the trainees nor the trainer could be sure as to who knows what. In order for the trainees to understand and accept the outcome, it needs to be presented, given reasons, and discussed with the trainees.

Both the outcome and the learning objectives for the training in Shipping International are expressed in behavioural terms. During training, the trainees should reach the learning objectives, while reaching the outcome describes the transfer, which is beyond the trainer’s control.

The learning objectives for the 8th graders are more general, since school children are supposed to learn skills that are applicable also outside of school. The outcome is nevertheless made specific for two areas of application within the school. Such short-term outcomes may be more motivating for the pupils. The outcomes also enable checking whether the kids are actually able to apply their spreadsheet skills outside the IT class.

The last learning objective for the 8th graders says “being able to explain ...” This is a way of expressing understanding in

behavioural terms. A student who can explain spreadsheet formulas to others has understood the concept, while those who can only select a formula in the spreadsheet only have skills.

Training objectives—Shipping International:

The company has bought a new intranet communication system to replace e-mail and Skype.

Outcome: After training, trainees stop using e-mail and Skype, convert files to the new medium and start using it.

Learning objectives:

- Being able to use the system for
 - asynchronous messaging with specified people (e-mail)
 - synchronous video communication with specified people
 - publishing documents and videos to all employees
- Being able to convert from e-mail to the new system by importing message threads

Training objectives—8th grade:

In the IT training, there is one month for spreadsheets.

Outcome: After training, the students are able to use spreadsheets in biology and the grammar project.

Learning objectives:

- Being able to use spreadsheets for
 - structuring data in columns
 - calculating totals and average
 - calculating percentages
 - drawing bar charts
- Being able to explain
 - Cell-referencing and formulas

Since understanding improves retention, pupils who have understood formulas are more likely to transfer their skills to biology and grammar.

The 8th graders had an explicit goal of understanding, and the learners at Shipping International must probably have to understand how messages and documents are organised in their new system. Understanding is in general known to enhance transfer of competence from courses to work (Bransford, 2000). Improved understanding also helps users remember longer during periods of not applying a software (Karuppan and Karuppan, 2008). Training for transfer should therefore address understanding.

When using IT in the business, conditions will always vary, so the benefit of problem solving competence should not be surprising. Transfer also requires the ability to continue learning constructively and independently in post-training situations (Van der Sanden and Teurlings, 2003).

People's general cognitive ability is another trainee characteristic that influences transfer (Grossman and Salas, 2011). Assuming that course participants cannot be selected according to their intelligence, trainers have to deal with the trainees attending a course, being an in-service course in an organisation or a class in school. Trying to improve problem-solving skills and understanding can contribute to improvements of general cognitive abilities.

8.3. *Realistic training environment*

The last factor to consider during training for improving transfer is to make the training as equal to the business as possible. When back in business after training, there is no trainer who demonstrates on a projector or walks around helping out. Hardware, software, data, assignments, documentation and peer trainees can be made similar, however.

Even if training is carried out on one hardware brand and the trainees use another at work, the computers may be similar enough. Response time and connectivity might be issues, though, if training concerns using databases on servers, possibly through internet connections.

Classroom computers are often set up with a separate server to avoid interference with production systems. Therefore, trainees have little opportunities to experience slow responses during training. If there is connectivity trouble in the business, the trainees have not learnt how to work in such circumstances and might give up the whole system for such reasons. Practicing with slow connections during the last part of training and discussing what to do might be a way to prepare the trainees for the not so ideal set up at work.

It may be obvious that the software used in training should be equal to the version at work, but this is not always the case. First, there may be several versions of a package in use, and if the training is centralised, the classroom might have only one of these versions. Second, for a web-based system, the browser and operating system may influence the user experience, and these components differ from computer to computer.

If spreadsheet training only uses accounting data as examples, while the participants are not familiar with accounting, they may not understand the meaning of formulas or columns. Making the trainees bring their own data to the training can resolve this issue, making sure

that the trainees don't misunderstand the technology because they don't share the information competence area of the trainer.

Also, courses go wrong due to the trainer not being familiar with the business of the learners. If the participants will use spreadsheets for statistical analysis of the local flora and the trainer gives assignments on analysing a budget, transfer is severely hampered.

Instruction sheets, models and other documentation provided during training can be brought back to the business. If training is taking place on specific classroom computers with soft copies of the documentation, the learners need to get an electronic copy in their hand, sent to their e-mail or published on the web. Trainers who say "you can find it on our server" are not aware of the poor impact of context-free user documentation, as also mentioned under the skill training section 2.4.

Last, but not least, there are co-learners in courses. If more people who collaborate in business participate in the same course, they can also collaborate on learning the IT after courses. Therefore, sending only one individual to a course means that there is nobody else around to ask when stuck. The result might be that the IT is abandoned and the expenses and time of training are lost.

8.4. Summary

When competence learnt during training leads to changes in the way the trainees carry out their business tasks, we say that there has been a transfer from training to business. Transfer can be improved during training by motivating the trainees, by letting them imitate and learn problem solving and by strengthening their self-efficacy. Transfer is eased by making the training environment similar to the business too. For transfer to happen, the trainees need to be able to use the system after the training.

6. Organise training at the same time as the system is installed.

7. Train a local group of users, not only individuals.

Chapter 9. Evaluation of training

The learning aim of this chapter is to be able to design appropriate evaluations of training.

The guidelines for creating learning material constitute a basis for heuristic evaluation, see Section 6.4 for brief explanation. This chapter will cover systematic evaluation of other aspects of training, including the teaching, the organisation of learners, learning outcome, impacts etc.

Organisations evaluate their activities to find out whether these are worth the cost or whether they can be done in a better way the next time. In general, there are four levels of evaluating training (Kirkpatrick, 1959, Kirkpatrick, 1975, Kirkpatrick and Kirkpatrick, 2006)

1. *Reaction*. Reaction is the participants' opinion of the course. It is normally gauged during the training session. For example, the reaction can be found through a questionnaire to the participants asking their opinion of the course and the trainer.
2. *Learning*. This is an assessment of what the participants learnt during the course. An exam assesses learning, but in order to evaluate training, the exam should be administered also before the training, such that the difference in competence before and after can be found.
3. *Behavioural change*. An investigation of people's use of the IT when back at work or in other activities. For example, ask the participants about which functionality in a software system they use two months after the course.
4. *Impact*. This is a measurement of changes in organisational performance, for example the number of clients, which can be taken care of by means of new IT learnt in the course.

Levels 3 and 4 evaluate the outcome of training, requiring transfer of competence from course to work, see Section 8.1. The four levels are ordered in time as shown in Figure 94.

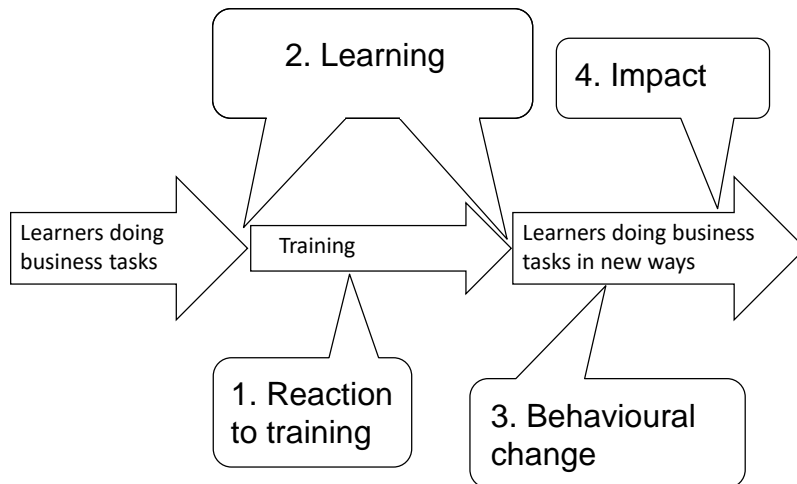


Figure 94. The timing of four ways of evaluating training.

A combination of evaluations will normally provide more relevant results than a single one. For example, an evaluation of behavioural change might find that some of the intended effects of the IT had not come about. Then an evaluation of reaction to training might point to particular weaknesses in the teaching. It might as well happen that evaluation of learning concludes that the staff has learnt what was intended, but the evaluation of impact did not demonstrate any change. This could mean that the training is fine, while transfer did not take place. There might, for instance, be no opportunities to use the competence at work.

A review of research on training in organisations showed that the effect sizes of training for the four ways of evaluation were similar (Arthur Jr. et al., 2003). However, within each study, there were large variations between the effects measured in the four ways. Therefore, we cannot assume that changes of behaviour will occur, even if we can observe that learning takes place during training.

The levels do not imply any causal relationships between reaction, learning, behavioural change and result (Holton III, 1996). Learner motivation will influence the learning process and enabling factors in the organisation have major consequences on change and impact. Despite this known weakness, the Kirkpatrick model dominates as an evaluation framework amongst practitioners (Aguinis and Kraiger, 2009).

9.1. Evaluation of reaction to training

Learners' reactions can be observed in the class as spontaneous statements concerning anything taking place there. The trainer might hear

Now I got it right.

I don't see the point of this topic.

Good I got to know you, so that I can ask you later.

The lunch was delicious.

Statement 1 concerns the learning process, while no 2 is about the motivation for some of the course contents. The third one hints at an important organisational issue, namely that the learner has met somebody whom can be approached for help during transfer. The fourth statement concerns the course environment. Knowing the learners' reactions to any of these topics may be relevant when revising courses.

A trainer observing learners constitutes no systematic approach to evaluation, however. If observation is wanted, an independent person will be able to observe both the learners and the trainers systematically. If no additional person is available, interviews or questionnaires are alternative ways of gauging the learners' opinions. Questionnaires have the advantage of anonymous responses.

Questions in interviews or questionnaires could address any of the topics in the examples above. Since learning and transfer might depend on several factors, and since there can be large individual differences, getting acquainted with other users might be as important as motivation and course environment.

Concerning the business side, questions like

Did the course address your needs in your job? If some needs were not met, which ones?

could be used. The first, closed question addressing the needs could be responded to on a scale from 1 to 6. The latter, open question can provide knowledge on specific tasks and functionality to include.

Participants' reaction to the IT and information contents of the training could be addressed by asking

Did the training provide a sufficient background for understanding the IT system? And for using it?

Did the course explain the data in the system sufficiently?

The latter question should be more specific, for example

Did the course explain the account types thoroughly enough?

if an accounting system was to be taught. Again, scales could be used in the response, and the closed questions could be followed up by open-ended ones as in the activity fit case.

In an evaluation of user training, the users responded that the hands-on exercises with real world data were useful for keeping their motivation (Mahapatra and Lai, 2005). This response was useful for the trainers, who became even more focused on crafting training material to fit the background and expectations of the learner group.

9.2. Evaluation of learning – assessing competence

Assessing IT competence is an activity, which can take place for many reasons in various settings.

- An organisation wants their staff to be at a certain competence level, hence they organise a test for everybody to take.
- When planning a course for a group of participants, we want to know their competence, such that the course can start at the appropriate level.
- When a course plan has been settled, we screen the possible participants, such that those at a too low or too high level are channelled into other training.
- Employers test the IT competence of job applicants.
- Applicants document their competence through completing a certified test.
- A school administers an exam in their IT class.
- For evaluating a training course, we assess IT use competence prior to the course and afterwards. The difference between the two levels will tell us the contribution that the course has made.

Levels of IT competence have been described as skill, understanding and problem solving competence, and tests of competence can be arranged accordingly. Basically, skills are assessed through practical tests, and understanding with written or oral questions and answers. Testing problem solving competence could involve both practical and theoretical tests. Telling the accountant Rigo, who sits in front of a computer to

Print the spreadsheet.

would test his IT skills. Assuming that a reimbursement claim is registered in a database,

Check the reimbursement claim.

is a task which can be given him to test his IT and business skills. In order to test his understanding, the following types of questions could do:

What is a spreadsheet program?

What is the purpose of double entry bookkeeping?

People can express competence that is only at the level of skills also, like Kirsten talking about her sequence of tasks in Section 3.2. Asking questions like

Which menu choices and buttons do you use for creating a numbered list, which starts at the number 3?

does not require a response at the level of understanding.

Questions concerning business purposes could be open ended. Focusing on a software tool:

Note down a task in your job where you use or could use spreadsheets. What is the advantage of using a spreadsheet in this task?

Taking a task as the point of departure, we could design a test for the level of understanding possible changes:

You are organising a sports event. For which tasks can IT be helpful, and which IT hardware and software would you use?

Skills test:

Create a spreadsheet for currency conversion.

Multiple choice questions

Multiple-choice questions are convenient when testing the competence of a larger number of users, since the responses can be assessed automatically. Properly designed, the multiple-choice questions are as reliable as any other measurement of learning. Three principles for multiple-choice questions concern the number of options, the length of options and the wording.

First, the optimal number of options for each question is 3 (Rodriguez, 2005), assuming that all options are realistic, such that someone would pick them. For instance, the option “Sending money to your mother” is an unrealistic option to the question “What is a Style in a text processor?” Having only two options, the test becomes much poorer in discriminating between the learners. Four or more options do not contribute much in discriminating and it takes more time for the learner. Completing 4 questions with 3 options each will yield a larger difference between good and poor responses than 3 questions with 4 options each, even if the two alternatives may consume the same amount of time both for the learners and the test author.

Second, people who are completely ignorant of the correct answer have a tendency to go for the longest option as the correct one. Therefore, to avoid random guessing yielding high scores, a wrong option should be the longest one for most questions. However, if you consistently follow this rule, the learners will detect it and thereafter avoid the longest alternative. Hence, some variation is necessary (Gronlund, 1998)

Third, there should be no correspondence in wording between the question and the correct option (Gronlund, 1998). Consider the following question:

What does it mean to store a file in the cloud?

- a. It will stay permanently in your computer.
- b. It will be stored in a cloud server of a trusted provider.
- c. It will be stored such that anyone can find it by searching with Google.

This question fulfils the first and second criteria, since there are three options and one of the incorrect options (c) is longer than the correct one. However, the word ‘cloud’ is used in the question and in the correct alternative (b), associating the two, thus pointing the ignorant learner to this alternative. The third criterion is broken. The case can be remedied by deleting the word ‘cloud’ in alternative b.

A multiple choice question concerning IT in a task:

Which of the following activities can you use a spreadsheet for when planning a new house?

- a. Draw the floor plan.
- b. Compare the cost of different floor covers.
- c. Find the formulas for areas of rooms and walls.

The incorrect response (c) includes the word ‘formula,’ which the learner could associate with spreadsheets. This option is therefore misleading the ignorant learner, and the question complies with all three criteria.

The question could also be based on the business:

You are planning a new house. Which of the following statements are correct?

- a. The contract can be written with Adobe Reader.
- b. Tables in text processors can be used for comparing wall and floor colours.
- c. I can communicate with the architect through sharing a folder on Dropbox.

Questions should comply with the learners’ experience, such that they are familiar with the background of the question. When testing for an organisation, addressing actual work tasks in the questions would be appropriate.

Assessing problem solving competence

Open ended questions for checking experimentation competence could be:

Find out what goes right and wrong when copying from a pdf document and pasting into a text processor.

Here is a new application. Find out what it does.

Multiple-choice questions could assess the part of experimentation that involves generating hypotheses. A test could be:

You have attached a file to an e-mail to Bob. Then you discover that the file contains some statements which you do not want Bob to read, so you make some changes in the file and save it. You wonder whether Bob will get the changed file. Which of the alternatives below will give you the answer you need? The alternative should make us learn what happens, not just solve the problem of sending a changed attachment this time.

- a. Remove Bob from the list of receivers and enter yourself instead.
- b. Send the file to yourself from another e-mail account.
- c. Remove the attachment and then re-attach the file before sending.

The flip side of such questions is that they do not necessarily test experimentation competence. If Manu, who answers, is very familiar with the e-mailing, he might answer correctly because he knows a lot about his e-mail service, and not because he is clever at setting up experiments.

IT users can also learn through troubleshooting. A general way of checking users’ ideas about troubleshooting is asking about repetitions:

You observe that the computer responded in a way that you did not intend. You repeat it, and this time it worked out. What can the reason be?

- a. The quantum mechanical circuit at the motherboard kicked in.
- b. You made a typing mistake the first time.
- c. The hard disk crashed.

Asking questions like the ones presented for experimentation is also possible, having the same drawbacks, in the sense that we cannot always know whether we are testing the users' troubleshooting competence or the mastery of the particular technology.

In addition to questionnaires, problem-solving competence also lends itself to observation. The trainer can observe how the learner handles the software and looks up resources for help. For a pair of learners, also their conversations can be observed. Observations of problem solving come out with significantly different results than questionnaires to the same users (Novick et al., 2007). Assuming that the difference is due to poor memorisation of details of actions, observations give a more correct account of problem solving.

Trainers normally observe learners' activities in order to adjust the training on the spot. Such evaluation is called *formative*, which has a profound effect on learning (Hattie, 2009). This chapter concerns the *summative* evaluation, which aims at improving the course the next time it is carried out. However, the observations, which trainers carry out informally on the learners' achievements may also be included in the assessment of learning.

9.3. Evaluation of behavioural change

Evaluation of reaction to training and of learning can be carried out in any training course, independently of what the learners are going to do afterwards. Evaluation of behavioural change is intended for company specific courses, where it is possible to approach the course participants some weeks or a few months after the course has ended. If the participants use IT in their work in a different way from what they did before, a behavioural change has taken place. Finding the type and extent of such change is what evaluation of behavioural change aims at.

When the IT to be learnt is a software package on a server, users' operations can be logged. Data on changes in user behaviour can therefore be found by analysing such logs. Details on individuals' use of specific functionality can be summarised with much less effort than asking users. Like any surveillance, employees should be informed that their behaviour on a computer system is logged. Some countries or trades may have regulations or agreements concerning surveillance systems.

Logs do normally not tell us all we want to know, however. They provide statistics, but not the reasons why the staff use specific functionality or avoid it. Again, observations, interviews and questionnaires constitute possible ways of investigation. Observation takes more time and provides more detail per individual, while questionnaires reach a larger number of users with less detailed response. Individual and group interviews lay in between the two.

When designing questions aiming at finding reasons, the factors from the Technology Acceptance Model (Section 3.1, p.27) can be a starting point. Usefulness, ease of use, social

pressure, facilitating conditions and combinations of these are likely reasons for the use or non-use of a specific functionality. For example, assume that the logs have demonstrated that the functionality for search for similar cases in the client system is used to a very little extent. A relevant question in a questionnaire to the users could be:

Regarding the “Search for similar cases” in the client system, rank your agreement with the following statements on a scale from 1 to 6 (1=Disagree completely, 6=Agree completely):

I find this search very useful in my work.

This search is easy to use.

The majority of my peers use this search.

The computer system responds quickly on this search.

A low score on usefulness may have two reasons. Either, this has not been taught properly during training, such that the users have not understood its usefulness, or the functionality is of minor value. If users say that it is difficult to use, the training might be to blame. If they say that it is easy, the reason for low use may be that it is actually of little value. Peer pressure and technical conditions should also be taken into account.

9.4. Evaluation of impact

The fourth level of Kirkpatrick’s model (Kirkpatrick, 1975) is evaluating whether the introduction of IT fulfils its goals, and how the training has contributed to the impact. While simple ways of measuring organisational performance, like the bottom line in a company, may exist, drawing inferences from changes in results back to training courses are often difficult due to a high number of intervening factors. For example, the business may observe that the number of clients served after introduction of the new client information system has risen by 20%. The reason for the change may be that the market has increased, that clients are served better, that a new system has been introduced through successful training, or that the staff has managed to get the new system working despite poor training.

A more feasible way is selecting some outcome that is closer to what was trained. Instead of the number of clients, we could for example measure the time for serving each client. This relieves us from dealing with markets or other external factors as possible explanations. Still, the possible impact of training on efficiency of client handling has to be established.

In an evaluation of training of an information system for reporting health statistics, it was possible to count the number of data items filled at any point in time and also perform some automatic comparisons to judge accuracy (Ngoma et al., 2008). In order to gauge the impact of training, the completeness and accuracy were first measured in several sites, some of which would be trained. After training was completed, the completeness and accuracy were

measured again in the same sites, and the results compared, see Figure 95. The sites, which were not trained constituted a control group. If the sites without training had improved as much as those with training, the training would not have had any effect. The untrained improved by 10% while the trained with 50%, leading to the conclusion that training was effective.

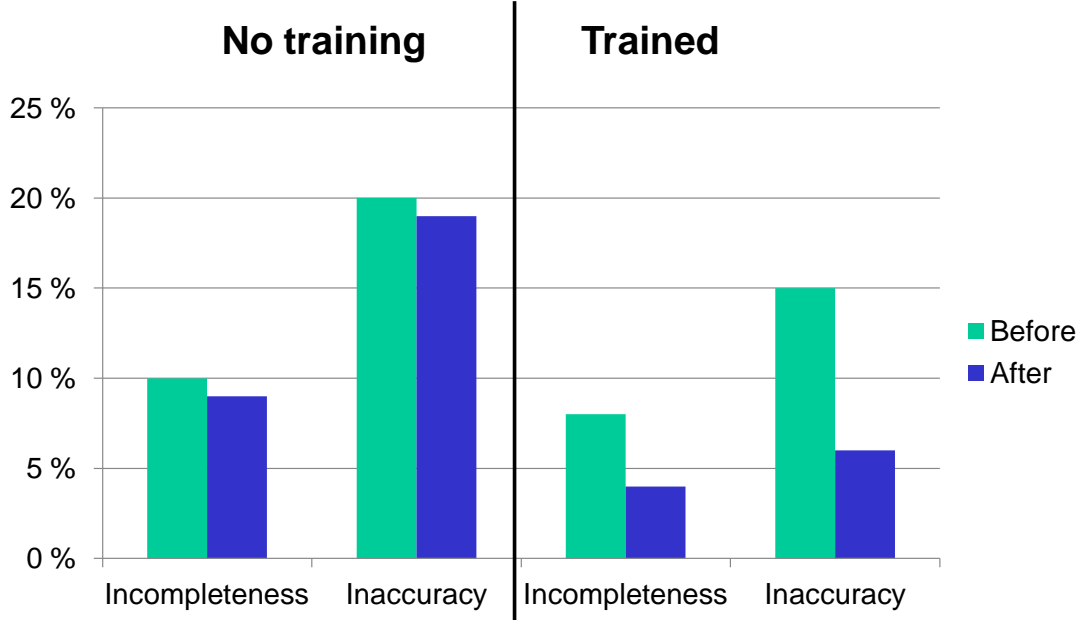


Figure 95. Measurement of results of training. The “No training” sites constituted a control group.

Chapter 10. IT user competence standards

The range of technology being used has needed has evolved over the years. In the 1980's, the file system, individual office applications and possible business information systems constituted the typical collection of IT for users to master. The 90's brought local networks and the Internet, with servers, browsers and e-mail added to the standard repertoire. During the last ten years, Web 2.0, mobile phones, tablets, digital cameras, music players and a number of other personal gadgets have sparked a diversification of modes of interaction as well as hardware. Business systems have moved into the browsers or migrated to enterprise resource planning software.

The continuous expansion of IT applications disables any stable description of the range of IT use competence. However, some comprehensive guidelines for IT user competence have been developed, either for the general public, for special occupations or pupils at school.

10.1. Standards and guidelines

An approach to the latter is the FITness (Fluency with IT) report, which describes a comprehensive set of skills, concepts and capabilities, the latter corresponding to problem solving competence to a large extent, see Table 5 (Committee on Information Technology Literacy, 1999). Contrary to many textbooks on software use, it addresses concepts and principles. FITness go even a step further, by including programming and algorithms, which is considered beyond IT user competence as described in this book.

Most organisations depend on their employees being capable of operating business critical systems. For example, the cashier needs to be able to check out goods and register payment, police officers need to know how to use the communication equipment, and the air traffic controller must be fluent in the IT system mapping the flights. In the latter case, and in other high risk tasks like handling surgical equipment and nuclear power plant control, the operators might have to be certified. A detailed specification of the competence, including information technology, will be required for constructing certification tests.

Table 5. IT user competences as described in FITness (Committee on Information Technology Literacy, 1999)

Intellectual Capabilities	Information Technology Concepts	Information Technology Skills
<ol style="list-style-type: none"> 1. Engage in sustained reasoning. 2. Manage complexity. 3. Test a solution. 4. Manage problems in faulty solutions. 5. Organize and navigate information structures and evaluate information. 6. Collaborate. 7. Communicate to other audiences. 8. Expect the unexpected. 9. Anticipate changing technologies. 10. Think about information technology abstractly 	<ol style="list-style-type: none"> 1. Computers 2. Information systems 3. Networks 4. Digital representation of information 5. Information organization 6. Modelling and abstraction 7. Algorithmic thinking and programming 8. Universality 9. Limitations of information technology 10. Societal impact of information and information technology 	<ol style="list-style-type: none"> 1. Setting up a personal computer 2. Using basic operating system features 3. Using a word processor to create a text document 4. Using a graphics and/or artwork package to create illustrations, slides, or other image-based expressions of ideas 5. Connecting a computer to a network 6. Using the Internet to find information and resources 7. Using a computer to communicate with others 8. Using a spreadsheet to model simple processes or financial tables 9. Using a database system to set up and access useful information 10. Using instructional materials to learn how to use new applications or features

Standards are operationalised through curricula and tests. Competence tests are used in level 2 evaluation of training, see Section 9.2. General tests of competences are presented below.

10.2. Tests

Both commercial and other organisations have developed IT user competence tests, see (Covello, 2010) for an overview. Three major ones are:

- Educational Testing Service is a US based, non-profit organisation, known for its Test of English as a Foreign Language (TOEFL). They offer the iSkills Assessment, which measures IT literacy (Educational Testing Service, 2011)
- Certiport is a commercial actor, also providing courses and tests for software professionals. (Certiport Inc., 2011)
- ECDL Foundation is a non-profit organisation providing the European Computer Driving License, also known as International Computer Driving License (ICDL). It was founded in 1995 by the Council of European Professional Informatics Societies in order to improve digital literacy across Europe. Later, it has gone intercontinental, and 11 million people have conducted tests given in 41 languages. (ECDL Foundation, 2011)

We will look at some sample questions to see how the tests are constructed. The ECDL is divided into 13 modules, mainly according to software types. In addition, there are three general modules:

- Concepts of ICT
- IT Security
- Project Planning

About Module 1, the ECDL / ICDL Sample Part-Tests (ECDL / ICDL, 2009 Module 1, p 1-2) says:

Module 1 Concepts of Information and Communication Technology (ICT) requires the candidate to understand the main concepts of ICT at a general level, and to know about the different parts of a computer.

The candidate shall be able to:

- Understand what hardware is, know about factors that affect computer performance and know about peripheral devices.*
- Understand what software is and give examples of common applications software and operating system software.*
- Understand how information networks are used within computing, and be aware of the different options to connect to the Internet.*
- Understand what Information and Communication Technology (ICT) is and give examples of its practical applications in everyday life.*
- Understand health and safety and environmental issues in relation to using computers.*
- Recognize important security issues associated with using computers.*
- Recognize important legal issues in relation to copyright and data protection associated with using computers.*

Assume that we constructed open ended test questions for these learning objectives, like:

What is the Internet?

Karl responds:

A network through which we access all places in the world

Karl is describing a function of the Internet, so he is at the functional understanding level. His understanding may be limited, since he does not specify the different types of functionalities, like the www, email, chat, etc. The ECDL has multiple choice questions for testing understanding (ECDL / ICDL, 2009 Sample Part-Test 1.2, p 3):

Which one of the following statements about the Internet is TRUE?

- a. The Internet is a global network that links many computer networks together.*
- b. The Internet is a private company network.*
- c. The Internet is a visual representation of linked documents.*
- d. The Internet is a network operating system.*

The statements a-d describes the Internet at the structural level. Given that Karl responded like above, he would most likely tick the a alternative, so his test result would show that he understands the Internet at the structural level.

Concerning software, the spreadsheet module is selected as an example (ECDL / ICDL, 2009 Module 4, p 1):

Module 4 Spreadsheets requires the candidate to understand the concept of spreadsheets and to demonstrate an ability to use a spreadsheet to produce accurate work outputs.

The candidate shall be able to:

- Work with spreadsheets and save them in different file formats.*
- Choose built-in options such as the Help function within the application to enhance productivity.*
- Enter data into cells and use good practice in creating lists. Select, sort and copy, move and delete data.*
- Edit rows and columns in a worksheet. Copy, move, delete and appropriately rename worksheets.*
- Create mathematical and logical formulas using standard spreadsheet functions. Use good practice in formula creation and recognize error values in formulas.*
- Format numbers and text content in a spreadsheet.*
- Choose, create and format charts to communicate information meaningfully.*
- Adjust spreadsheet page settings and check and correct spreadsheet content before finally printing spreadsheets.*

The learning objective specifies a series of skills, which are described in some detail. The “concept of spreadsheet” is not explained, so the understanding part of the goal is unclear. The tests are mainly of the practical kind, for example (ECDL / ICDL, 2009):

Enter a formula in cell F5 with an absolute cell reference for one cell only that divides the content of cell E5 by the content of cell E11. Copy the formula in cell F5 to the cell range F6:F10.

So the goal of skills seems to correspond to the test type. An open ended question which addresses understanding is also included (ECDL / ICDL, 2009):

Which of the two cells F4 or F5 displays good practice in totalling a cell range? Enter your answer in cell B14.

Answers to open ended questions like this one can be assessed right or wrong or according to a scale, for example skill – functional understanding – structural understanding. .

Responses to multiple-choice tests are easy to assess. Assessing whether the candidate has written a correct formula in a spreadsheet also requires only a quick view. Reading, interpreting and grading an open-ended answer is much more tedious.

ECDL’s division of IT competence into software types hinders questions, which relate concepts from two IT tools. For example, the following question could not be included:

What is the similarity between master slides in presentation programs and styles in text processors?

- a. They provide information for the table of contents.
- b. They enable coherent formatting of the file.
- c. They enable import of slides into word processors.

Also differences between concepts could have been included if the tests could span more applications, for example:

What is the difference between tables and column layout in a text processor?

- a. Tables are imported from a spreadsheet, while column layout is generated within the text processor.
- b. Column layout is the vertical sequence of cells in a table.
- c. Tables are composed of separate cells of text, while column layout means that the text is displayed in sequential, vertical stripes.

The Instant Digital Competence Assessment (iDCA) is a recent test aimed at 14-18 year olds (Calvani et al., 2012). It is organised in the three dimensions technology, cognitive and ethics, instead of the organisation according to IT applications found in the ECDL. The cognitive dimension addresses management and evaluation of data. Ethics covers general principles and constraints for IT and information use and is in the business area. Since iDCA is not compartmentalised into software products, it could cater for the two questions above.

iDCA consists of multiple choice questions and does not address skills by asking the respondents to carry out operations on the computer. Its technological area addresses problem solving.

Competence tests versus self-reporting

Performance on competence tests have been compared with students' self-reporting of their competence level. The latter was gauged by users responding to statements like:

- My spreadsheet skills are good.
- I am a more experienced spreadsheet user than most of my peers.
- I feel competent to use a range of applications.
- I feel comfortable opening and saving spreadsheet files.

The respondents would agree or disagree on a scale.

Most studies conclude that there is no correlation between how people self-report their level of IT competence and how they perform in tests (Larres et al., 2003, Merritt et al., 2005, Sieber, 2009, van Vliet and Kletke, 1994, Ballantine et al., 2007, Sink et al., 2008, Grant et al., 2008). Low performing users overestimate their capabilities. On the other side, one study found a correspondence between self-reporting and test results of IT competence (Hakkarainen et al., 2000), and this is in line with the general findings that school students have a very accurate conception of their level of achievement (Hattie, 2009). In school, students are used to comparing their work with grades, which provides a good basis for reliable self-reports. IT use is a minor topic in schools, hence pupils may not have had such experience concerning their IT competence.. Since the majority of IT competence studies do not find correlations, we consider self-reports and levels of competence uncorrelated.

A consequence of users overrating their competence is that trainers and educators who rely on self-reporting assume a higher competence level of their students than what is the case. For example, in a study of 173 college students 75% perceived their word processing proficiency as high and 20% as average (Grant et al., 2008). In the skills test, questions were grouped as basic, moderate and advanced. Table 6 shows the ten tasks which the researchers had characterised as moderately difficult. Tasks with correctness rank 1-7 are operations on the main document text flow, so no understanding of the data structure of document files is necessary. Tables and headers (rank 8-10) introduce independent text flows, requiring the students to alter their structural understanding of a document as a single sequence of characters to a multi sequence model. The majority of students seem to be stuck in the single text flow understanding, even though they characterise themselves as average or highly proficient.

Table 6. Performance of college students in the US on word processing tasks (Grant et al., 2008)

Moderate tasks	Correct performance	Rank
Count words	91%	1
Add bullets	88%	2
Highlight text	82%	3
Find and replace text	60%	4
Use the Thesaurus	57%	5
Insert a date	54%	6
Justify a paragraph	47%	7
Enter data in a Word table	33%	8
Insert rows in a table	27%	9
Create a document header	8%	10

Although the competence tests do not distinguish clearly between a skill and understanding level, this test indicates that college students have a limited IT understanding. They might base their high self-confidence on their skills in getting a document produced.

10.3. Differences in IT competence levels

This book has described three levels of individual user competence: skills, understanding and problem solving. Results of measurements of IT user skills worldwide follow their own ways of grading competence.

An international survey of digital reading competence at school level 5 concerned the pupils' ability to navigate and find appropriate web pages efficiently (OECD, 2011). Also, they were assessed on their skills in evaluating the information retrieved. Interestingly, South Korean children outperformed the students from the other countries, including New Zealand and Australia, Japan, European and South American countries in this ranking. Africa and North America were not represented. While a common opinion may be that people in the newly industrialised countries in Asia are well versed in electronics, while the European children are more literate in the original sense, this OECD study only partly supports such a view. Korean students perform better in digital than in print reading, while the opposite is true in Eastern Europe and South America.

Girls outperform boys in both digital and print reading (OECD, 2011). The same is found in a study of college students in the US (Hignite et al., 2009). An ICT literacy test amongst 6 and 10 year old children in Australia included a range of tasks typical for the age groups. Both technological and business fit competence were tested (MCEECDYA, 2010). Also in these areas of competence girls performed better than boys. A test of high school students in China with iDCA showed no performance difference between the sexes (Li and Ranieri, 2010), while boys performed better than girls with the same test in Italy (Calvani et al., 2012).

The findings that girls outperform boys on technological topics contrast the results from more than 30 previous studies summarised in (Cooper, 2006). One reason for this difference could be that the former IT assessments were more biased towards technology, while interpretation of information and use of IT in tasks have been given larger emphasis in recent years. Another factor may be that young children now grow up with mobile phones and social media on the internet, and that communication is more aligned to girls' interests, while boys are competing in computer games. The recent studies showing female superiority were carried out amongst children, while former studies have addressed all age groups.

Socio-economic factors are generally influencing competence levels, and this is also the case for IT related competences (OECD, 2011). Having a computer at home has a positive effect on children's IT literacy.

10.4. Summary

In high-income countries, children play with digital devices from an early age and become skilled at manipulating computers. People in low-income countries may meet the digital age through a mobile phone and few acquire computer skills. Regardless of skill levels, standardized tests show that people may struggle with understanding and problem solving.

Having a certificate of IT user competence may help getting a job, and millions of users have passed such formal tests. Many tests address skills to a larger extent than higher order IT use competence. Users who perceive themselves as skilled often fail tests, which require more understanding.

While girls in high-income countries score higher than boys on competence tests addressing information use, the reverse is true for the pure IT competence. Interestingly, these differences were not found in China.

Part III - Managing development of digital competence in organisations

The previous parts have considered the individual's competence and learning. In order to consider organisational aspects of IT competence, we shift focus from individuals to groups. We will base the identification of a group on people who share a set of activities, called a *practice*. Such groups constitute the units in a theory of learning at work within the class of situated learning theories. Situated learning refers to learning that takes place within the practice where the learning is applied.

Pedagogical theory – Situated learning – Communities of Practice

According to (Wenger, 1998) a *community of practice* (CoP) has three crucial elements; domain, community and practice. The identity of a CoP is defined by a shared *domain* of interest with shared competence for dealing with that domain. Members in a CoP value their collective competence and learn from each other. Second, members in a CoP create a *community* through engagement in joint interactions and discussions, by helping each other, and also by sharing information. They also build relationships that enable them to learn from each other. However, members of a CoP do not necessarily work together on a daily basis. The third characteristic element of a CoP is the *practice*; the doing which provides meaning and structure to the activities. The shared practice is created by practitioners who develop a shared collection of resources such as tools, experiences, and ways of addressing recurring problems. For example, a group of supermarket workers would constitute a CoP when they share the concern for the goods and customers, they interact, discuss and help each other, and they use common tools for sales and pricing of goods.

CoPs often differ from the formal organisational units, appearing neither on an organization chart nor on a balance sheet. In a small shop with a handful of staff, managers may be part of the cashiers' CoP, and in a large organisation, the accountants spread around in different departments may interact sufficiently to constitute a CoP.

Newcomers get socialised into a CoP by imitating its members, and also by getting punished or neglected if behaving in ways which are not acceptable in the community. The members may also tell newcomers explicitly how to behave, and the novices may have attended formal education, which has prepared them for the introduction. When a community of practice receives a new member, it is mainly the newcomer who will have to adapt, while the community is less receptive for changing their practice.

Pedagogical theory – Situated learning – Interaction between CoPs

In line with (Wenger, 2000) and (Cobb et al., 2003), we consider three aspects of interaction between CoPs; boundary interactions, brokers, and boundary objects. In *boundary interactions*, members from different communities take part in common activities. This might be short encounters, like when a manager calls the computer support for getting help in connecting to the network, or longer practices, for example when health managers participate in a course conducted by health information specialists.

A *boundary object* is a material thing which makes sense in more than one CoP, and which has a structure that is common enough to be recognized in both CoPs (Star and Griesemer, 1989). Boundary objects are used for communication between CoPs, and they may provide a common understanding of a phenomenon as well as give rise to misunderstandings. A database could be a boundary object for accountants and computer scientists, where both parties would recognise its ability to store and retrieve financial data. However, the accountants would emphasize its role of representing the financial affairs of their company, while the computer specialists could regard it as an instantiation of a relational database management system.

Brokers are at least peripheral members of two CoPs and can introduce parts of practice from one CoP to the other. A headmaster could be a broker between the community of teaching practice and the community of school management practice in the town. Construction engineers could be members of engineering, architectural and construction work practices, providing some joint understanding between the three partners.

Chapter 11. Superusers

The learning objective of this chapter is to be able to identify groups with different roles as learners and supporters and to specify conditions for these groups developing into communities of practice.

11.1. Roles

Users in general are specialists in their business and the domain of the system. Trainers, as a special user group, are specialists on metacognition, being a central ingredient in problem solving competence, and IT personnel have IT as their expertise. The areas of practice of these three types of specialists are summarised in Table 7.

Table 7. Areas of practice and corresponding communities.

Role	Area of main practice	Examples
Users	Tasks and business	Farmer. Nurse. Cook. Salesperson.
IT personnel	IT	IT support staff. Software developer.
Trainers	Metacognition	School trainer. Business instructor.
Superusers	IT + at least one of the other	

Users are characterised by having the domain of the information system as their primary domain of work. Second, *IT personnel* have IT as their main domain of work, so these are in the IT practice. Third, *trainers* enable learning and have metacognition in their competence base. People working in each of these three roles can constitute communities of practice, since they share a main practice.

Finally, some users or trainers develop more competence in using computers than others, such that they provide computer support to their colleagues, and this group will be called *superusers*. Superusers have at least a better understanding of some software than others and some problem solving competence. A superuser can also come from the IT side and acquire competence from one of the other areas. Superusers will be members of at least two communities of practice, hence become brokers between these communities. They can also develop into a community of superuser practice. Each of these roles will be described in more detail below.

Users

For the majority of IT users, the technology is a means to get work done, and not an aim in itself. Users find IT problems annoying and prefer to spend their time on their primary business. Their shared domains of interests are therefore not IT or data, but any other work area; hence they may constitute communities of non-IT practice. Correspondingly, the eventual learning of IT use taking place in these communities will be of secondary importance to the learning of the main business.

IT personnel

Larger companies or agencies would have IT personnel involved in a mix of activities. Network administration and user support would normally constitute two time consuming ones, while procurement and application tailoring could be other tasks.

The idea of a community of practice is that people share a domain of interest, and we could say that the IT systems and their users in the organization is the domain of the IT personnel. They would normally share information about the technology and its users through lunch conversations, meetings, e-mail, documentation and random encounters in the corridors. Larger organisations could also have a ticket system for storing user requests and responses, where the IT personnel can search for topics with which they are unfamiliar. In these ways they may develop a shared repertoire of cases, problems, software and users, so that they constitute a *community of IT practice*. IT specialists meet users in boundary encounters on the phone and face to face, helping out those who need more IT competence, and they learn about users' business through interacting with them. They also have boundary interaction with other communities of IT practice, e.g. at computer vendors, thus keeping updated in the IT field.

Software companies and IT vendors can also have departments for support. For these organisations, their customers constitute their users, who could be IT departments in other organisations. A newly established, small company might just have a flat structure, where all members carry out development and support. These would constitute a community of practice, where the software product constitutes the shared domain of interest. A big vendor, on the other hand, might have a call centre in India with several hundred staff members who serve customers worldwide. If they have the opportunity to communicate and exchange experience, they may also become a community of practice, where the users' requests and the corresponding responses constitute the shared domain. In between these extremes, there are many medium sized IT companies, where the user support is located in a department of a smaller size, such that the formation of the community is simpler than in the huge call centre case.

An IT department in a non-IT company would use the software and hardware vendors as their lifeline for support. They would engage in boundary interactions with the vendors, and the software and documentation would constitute the boundary objects of these practices.

Trainers

Larger organisations have human resource departments where educationalists are hired for organising and planning training, and who may also act as instructors themselves. Schools are obviously special in this respect, since their teachers have formal pedagogical qualifications. They would normally constitute one or more *community of teaching practice* in each school.

School-teachers and business instructors sometimes also do IT training. In schools, IT competence could be an end in itself or a means for the students to learn other topics. In the latter case, the teacher may be fluent in the subject matter to be taught, but short on the technological competence. Professional teachers bring training methodology and competence

about learning, including knowledge of metacognition, into the realm of user support and training. This pedagogical competence is hardly found amongst IT personnel.

Superusers

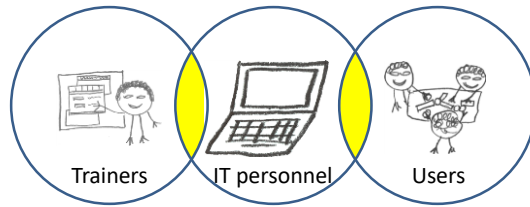


Figure 96. Super-users (yellow) being members of at least two communities of practice and brokers between these communities.

Superusers are users or trainers who have specific IT competence and have taken on the role of supporting their peers in an organization. Other terms for superusers are boundary spanners, business coaches, computer gurus, key users, lead users, local experts, local heroes, peer coaches, power users, subject matter experts, super power users and translators. They have also been grouped into ‘recognised experts,’ who have an extended reputation in the company, and ‘local experts,’ who are consulted by their close peers (Spitler, 2005).

Beware that in computer science, the term superuser is often denoting someone with administrative privileges for a computer or software system, who can grant access rights to others (2014). In this book, superusers are characterised according to their role of helping others and not their software privileges. However, some superusers might also have extended access rights in



I am Mozhdeh and my main job is administration of international students. Since I had been working with archival systems previously, I was selected as the local superuser when the new system Erchive was implemented. I was introduced to Erchive in a training course. The course covered the IT system but not archive codes and whether a document has archive value. Many user requests concern whether a letter should be archived, and then I have to find out whether it has archival value. There is a list where I can look it up, and I can also call the central archive if in doubt. Other user requests concern how to operate the system. Also, I solve logon problems and upgrading to the recent version of the browser. They need that when Erchive is updated.

Each department has a superuser, and we meet 4 times a year. We discuss changes we would like and communicate these to the IT department. We also get informed about changes and have to bring the news to our local colleagues.

یک کاربر حرفه ای کسی است که خود از سیستم بطور مداوم استفاده کند.

the computer system for providing more effective help or for creating accounts for other users.

Superusers could have a primary domain of work completely remote from information or IT, for example nursing, sales or farming. They would therefore belong to two communities of practice; one on the IT side, and another on their primary domain, and they would also be brokers between these communities, see Figure 96. They could influence the communities of IT practice with their main competence, and introduce IT competence amongst others.

The text boxes present three superusers. Mozhdeh had some experience with archives when she did archiving. She became a superuser of an archiving

system after her job was changed, but she still draws upon her knowledge of archives in her superuser role. Oksana is a superuser of a system which she uses frequently in her accounting job. She knows the information in the system, how to operate the IT, and how it supports the business. Sigrun has a computing background and was selected superuser for a web publication system. She is familiar with how it can be used for creating structured web pages.

While these three superusers had been appointed, superusers also emerge spontaneously when no formal appointments are made. In a purchase department of around 100 staff in a Finish company, all staff were provided training when a new information system was installed, but no system of superusers was established (Sykes et al., 2009). During a period of three months after training, all staff had either given or received help from others. On the average, a user helped five others, but some became more central, helping more than ten, thus informally becoming superusers. A summary of studies on organisational learning shows that people learn more from others whom are trustworthy (van Wijk et al., 2008), and superusers can gain trust in their role by proving their abilities. An early study of superusers found that they not only regarded themselves as having better IT skills than others, they also used a larger number of software tools in their work (Eveland et al., 1994)



My name is Oksana, and I do accounting. When staff members claim reimbursement for travels, they fill a form in the human resource system, and thereafter I check the form. I am also helping them getting the information into the right fields. Sometimes I invent solutions to avoid some restrictions in the system. When I cannot solve the problem, I ask staff members to send an e-mail to the central support section and explain the problem, get the answer and to try “to do the best you can on your own.”

I was introduced to the system through a course, but I learnt nothing there. Instead, I have gone through the e-learning material and learnt it that way.

Last week, I presented the system to a large group of people from all departments in a training course. We also helped them out in their practical tasks. Тем самым большому количеству пользователей стало известно об уловках, чтобы обмануть систему.

11.2. Community of superuser practice

A group of superusers could develop into a community of superuser practice if they engage in activities where they meet and exchange experience specifically on their superuser activities and role. The emergence of a community of superuser practice was deemed necessary for an enterprise information system to be adopted by its users, due to the distance in practices and purposes between the IT and the user communities (Volkoff et al., 2004). Almnes (2001) conducted a study of superusers amongst nursing home personnel, and McNeive (2009) reports from nurse superusers in a hospital. Both emphasize that belonging to a group is important for superusers, since their role is the only one of its kind amongst those whom they meet daily. In addition to group meetings, e-mail lists, newsgroups and lists of frequently asked questions may be advantageous. The organised group should also provide the necessary opportunities for the superusers to update their skills, whether new software or other upgrades necessitates it. An accounting company formalised their superusers into a group with a coordinator in charge (Åsand and Mørch, 2006). Mozhdeh and Sigrun belong to such groups.

Selected superusers who were going to help out during implementation of a clinical information system, spent on the average 13 hours per week for preparing themselves (Halbesleben et al., 2009). The longer time the superusers spent on learning the computer system, the more positive attitude did the users develop towards the system (Halbesleben et al., 2009).

An organised group has to cross the organisational units. In the Finnish company, the department was divided into three product lines, which again was split into a total of 11 groups (Sykes et al., 2009). A lot of the help was given across groups and also across the product lines.

The superuser is the first person in the support chain. She should handle most of the normal requests dealing with use of the computer system, for which she has received special training. In addition, the superuser should be able to take care of user requests concerning the operating system and standard tools. Both Mozhdeh and Oksana help out on information issues like getting the information into the right field and on IT issues like updating software.

Communicating frequently with users, the superusers receive requests for changes of computer systems. They are in a good position to communicate these requests to the computer department or those in charge of the



I am Sigrun and I have a master degree in IT. Currently, I am the leader of an administrative group of seven and also the superuser for the web publishing system WePublic for the whole department. A majority of the 250 department staff use WePublic. The most frequent questions concern operations that are blocked to the common users. They also often ask about how to reuse past information for this year's schedule.

All departments have a WePublic superuser, but we never meet. However, we e-mail each other and solve many problems in that way. På den måten opprettholder vi en gruppe av Webpublic spesialister.

software and hardware. This aspect of their role should be exploited, such that the requests from the users are taken into account. The meeting of superusers could also be an agenda for discussing and distilling such requests. This is a regular item on the agenda in Mozhddeh's group.

The superuser should be given responsibility of the resources necessary for carrying out the role (Almnes, 2001, McNeive, 2009). A dedicated amount of time for the superuser activities is recommended (Almnes, 2001, Åsand and Mørch, 2006). The resources for sending users for training, is a responsibility that should be attributed to the superuser.

The selection of people for the superuser role is an important issue for creating a decent support system. Superusers' ability to connect socially with other people is in general improving the chances that others will learn from them (van Wijk et al., 2008). People with poor social skills should therefore be avoided. They could preferably be amongst those whom people often calls for assistance, which would guarantee a caring person. In a survey of users and superusers, Boffa and Pawola (2006) warn against selecting users who are indifferent about or have negative attitudes towards the information system as superusers. Halbesleben et.al. (2009) confirm that a positive attitude amongst superusers spill over to the users. People who are unwilling to take on the role should also be avoided. They may behave hostile or less caring towards their peers, and if so, the users will soon stop consulting them.

Almnes (2001) warns against local managers, since they are often too busy and not always available. In addition, many people do not like to expose their misconceptions to their superior. Sigrun is such a boss-and-superuser, which is not recommended, but she mainly helps out people outside her subordinates.

Since superusers help others solving IT problems and also guide them in problem solving, problem solving competence is important for superusers, as stated for the selection of superusers during implementation of a clinical information system (Halbesleben et al., 2009). In addition to computer skills, the superuser also ought to have skills in guiding others, something, which the trainer needs in particular. (Poe et al., 2011) emphasized that the superusers also should have some teaching competence, placing them also as peripheral member of a community of teaching practice.

Some IT personnel like Sigrun change their career into other occupations, and they will naturally be more skilled in IT than their peers. If they have the necessary inter-personal skills, they would become very good at supporting colleagues as well as communicating with the IT specialists. People having at least one formal computer course were more likely to become superusers in a higher education institution (Eveland et al., 1994).

11.3. Superusers' roles

Superusers may provide a variety of tasks, as recognised in guidelines for user support (ITIL - Axelos, 2011). The roles of superusers can be divided into those relating to other users and roles concerning their interaction with IT specialists, see Figure 97. The chauffeur role is taking on other users' primary work (Culnan, 1983) and helping other users (Spitler, 2005,

Halbesleben et al., 2009) means either solving their problem or mentoring the users on solving it themselves. These roles are carried out on the requests of other users who influence superusers' actions, hence these roles are placed near the head of the arrow from users to superusers in Figure 97. Superusers also undertake the role of training other users (Volkoff et al., 2004, Gallagher and Gallagher, 2012) and champion changes (Poe et al., 2011, Stuart et al., 2009). The trainer and champion roles are initiated by the superuser aiming at influencing the users.

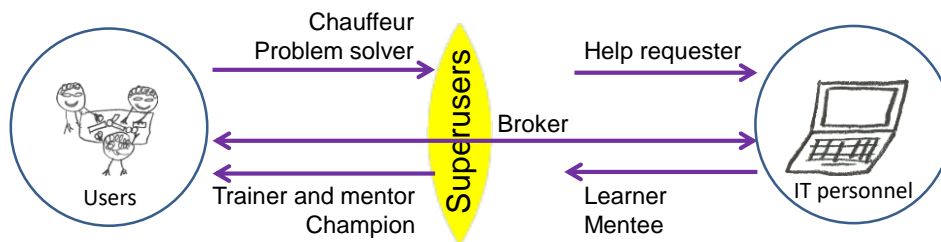


Figure 97. Superusers' roles related to other users and IT specialists. Arrows denote direction of requests (top) and influence (bottom).

Superusers also liaise with IT specialists for support, requesting IT changes (Mackay, 1990) or participating in developing IT solutions, thus learning to become suited for training and helping other users (Volkoff et al., 2004). They request help from and are being mentored by IT specialists. In addition, they mediate news about IT from the specialists to the other users.

Chauffeur

Some people interested in output from information systems do not search themselves, but get others to do it, and this role has been termed *chauffeur* (Culnan, 1983). When Oksana generates a financial report to employees who do not have access to the accounting system, she is their chauffeur.

While Oksana will continue generating the report for other staff due to her access rights, the chauffeur role also appears for other reasons. In a study of implementation of a companywide information system, adoption was slow (Boudreau and Robey, 2005). It was found that most user groups did not attend the initial training programme, and when the software was implemented, the users found ways of avoiding using it. Rather than entering data, they got some groups accountants to be their chauffeurs by carrying out their data entry.

Problem solvers

Later in the implementation reported in (Boudreau and Robey, 2005), some self-initiated superusers found out how to operate the new software, and this competence was spread in the organisation. In the end, most people used the system, after the user communities had found workarounds (see Section 5.8, p.88) in order to get the system performing as needed. This competence was also spread throughout the relevant user communities. Being a broker between business and technology, superusers would be in the perfect position for fitting the two through problem solving.

An e-mail support system had been set up in a global company years before the www was introduced. It was found that users' problems were solved if the helper was either from the IT department or from users who considered themselves more competent on the problem than average, who came from another country and who was not a manager (Constant et al., 1996). There is no obvious reason why an international response is better than a domestic one. Diversity in responses in general seems to help, and the global community may contribute in that respect.

In a management consultancy company (Spitler, 2005), Goran was a superuser on most IT, but he did not know a statistical software package needed for some analysis. He contacted a superuser on the package, who had already set up an analysis model. Goran then acquired the problem solving competence of customizing (see p.86) this package through learning how to write scripts. He thereafter customized the analysis to the particular user's needs.

Broker

Being a peripheral member of a community of IT practice and also another community of practice makes the superuser a broker between the communities, enabling communication between the two communities. Mozhdeh participated in a group which requested changes to the IT department every quarter.

In a study of distribution of software set ups in a computer company, Mackay (1990) observed that for each group in the organisation, one person took the role of a broker between IT and other staff. This person was not an IT professional and also not amongst those with low IT competence. Those who played this broker role of superusers volunteered, such that there was no need for formal organisation, which was established in Mozhdeh's case.

Trainer

In a community of practice, the practice would constitute the tasks of the majority, while the minority would be peripheral people who could learn the tasks through interacting with the majority. A training session is of an opposite kind, where the majority of learners is supposed to adapt to the minority of trainers. Unless all trainers are superusers, it is highly unlikely that the trainers and the trainees develop a common knowledge base during a short in-service training session; hence, such activities constitute boundary interactions rather than CoPs.

In-service training is acknowledged by Wenger (1998) as useful when providing a place for reflection on the practice, and as an opportunity for getting to know people whom one would otherwise not meet. However, Wenger remarks that often in-service training or education are too detached from practice to foster learning strengthening the individuals' participation in the communities, an issue which was considered as transfer in Chapter 7. This could easily happen when the business is not included in IT user training, or when the business included does not match the learners' experience, see the need for a realistic training environment in Section 8.3, p 124. Including superusers amongst the trainers could make training more realistic. Superusers who are also ordinary users could bring their understanding of IT in their own tasks and the business into training and provide familiar examples. Superusers who also

belong to the community of teaching practice could teach ways for further learning, like problem solving competence.

Leaving the training to IT personnel only creates the risk of restricted interaction between users and trainers in the classroom. Including a superuser also creates variety amongst the trainers. The latter is in general an advantage for learning. Both Almnes (2001) and McNeive (2009) recommend that superusers should be involved in planning and conducting IT user courses, in order to include user activities in the training contents. Also, users feel more comfortable by being taught by one from their own profession than by a computer specialist.

Oksana lectured and guided other users on the human resource system. Her experience enabled her to convey how the system should be used to support the accounting tasks. She could also bring her inventions to the larger audience.

Champion

The extent of people's use has been employed as a measure of success of introducing an IT system in an organisation (Davis, 1989, Venkatesh et al., 2003). Perceived usefulness, perceived ease of use, social pressure and facilitating conditions have been found to influence the amount of use (Venkatesh et al., 2003), see Section 3.1. Documentation (Chapter 2 and 4-6), training (Chapter 7) and IT support (Chapter 12) constitute aspects of the facilitating conditions, while superusers are members of the community of user practice, hence they can exert pressure on system use amongst its members. In order to know whether to put the effort into training, IT-support or superusers, knowing the relative influence of each of these factors would be needed. No comprehensive studies of all these factors have been carried out, but a comparison or co-workers' influence versus training provides some insight.

In a non-profit organisation in the US, half of the 200 employees responded to a survey on IT use, perceived usefulness, perceived quality of user training, amount of user training, and co-workers' IT use (Gallivan et al., 2005). 80% of the respondents were female, and the large majority had a university degree. The extent of co-workers' IT use had the strongest impact on an individual's use. Co-workers' perception of the training quality and to a smaller degree the individual's perception of training quality also influenced the extent of the individual's use. The amount of user training and the perceived usefulness had no influence. The latter contradicts the technology acceptance model (Venkatesh et al., 2003), and there is no obvious explanation for this finding.

The study points to the importance of what happens in the local work group and the possible futility of putting many resources into training (Gallivan et al., 2005). Given that people imitate colleagues in their computer use, and that a new system is to be introduced, people will only use it if their co-workers do. For an innovation to be taken up, some have to lead, such that the rest of the community can follow suit. Superusers are in a favourable position to be the leaders, since they are well versed in IT in addition to being a colleague of the others. In order to become a strong leader, superusers would need to be well trained and preferably a member of a community of superusers, such that they also can learn from each other.

Consequentially, providing thorough training for superusers could be more effective than training the whole group of users.

The study also points to that the quality of the training is more important than the quantity (Gallivan et al., 2005). Since being able to help others would probably ease the leadership role of the superusers, their training should particularly emphasize understanding and problem solving.

IT innovations in organisations are often driven by champions, being people who persistently and convincingly argue for changes (Beath, 1991). McNeive (2009) and Poe et.al (2011) emphasize that superusers should be champions for the changes that the computer system should support. Champions who get support from the IT department are more likely to succeed.

Innovation champions in general have a breadth of interest, view their role flexibly and believe that they can influence events (Howell, 2005). They employ official and informal channels to persuade colleagues and are able to frame new ideas as opportunities specifically targeted at potential users. For instance, a successful champion will demonstrate a prototype of a system to case file handlers, pinpointing how the new system could ease organisation and access to their data. The champion would show the manager how the same system would improve the overview of the company's performance, and the IT department would be told how the system could relieve the staff from previous maintenance trouble. Selling an innovation means making people understand how the system improves their business and provides a useful tool for their work. Managers can help out by letting users with particular interests volunteer to champion systems and by recognising achievements by the champion (Howell, 2005). Recognition could entail assigning the champion to a new, exciting project, to make them cooperate with leaders who appreciate their style of working or with other champions, or offer them educational opportunities.

A study of Canadian managers' intentions to champion IT found that their knowledge of the applications and their access to other people with IT competence were particularly favourable for their championing role (Bassellier et al., 2003). In a study in New Zealand, school managers' IT competence, primarily developed through use, was the most important factor for their intention to champion IT in their school (Stuart et al., 2009). The same conclusion was drawn from a similar study of school managers in Iran (Afshari et al., 2012). If the same holds for superusers, those with higher level of IT use competence and those with stronger connections to IT personnel will be better champions.

Users who hear negative (positive) remarks from peers about a system develop negative (positive) attitudes towards it, and subsequently negative (positive) motivation to use it (Galletta et al., 1995). Superusers should therefore distribute positive remarks about a system to be championed. Unfortunately, negative remarks in general have stronger effects towards people not adopting an innovation compared to the effect that positive remarks have in the other direction. To champion a system, superusers therefore have to try to silence those against it and be prepared to counter their remarks and arguments.

11.4. Organising for competence development

The accounting company mentioned at p.148 appointed one superuser per 10 employees, and gave the superusers the obligation of training the others (Åsand and Mørch, 2006). Being organised in a community of superusers helped them becoming capable of carrying out this task. The same proportion was also utilized when introducing a patient record system for nurses (McIntire and Clark, 2009), while in another hospital, there was one superuser per 15 nurses (Poe et al., 2011). In contrast, the superusers Oksana and Sigrun support 250 users. The reason is, the information systems for which they are superusers are only used now and then by most of the staff.

The Finish company trained all 100 users simultaneously (Sykes et al., 2009). The study revealed that there was a positive correlation between how often a user gave or received help and how much she or he used the system. If the goal is high system use, helping each other after training is therefore effective. Seen from the individual user point of view, it would constitute a facilitating condition in the technology acceptance model (Venkatesh et al., 2003), see Figure 12, p.28.

When a new system has a large number of users, training is costly and can lead to disruption of the organisational performance. The latter is unacceptable when clients have to be cared for, like in a hospital, or when processes cannot be halted, like in a power plant. To reduce costs and avoid disruptions, training is often provided only to a group of superusers, who are selected from each organisational unit. All staff is given access to user documentation, and the superusers are thereafter supposed to help the rest of the staff when needed.

11.5. Summary

Users who help out colleagues on IT related issues are called superusers. People may develop informally into superusers because others seek their assistance, or they may be appointed by the organisation, and given advanced training. Their background as regular staff and their additional IT competence make them experts on the fit of IT in their part of the business.

Appointing superusers has become a common strategy for introducing information systems in large organisations. Selected users from each department are appointed superusers with the responsibility of assisting the other users and functioning as a liaison between their users and the IT services.

Superusers need time and authorities to carry out their tasks. They learn more about the IT through regular discussions with each other. Engaged superusers can convince people to start using a system and keep them afloat by encouragement and support.

8. Identify, organise, authorise and cultivate superusers.

9. Include superusers as trainers and champions for new IT systems.

Chapter 12. IT support

The learning objective of this chapter is to be able to organise user support such that IT support personnel help users increase their competence.

Learning can take place anywhere and anytime, but some activities are carried out with learning as their main purpose. In addition to training, supporting users also constitutes an activity where learning may be the main objective.

12.1. How IT supporters learn

Support is normally a boundary interaction between an IT specialist or a superuser on the supporting side and a user at the receiving end. The IT is a boundary object in the interaction, and documentation and data may constitute other boundary objects.

Support interactions are normally of limited duration, being a few minutes conversation or a couple of written messages. Contrary to training sessions, the topic of the support sessions are initiated by the users and the IT support is targeting the user's current problem. Support personnel would normally not prepare specifically for an encounter, but they may subsequently note down information about it.

While superusers have the advantage of knowing the business, IT personnel would constitute the expertise for IT problems. Also, staff in an IT department in a larger organisation would normally have user support as a main part of their job, while helping others constitutes an additional role for superusers.

When users and IT personnel meet, they talk about the same phenomena in different ways. For example, when a user says

we have a group of students who cannot synchronize

the technician talks about

IP-errors or server-errors (Kanstrup and Bertelsen, 2006)

We see the terminology problem from search in documentation (Section 2.1) reappearing. When the user and supporter are co-located, they also have boundary objects like software and documentation, which they can look at, point to and interact with, and they can observe each other's actions. When helping on the phone, the oral interaction is the only communication channel. The following conversation took place when a user of a printer/copier called the vendor's support centre for help. The support person searches a knowledge base for finding possible solutions (Crabtree et al., 2006):

Troubleshooter: OK, and what's the problem you're having with the machine?

Customer: I'm getting poor quality prints – sort of smudges on them.

Troubleshooter accesses knowledge base and selects 'image quality'.

Troubleshooter: When it's printing?

Customer: Yes.

Troubleshooter: OK, do you get this when it's copying?

Troubleshooter: So you get it printing and copying and they're like smudges?

Troubleshooter selects 'smears and smudges' in knowledge base.

The troubleshooter questions the user to come up with a more precise description of what had happened and what the result looked like, guiding the user in the precise observation (see the problem solving approach at p.73). The troubleshooter might have gained improved skills in learning the user's terminology through this interaction, see the skill learning process to the left of Figure 98.

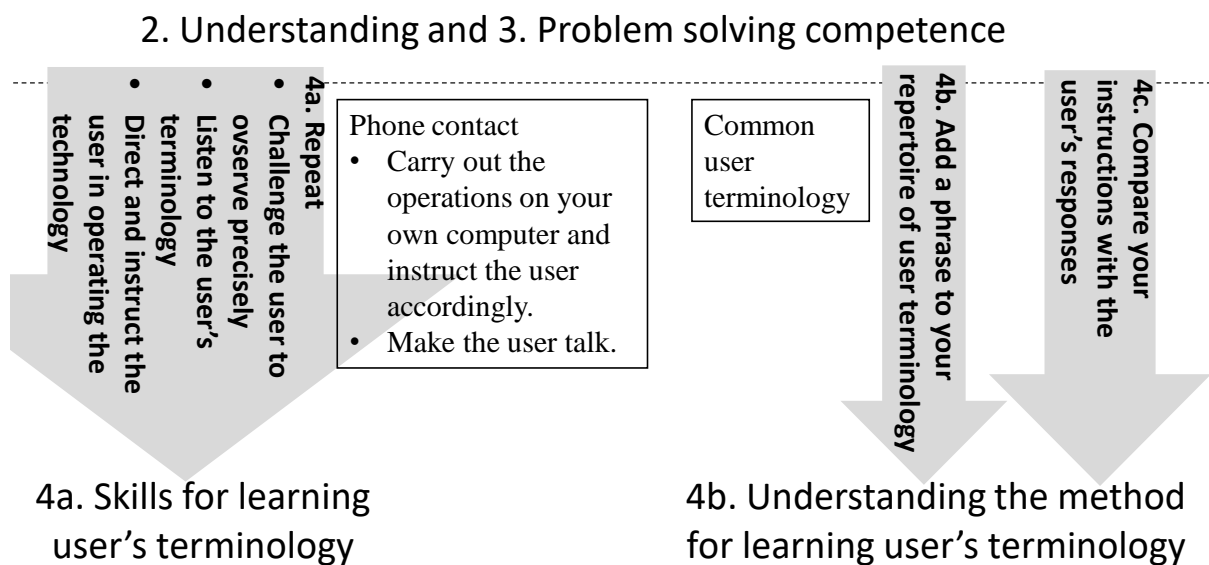


Figure 98. The learning processes of an IT supporter who tries to understand users' terminology.

Likewise, the customer has to grasp the technical terms 'image counts,' 'xerographic' and 'fuser module' in the following (Crabtree et al., 2006);

Troubleshooter: You know your image counts, which is the amount in thousands of copies that the xerographic and fuser module have done, check them just to see if they're running over their copy limit and causing that problem for you.

Troubleshooter: Of course, yeah, take your time, that's fine.

48 second pause.

Customer: Where do I find them?

Troubleshooter describes how to use the menus to find the counts and customer goes to find them.

70 second pause.

Customer: 43

Troubleshooter: Hi, that's from your fuser module

(writes down count).

Troubleshooter: OK could you - do you know where the xerographic module is in the machine?

Troubleshooter: OK, I'll tell you exactly where it is as there's something I want you to try, just to see if this will rectify the problem for you – if you open the front door of the machine ...

Here we see that the troubleshooter also instructed the user through some operations and observed the user's language at the same time. The troubleshooter may then learn how the instructions are interpreted by the user, thus improving the skill for learning user terminology and also possibly come to an understanding of the method for learning user terminology. Accumulating a set of terms expressed by users contribute to improving the supporter's competence in this area, see the right arrow in Figure 98.

If the customer remembers the steps such that he can do them without support the next time the problem occurs, he has learnt the particular research cycles for troubleshooting the machine.

The troubleshooter could more easily recognise user terminology if having a rich repertoire of how users expressed themselves. Correspondingly, an IT support person who is acquainted with common misunderstanding may more easily detect these and clarify for the user. Consider Herbert, who had not understood the difference between closing a program and minimising it (Section 4.10, p.62). A support person may provide an explanation of this difference, such that the user achieves a more adequate understanding.

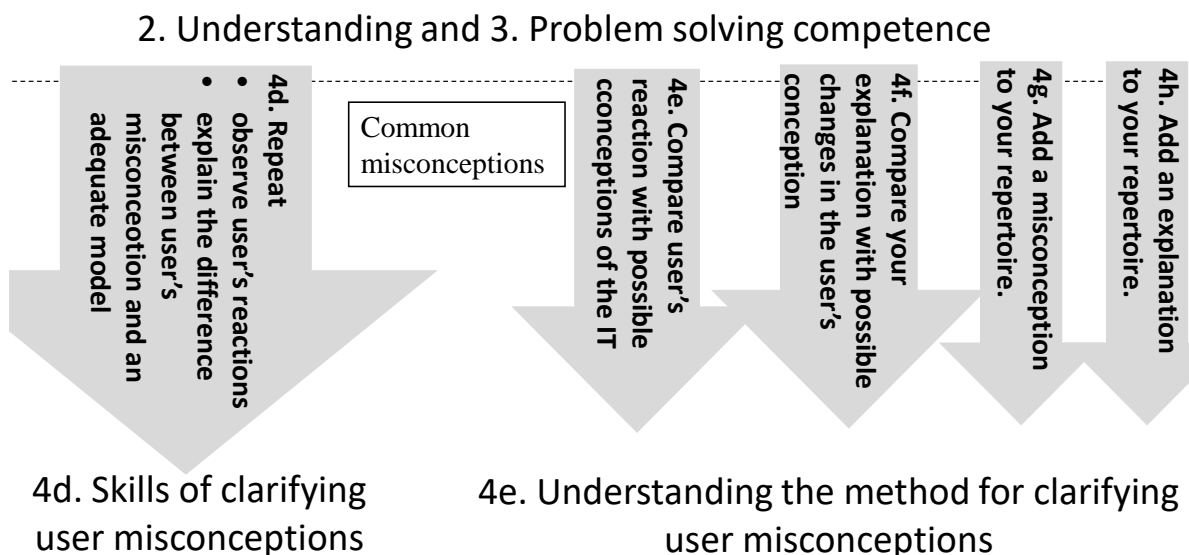


Figure 99. The learning processes of an IT supporter who tries to clarify users' misconceptions.

IT supporters do observe users' reactions and diagnose possible misconceptions (Allen et al., 2013). The support person may develop skills in clarifying misconceptions through observing users' reactions to the explanations, which the support person provides, see the left arrow in

Figure 99. Comparing user reactions with possible conceptions of the IT and comparing explanations provided with possible changes in the user conceptions improve the support person's understanding of what it takes to clarify misconceptions, see the arrows to the right in Figure 99.

While the cases reported here concerns IT support staff learning, superusers and trainers may learn about the users whom they guide in the same way. Since support staff interacts with users constantly, while superusers may help colleagues less frequently, superusers may not become equally skilled in learning others' terminology and clarifying misconceptions as support personnel.

Support persons are often technical experts, and experts in general overestimate the competence of those whom they support (Hinds, 1999, Nathan and Koedinger, 2000). The troubleshooter in the case above communicated interactively with the user, enabling the troubleshooter to find out about the users' competence level. This is more difficult if the support person only reads a short, written message. Supporters who know the competence level of the user will adapt their responses by translating their jargon into less technical terms, like the troubleshooter did, and include only explanations which the supporter thinks are at the appropriate level for the user to understand (Nückles et al., 2006). Efficient supporters interact with a variety of users and experience a multitude of user problems and terminology, and they are 2-10 times as efficient as supporters with more homogenous interactions (Chi and Deng, 2011). Their effectiveness may be due to having collected rich repertoires of user terminology, misconceptions and explanations.

In addition to learning about users, IT supporters also learn IT and business fit topics during their interaction with users. The interaction between 11 IT supporters and 61 users was observed during implementation of a work flow system in a US bank (Santhanam et al., 2007). In addition to support, several meetings were organised for the users and IT personnel to jointly discuss issues. It was found that users mainly learnt IT skills during interaction. The IT personnel gained know-why, i.e., they understood how the IT worked in users' business during the same encounters. Also understanding of the technology was shared. User competence, particularly on how the IT is used in business, is hence introduced into the community of IT practice and shared amongst the IT personnel (Santhanam et al., 2007).

12.2. Support quality

In a qualitative study, 39 users in the Finnish public and private sectors were interviewed about their learning preferences (Korpelainen and Kira, 2010). In general, they preferred learning IT use on the spot; formal training courses take too long. Said one of the interviewees:

There are so many [user training] courses and other rubbish that I can't be bothered to do an extra thing. I haven't left a single task uncompleted, so why would I bother. [. . .]. I don't need the extra information, and I am not interested. I am only interested in being able to do my tasks; I just want to find the information and complete my tasks. That is all I need the system for. (Korpelainen and Kira, 2010)

Also, users hardly read documentation (Novick et al., 2007), they rather ask others, unless they try and err or succeed. Getting support is therefore essential for most users both for learning and for solving IT problems without learning how to do it themselves the next time.

A survey of 484 users in a US university examined the correlation between support factors and user satisfaction (Shaw et al., 2002). The factors that influenced satisfaction the most are listed in Table 8. Factor 1, 5 and 6 are all qualities of IT support. Factor 3, user understanding, is partly influenced by previous training and support. Factor 4, software upgrades, is also a product of decisions in IT departments. Many users get annoyed when new upgrades appear, since they have to relearn the software, however, others push for new versions.

Table 8. Factors influencing user satisfaction (Shaw et al., 2002).

Rank	Factor
1	Fast response time from system support staff to remedy problems
2	Data security and privacy
3	User's understanding of the system
4	New software upgrades
5	Positive attitude of information systems staff to users
6	A high degree of technical competence of systems support staff

In general, the findings point to the central position of user support and to learning issues for user satisfaction with IT. Software upgrades and response time were the only factors found to correspond with studies from the beginning of the 90's.

When broken down into three distinct user groups, administration, academics and students, there was a great variation in the factors. A previous study also found variations, concluding that the specific business of a user department influences its perception of IT support (Speier and Brown, 1997).

Users' opinion of the performance of the IT support gave the lowest score to documentation which supports training (Shaw et al., 2002). This issue is not amongst the general top factors influencing satisfaction, since users regarded it as less important. However, non-academic users had this item in their top factors of dissatisfaction (Shaw et al., 2002). Low quality of training material is particularly bad since Minimal Manuals and models for understanding used during training are twice as effective for problem solving compared to material found elsewhere (Novick et al., 2009).

The physical place where users find IT support personnel has been called a *helpdesk*, while the phone call support is called a *helpline*. A survey of user satisfaction with helpdesks and helplines in the Netherlands compared user preferences (van Velsen et al., 2007). There were 64 responses concerning the helpdesk and 242 for the helpline (11% response rate). Concerning the helpline, user satisfaction depended to a high degree on the quality of the solution which the support personnel came up with. Surprisingly, the users who contacted the helpdesk were more satisfied when having a good time at the helpdesk, while the solution was of secondary importance. Thus, the helpdesk should have friendly staff, while the knowledgeable ones should be working on the helpline.

12.3. Improving IT support

An IT department will normally be responsible for parts of the IT support for the whole organisation. IT departments previously often conceived their tasks as management of the technical installations and organised their activities according to the hardware and software packages. Consequently, there was one support service for each software package; hence a user who did not know whether his problem was connectivity, web-browser or credentials might have had to approach three support people to get a useful response.

Lately, a view of themselves as providing service under the heading IT Service Management has become more common amongst IT departments (Iden and Eikebrokk, 2013). This change has been influenced by a set of guidelines for IT service operations called ITIL – Information Technology Infrastructure Library. ITIL has been developed by the British Office of Government Commerce. The guidelines include strategy, design, transition and operation of services. ITIL Service Operation (ITIL - Axelos, 2011) concerns user support and handling of incidents. It suggests organising one *service desk* as a single point of contact to handle all user contact, including troubleshooting, requests for software changes and warnings concerning threats to data security. The service desk also informs users about planned downtimes and equipment which has failed, and it monitors networks and servers, such that it often knows the issues before users contact them. Organisations which have implemented ITIL or similar guidelines have improved their user satisfaction and service quality, including reduced response time (Iden and Eikebrokk, 2013).

ITIL succeeded more in larger than smaller firms in a study in Malaysia (Kanapathy and Khan, 2012). In an international survey of IT managers, the benefits to the operation of the IT support increased as the maturity of adoption of ITIL grew (Marrone and Kolbe, 2010)

The organisation of user support at Digibank is a common way of splitting up the services. Hayley talks about the Ticket system, which is an issue tracking system, where each user request is registered. This system enables communication between the support sections. If the first line of support cannot handle the request, the issue tracking system sends a message to the second line for them to pick it up.

An issue tracking system also allows statistical analysis of the tickets, such that Hayley can see

Digibank – user support manager Hayley – 1:

We have organised user support in three sections. The Customer Desk is the service point for all of our hundred thousand customers. Therefore, customers can call and e-mail us 24/7, and our average phone response time is 30 seconds.

For our staff, we run the Front Desk within working hours. After we installed the Ticket system there also, user satisfaction has improved. Previously, some requests got lost and remained unanswered.

Back Office is our second line of support. If Customer or Front desk cannot solve the problem, it is referred to the Back office through the Ticket system. If they can't deal with it, they may call our IT vendors.

the common issues. She would then bring the common issues to the attention of IT personnel dealing with software modification, such that they might prevent the trouble from reappearing through altering the systems.

Support personnel should have a wide range of capabilities. An experienced support manager summarised the characteristics of an ideal support person like this (Bruton, 2002):

- Patience
- Assertiveness
- Thoroughness
- Enthusiasm
- Responsibility
- Technical knowledge
- Empathy
- Communicative ability
- Works well under pressure

This list contains a mix of technical and personal qualities which is hard to find amongst job applicants, and those who possess all of these are unlikely to be satisfied with a technical support job, which is often low paid. Hayley is well aware of this issue and has a solution. A large company like hers can take advantage of people's strong and weak sides by assigning them to a job where their competence is most needed.

Digibank – user support manager Hayley – 2:

For the Customer Desk, we recruit young, service-oriented people, since it is easier to provide them with some technical skills than to convert an introvert computer nerd into a social person. We teach the recruits the solutions to the 10 most frequently asked questions, which constitute 85% of the calls. Then we emphasize that other questions be forwarded to the Back Office.

And that's where we hire the nerds! We love their efficiency and perseverance when solving the intricate issues.

Novice support persons would benefit from being trained up to the problem solving level for the most frequently occurring problems. ITIL suggests that they thereafter listen in to experienced staff before they start responding to calls and messages themselves under supervision of a mentor (ITIL - Axelos, 2011). Having a possible career path into technical, training or managerial positions may keep valuable service staff in the organisation.

12.4. IT support versus superusers

In contrast to the single point of support by a service desk, a superuser may provide support for specific software, which is also recognised in ITIL (ITIL - Axelos, 2011). There have been several studies on the type of support that

users prefer, and users' preferences seem to depend on many factors.

Interviews of 40 users with education above high school in the US, showed that users preferred asking IT-personnel and colleagues at roughly the same rate (Novick et al., 2007).

In a survey amongst university staff in Norway, 49% preferred support from the IT services, while 31% chose colleagues (Nilsen and Sein, 2004). There were 222 responses to the questionnaire (37% response rate).

A survey of US middle level managers' opinion on support gave the opposite result. 38% preferred superuser support, 26% other colleagues and 19% an IT centre (Govindarajulu et al., 2000). These results are based on 98 informants (response rate 11%).

These three studies agree that users prefer support from both IT personnel and superusers, but there is no consensus on which one is superior. A survey in Norwegian organisations investigated some possible causes for choice of support (Munkvold, 2003). Responses came from 277 informants, yielding a response rate of 41%. Short distance to the IT support personnel made users go there, while when the distance was longer, users preferred colleagues. Users with higher IT competence consulted the IT support to a lesser degree, while they solved problems more often themselves than did less skilled users (Munkvold, 2003). Convenience and the users' competence level may therefore be reasons for choosing one source of support to the other.

12.5. Summary

IT support is an organised service for users who need help for solving their problems. Users may learn IT during support encounters and IT support personnel may learn business fit from the users. Support personnel learn about users' IT competence through a diversity of interactions, thus becoming increasingly able to adjust their technical explanations to the user's level. Support staff complements superusers, who are the business fit experts.

While users want support to help them understanding more, quick response time is the most important quality of support. Large organisations have improved their user support through a service desk, which is a single point of contact for all users. The service desk is staffed with service-minded people, who forward more tricky questions to computer scientists in the back office.

10. Organise one service desk for all user requests with service minded staff.

Chapter 13. Mutual learning during business fit

The learning objective of this chapter is to be able to organise collaboration between users and IT personnel during development of IT solutions such that these two groups learn from each other.

There have been numerous cases where IT designers have developed systems, which do not fit users' business and which therefore have not been used. When developing an information system or any digital device, the IT personnel involved in the design hence need to know the business into which the software and hardware are going to fit. While IT personnel have the IT competence, users would know the business fit. In order to bring all competence areas into development, users and IT personnel are often cooperating in what has been termed participatory design.

While IT personnel's need for understanding users' business was recognised early, it was also noted that for participatory design to become effective, users also needed to learn about IT, so the term *mutual learning* was adopted (Bjerknes and Bratteteig, 1987).

Based on the identification of roles of users and IT personnel, this chapter will characterize the competence of these roles during development of IT applications in organisations.

The competence needed for a development group consisting of both users and IT personnel was suggested by Kensing and Munk-Madsen (1993) to consist of six categories as shown in Table 9. This categorisation of competence in mutual learning has been repeated until lately (Bødker et al., 2004, Bratteteig et al., 2013).

Table 9. Areas of competence in user-developer communication. Based on (Kensing and Munk-Madsen, 1993).

	Users' present work	New system	Technological options
Understanding	Relevant structures on users' present work	Visions and design proposals	Overview of technological options
Skill	Concrete experience with users' present work	Concrete experience with the new system	Concrete experience with technological options

Before starting cooperation, users would have the concrete experience with their own work and not necessarily more (Kensing and Munk-Madsen, 1993). This implies that if they used computers, they would have the skills, but not necessarily understanding of the technology or the business fit. IT personnel would start out with the technological skills and understanding.

In order to reach joint understanding of a new system, the two partners first need to understand parts of the current situation. Mutual learning can be considered as consisting of three processes.

1. Users learning about IT
2. IT personnel learning about the business fit.
3. Based on such a mutual understanding of each other's competence areas, they jointly create understanding and skills of the new IT and business fit.

Users normally do not constitute one homogenous group. Rather, they do a variety of jobs in different departments and levels in the organisation. When building new systems for larger parts of an organisation, a diverse group of users would be needed to bring in all the organisation's skills in using IT for the different activities (Markus and Mao, 2004). Including some existing superusers would have the advantage of bringing competence of the relation between IT and business into the project team.

13.1. Users learning about IT

This is what the two first parts of this book is about. The special requirement here is that users need to learn about technological options which can be utilized in their organisation, but which currently are not in place. Implementations of these options in other applications can be used for practical training, and functional and structural models can create some understanding.

The two other learning processes differ from those having been presented previously in this book, so they will be elaborated in the sequel.

13.2. IT personnel learning about business fit

As seen in the previous chapter, IT personnel learn about business fit during encounters with users (Santhanam et al., 2007). This can be done in more systematic ways, including interviews and observations. Interviews can provide an understanding of how IT fits in the organisation in a broader sense than for one user, thereby contributing to understanding the current situation. Observations can complement interviews through providing understanding of how individuals use IT for their tasks.

For users lacking the understanding of IT in the business, the interviews should be as concrete as possible, making the user telling what she actually does. Carrying out the interview at the workplace enables talking about the IT being present, thus details of work can be included.

Other users may have more elaborate understanding of their work than anticipated. In a study of a lawyers' office, the interviewer noticed that they frequently discussed the meaning of codes and that they

... were continuously experimenting with alternative strategies for coding documents. One lesson we (re)learned was the degree to which workers themselves are engaged in reflecting on and redesigning their own practice. (Blomberg et al., 1996)

They concluded that for learning about users' business, the investigator should search for the knowledge work in what is called routines, and the routines in what is called highly qualified work.

Interviews and observations need to be thorough enough to avoid generalisation based on single cases. Anthropologists experienced in studying work in organisations emphasize that investigations should include

comparing observations of the same individual over time and in different settings; comparing interview and observational data from the same individual, investigating apparent disparities between them; and comparing what people say about each other with what they can be seen to do, again using apparent disparities to guide further investigation. (Forsythe, 1999)

Aiming at learning the diversity of users' business, investigations should ideally continue as long as one learns something new. In reality, there will be budget and deadlines, which limit the amount of interviews and observations which IT personnel can carry out.

13.3. Joint creation of understanding and skills of new system

After users and IT personnel have learnt about each other's area of competence, they would have a platform for joint creation of new systems. Users and developers exploring new options together and comparing with wanted changes may generate skills and understanding of a new system. Also, the users and developers acquire skills for mutual learning, see the left arrow in Figure 100. Understanding mutual learning can be achieved through comparing this experience with suggested techniques for mutual learning, three of which will be presented in the sequel.

2. Understanding current use of IT and technological options

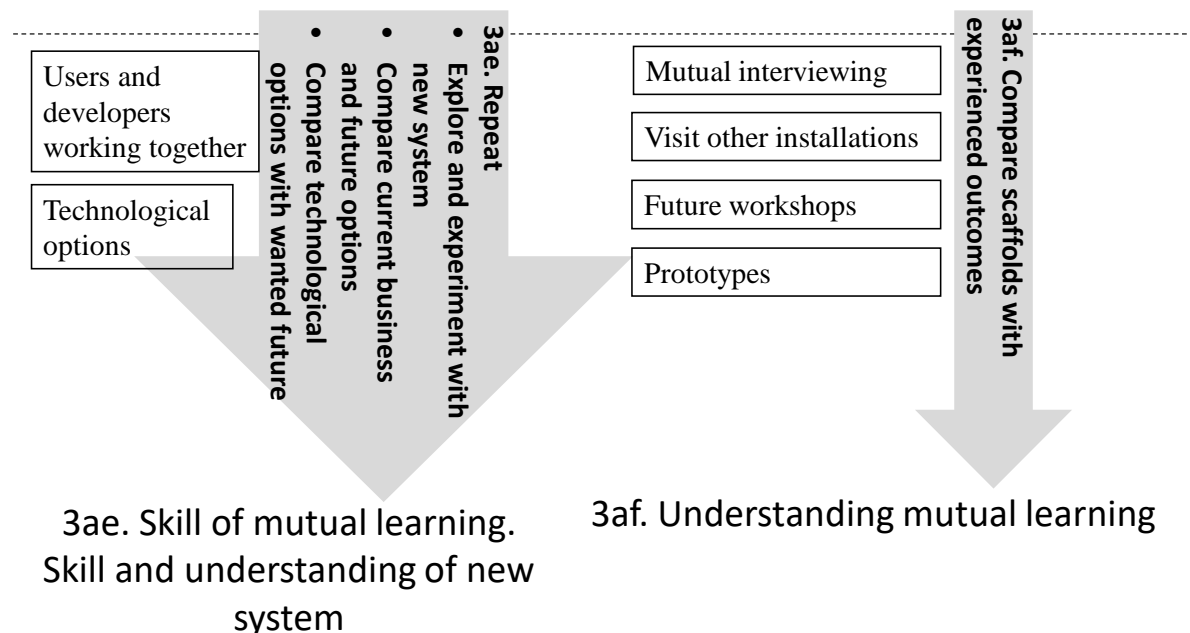


Figure 100. Learning mutual learning.

User participation during development of information systems has in general been found to contribute to better solutions and less user resistance during implementation, although that is no automatic consequence (Bano and Zowghi, 2013). One way of obtaining the positive outputs has been identified as engagement or involvement, meaning the importance and

personal relevance of a system to a user (Hartwick and Barki, 1994). Involvement in participation can be strengthened by allocating responsibilities for some tasks to users. It has been noticed that one reason why users do not involve themselves properly during development is scarcity of time. Rather, they become engaged after implementation, when changes are more costly (Wagner and Piccoli, 2007). Either, management can allocate sufficient time and promote engagement during development, or more of the mutual learning can take place during support, as illustrated in the previous chapter.

Many ways of mutually creating new visions and plans have been proposed, and comprehensive methods are found in the literature on user participation (Bødker et al., 2004, Greenbaum and Kyng, 1991, Simonsen and Robertson, 2013). We will point to four techniques with learning outcomes in different areas, and these techniques can strengthen engagement.

Mutual interviewing

Those who ask the questions set the agenda and introduce the concepts for how to perceive the world. An IT person interviewing users will thus be given power to interpret the situation (Bråten, 1973). In order to create communication on more equal terms, users could also interview IT personnel on their work. Such mutual interviewing has been carried out (Nielsen et al., 2003) and contributed to a dialogue fostering mutual learning.

Visiting other installations

Users and IT personnel jointly visit another organisation, which has implemented a similar system. When representatives from the other organisation demonstrate and explain the function and structure of the system, the visitors can acquire an understanding of its technology and business fit. When trying the system themselves, they can obtain some skills.

When prospective users discuss with people in corresponding positions in other organisations, the users may identify with these people, something which can trigger engagement. Another way of strengthening involvement can be to make every visitor responsible for reporting the experience concerning their speciality to the whole group when returning home.

Future workshops

The aim is primarily to create an understanding of current business fit and of future IT and business fit. Users and IT personnel participate through three phases:

1. Critique session
2. Fantasy phase
3. Realisation phase

During the critique session, the participants create a joint understanding about problems in the current situation. The result of the critique phase could include statements like

We have to log into three different systems. This is annoying, especially since you are logged out if inactive for a while. And I always mix up the passwords.

The fantasy phase is intended for developing ideas and visions for a different future situation through brainstorming technological solutions and work processes. The participants can come up with a list like

The computer will see who I am, such that there is no need for login.

All customers will enter their invoice directly into our accounting system. All we will have to do is controlling.

I wish we had an overview of all heavy-duty machinery on an app, such that we could easily see the closest site when allocating staff to customer visits.

In the realisation phase, plans for implementing proposals that can be realised are worked out. The realisation also contributes to enriching the understanding of new systems such that users can learn the technological options and the IT personnel can obtain insight into the business.

Future workshops do not require any specific professional background. This means that anyone can be allocated responsibility for any of the phases, thus strengthening their engagement.

More comprehensive introductions to future workshops can be found in (Greenbaum and Kyng, 1991) and (Bødker et al., 2004).

Prototypes

While future workshops provide some understanding about possible designs, prototypes provide the concrete experience. A prototype is a technological implementation of aspects of an IT system. It may provide some functionality, show parts of the user interface, demonstrate a new technological option, etc. When working with prototypes, the IT personnel will have the specific role of developing the IT solutions.

When trying out a prototype, experimentation will be a main way of learning for users and information officers. They will see whether it can do what they expect. They may also explore other aspects of the system. Experimentation and exploration contribute to both skills and understanding of the IT.

In order to learn about business fit, the prototype should be tested in the business where it is going to be used or in similar conditions.

Development and testing of prototypes takes place in an iterative fashion. IT personnel learn about the same topics as the users when observing their operation of the prototype and discussing their experience.

Responsibilities can also be distributed during prototyping. Users could for example be required to find their most complicated data structures to see whether the prototype can handle these. Another option is that users can find typical and exceptional cases to be used for testing functionality.

13.4. User representatives as superusers

Users having participated in mutual learning during development will be well suited for joining the training team of a new system and act as superusers (Volkoff et al., 2004).

A study of post-implementation of enterprise resource planning systems, mainly in the US, showed that the companies had included user representatives in their development and implementation projects. After implementation, the majority returned to their original departments and functioned as superusers there (Gallagher and Gallagher, 2012). While the goals of the projects had been business process improvements before the implementation, the success criteria afterwards had shifted to timely response to user needs and user satisfaction. Whether success depended on the transformation of user representatives into superusers was not investigated.

13.5. Summary

User participation in development of information systems is considered advantageous. To achieve fruitful participation, users need to learn about new IT options, and IT personnel need to learn about tasks and the business. In order to design new solutions, users and IT personnel need to develop joint understanding and concrete experience of IT and business fit for future use of new technology. Exploring and experiment with new systems, discussing the current situation and wanted futures, and comparing wanted futures with technological options constitute activities which foster mutual learning. Mutual learning can be strengthened if users become personally engaged, something which requires sufficient time and responsibilities during development.

References

2014. Superuser. *Wikipedia*.
- ABRAMI, P. C., BERNARD, R. M., BOROKHOVSKI, E., WADE, A., SURKES, M. A., TAMIM, R. & ZHANG, D. 2008. Instructional Interventions Affecting Critical Thinking Skills and Dispositions: A Stage 1 Meta-Analysis. *Review of Educational Research*, 78, 1102–1134.
- ADOBE. 2012. *Adobe Captivate 5.5 - Elearning authoring software* [Online]. [Accessed].
- AFSHARI, M., BAKAR, K. A., LUAN, W. S. & SIRAJ, S. 2012. Factors Affecting the Transformational Leadership Role of Principals in Implementing ICT in Schools. *The Turkish Online Journal of Educational Technology*, 11, 164-176.
- AGUINIS, H. & KRAIGER, K. 2009. Benefits of Training and Development for Individuals and Teams, Organizations, and Society. *Annual Review of Psychology*, 60, 451-474.
- ALLEN, J. M., GUGERTY, L., MUTH, E. R. & SCISCO, J. L. 2013. Remote Technical Support Requires Diagnosing the End User (Customer) as well as the Computer. *Human-Computer Interaction*, 28, 442-477.
- ALMNES, T. C. C. 2001. *Superbruker. Hvordan forbedre brukerstøtte of informasjonsflyt*. MSc, University of Oslo.
- ANDRADE, O. D., BEAN, N. & NOVICK, D. G. 2009. The macro-structure of use of help. *SIGDOC '09*. New York: ACM.
- ARTHUR JR., W., BENNETT JR., W., EDENS, P. S. & BELL, S. T. 2003. Effectiveness of Training in Organizations: A Meta-Analysis of Design and Evaluation Features. *Journal of Applied Psychology*, 88, 234–245.
- BABIN, L.-M., TRICOT, A. & MARINÉ, C. 2009. Seeking and providing assistance while learning to use information systems. *Computers & Education*, 53, 1029–1039.
- BAGAYOGO, F. F., LAPOINTE, L. & BASSELLIER, G. 2014. Enhanced Use of IT: A New Perspective on Post-Adoption. *Journal of the Association for Information Systems*, 15, 361-387.
- BALLANTINE, J. A., LARRES, P. M. & OYELERE, P. 2007. Computer usage and the validity of self-assessed computer competence among first-year business students. *Computers & Education*, 49, 976–990.
- BANNON, L. J. 1986. Helping users help each other. In: NORMAN, D. A. & DRAPER, S. W. (eds.) *User centered system design : new perspectives on human-computer interaction*. Hillsdale, NJ: Lawrence Erlbaum.
- BANO, M. & ZOWGHI, D. 2013. User involvement in software development and system success: a systematic literature review. *Proceedings of the 17th International Conference on Evaluation and Assessment in Software Engineering*. New York: ACM.
- BARNES, L. E. 1890. *How to Become Expert in Typewriting: A Complete Instructor Designed Especially for the Remington typewriter*, A.J. Barnes.
- BASSELLIER, G., BENBASAT, I. & REICH, B. H. 2003. The Influence of Business Managers' IT Competence on Championing IT *Information Systems Research*, 14, 317 - 336.
- BEATH, C. M. 1991. Supporting the Information Technology Champion. *MIS Quarterly*, 15, 355-372.
- BEN-ARI, M. & YESHNO, T. 2006. Conceptual models of software artifacts. *Interacting with Computers*, 18, 1336–1350.
- BHAVNANI, S. K. & JOHN, B. E. 2000. The strategic use of complex computer systems. *Human-Computer Interaction* 15, 107-137.

- BHAVNANI, S. K., PECK, F. A. & REIF, F. 2008. Strategy-Based Instruction: Lessons Learned in Teaching the Effective and Efficient Use of Computer Applications. *ACM Transactions on Computer-Human Interaction*, 15, 1-47.
- BILAL, D. 2002. Children's use of the Yahoo!igans! Web search engine. III. Cognitive and physical behaviors on fully self-generated search tasks. *Journal of the American Society for Information Science and Technology*, 53, 1170-1183.
- BJERKNES, G. & BRATTETEIG, T. 1987. Florence in wonderland. In: BJERKNES, G., EHN, P. & KYNG, M. (eds.) *Computers and Democracy—A Scandinavian Challenge*. Aldershot: Avebury.
- BJØRGE, E., JØNSSON, A., KAASBØLL, J. & PINARD, M. 2015. From User Training Courses and Central Support to Creating Local User Competence for Mentoring Colleagues: A Preliminary Study in Malawi. In: CUNNINGHAM, P. & CUNNINGHAM, M. (eds.) *IST-Africa 2015 Conference Proceedings*. IIMC International Information Management Corporation.
- BLOMBERG, J., SUCHMAN, L. & TRIGG, R. H. 1996. Reflections on a Work-Oriented Design Project. *Human-Computer Interaction*, 11, 237-265.
- BLOOM, B., ENGLEHART, M. D., FURST, E. J., HILL, W. H. & KRATHWOHL, D. 1956. *The Taxonomy of Educational Objectives, The Classification of Educational Goals, Handbook I: Cognitive Domain*, New York, David McKay.
- BOFFA, D. P. & PAWOLA, L. M. 2006. Identification and conceptualization of nurse super users. *Journal of Healthcare Information Management*, 20, 60-68.
- BORGMAN, C. L. 1986. The user's mental model of an information retrieval system: an experiment on a prototype online catalog. *International Journal of Man-Machine Studies*, 24, 47-64.
- BRANSFORD, J. 2000. *How people learn: brain, mind, experience, and school*, Washington, D.C., National Academy Press.
- BRATTETEIG, T., BØDKER, K., DITTRICH, Y., MOGENSEN, P. & SIMONSEN, J. 2013. Participatory IT Design: Designing for Business and Workplace Realities. In: ROBERTSON, T. & SIMONSEN, J. (eds.) *Routledge International Handbook of Participatory Design*. New York: Routledge.
- BROWN, J. S. & NEWMAN, S. E. 1985. Issues in Cognitive and Social Ergonomics: From Out House to Bauhaus. *Human-Computer Interaction*, 1, 359-391.
- BRUTON, N. 2002. *How to Manage the IT Help Desk: A Guide for User Support and Call Center*, Oxford, Butterworth Heinemann.
- BRÅTEN, S. 1973. Model Monopoly and Communication: Systems Theoretical Notes on Democratization. *Acta Sociologica*, 16, 98-107.
- BØDKER, K., KENSING, F. & SIMONSEN, J. 2004. *Participatory IT Design: Designing for Business and Workplace Realities*, Cambridge, Mass., MIT Press.
- CALVANI, A., FINI, A., RANIERI, M. & PICCI, P. 2012. Are young generations in secondary school digitally competent? A study on Italian teenagers. *Computers & Education*, 58, 797-807.
- CARROLL, J. M. 1990. *The Nurnberg Funnel: Designing Minimalist Instruction for Practical Computer Skill*, Cambridge, Mass., MIT Press.
- CARROLL, J. M., MACK, R. L., LEWIS, C. H., GRISCHKOWSKY, N. L. & ROBERTSON, S. R. 1985. Exploring Exploring a Word Processor. *Human-Computer Interaction*, 1.
- CARROLL, J. M. & ROSSON, M. B. 1987. Paradox of the active user. In: CARROLL, J. M. (ed.) *Interfacing thought: Cognitive aspects of human-computer interaction*. Cambridge, MA: MIT Press.

- CASE, D. O. 2012. *Looking for Information: A Survey of Research on Information Seeking, Needs, and Behavior*, Bingley, Emerald.
- CERTIPOINT INC. 2011. *Certiport* [Online]. Available: <http://www.certipoint.com/> [Accessed].
- CHAI, C. S., HWEE LING KOH, J. & TSAI, C.-C. 2013. A review of technological pedagogical content knowledge. *Journal of Educational Technology & Society*, 16.
- CHI, L. & DENG, X. N. 2011. Knowledge Transfer in Information Systems Support Community: Network Effects of Bridging and Reaching. *Thirty Second International Conference on Information Systems*. Shanghai.
- CHILLAREGE, K. A., NORDSTROM, C. R. & WILLIAMS, K. B. 2003. Learning from Our Mistakes: Error Management Training for Mature Learners. *Journal of Business and Psychology*, 17, 369-385.
- CIALDINI, R. B. 2001. Harnessing the Science of Persuasion. *Harvard Business Review*, 79, 72-79.
- CLARK, R. 2007. Leveraging multimedia for learning. Available: http://www.clarix.com/whitepapers/captivate_leveraging_multimedia.pdf.
- COCKBURN, A., GUTWIN, C., SCARR, J. & MALACRIA, S. 2015. Supporting Novice to Expert Transitions in User Interfaces. *ACM Computing Surveys*, 47, 31:1-36.
- Creating formulas using cell ranges in an Openoffice calc spreadsheet* 2010. Directed by COL CCNC. YouTube.
- COMMITTEE ON INFORMATION TECHNOLOGY LITERACY 1999. *Being Fluent with Information Technology*, Washington, D.C., National Academy Press.
- COMPEAU, HIGGINS, C. A. & HUFF, S. 1999. Social Cognitive Theory and Individual Reactions to Computing Technology: A Longitudinal Study. *MIS Quarterly*, 23, 145-158.
- COMPEAU, D. R. & HIGGINS, C. A. 1995. Application of Social Cognitive Theory to Training for Computer Skills. *Information Systems Research*, 6, 118-143.
- CONSTANT, D., SPROULL, L. & KIESLER, S. 1996. The Kindness of Strangers: The Usefulness of Electronic Weak Ties for Technical Advice. *Organization Science*, 7, 119-135.
- COOPER, J. 2006. The digital divide: the special case of gender. *Journal of Computer Assisted Learning*, 22, 320-334.
- COULSON, T., SHAYO, C., OLFMAN, L. & ROHM, C. E. T. 2003. ERP training strategies: conceptual training and the formation of accurate mental models. *SIGMIS '03*. Philadelphia, Pennsylvania: ACM.
- COVELLO, S. 2010. A Review of Digital Literacy Assessment Instruments. Syracuse University.
- CRABTREE, A., O'NEILL, J., TOLMIE, P., CASTELLANI, S., COLOMBINO, T. & GRASSO, A. 2006. The practical indispensability of articulation work to immediate and remote help-giving. In: HINDS, P. J. & MARTIN, D. (eds.) *CSCW'06*. Banff, Alberta: ACM.
- CUEVAS, H. M., FIORE, S. M. & OSER, R. L. 2002. Scaffolding cognitive and metacognitive processes in low verbal ability learners: Use of diagrams in computer-based training environments. *Instructional Science*, 30, 433-464.
- CULNAN, M. J. 1983. Chauffeured versus end user access to commercial databases: the effects of task and individual differences. *MIS Quarterly*, 7, 55-67.
- DAVIS, F. D. 1989. Perceived usefulness, perceived ease of use, and user acceptance of information technology. *MIS Quarterly*, 13, 319-340.
- DAVIS, S. A. & BOSTROM, R. P. 1993. Training End Users: An Experimental Investigation of the Roles of the Computer Interface and Training Methods. *MIS Quarterly*, 17, 61-85.
- DE MARCO, T. 1979. *Structured Analysis and System Specification*, Prentice Hall.

- DE VRIES, B., VAN DER MEIJ, H. & LAZONDER, A. W. 2008. Supporting reflective web searching in elementary schools. *Computers in Human Behavior*, 24, 649–665.
- DE WAAL, B. M. E. 2012. What makes end-user training successful? A mixed method study of a business process management system implementation. *International Journal of Knowledge and Learning*, 8, 166-183.
- DENG, X. & CHI, L. 2012. Understanding Postadoptive Behaviors in Information Systems Use: A Longitudinal Analysis of System Use Problems in the Business Intelligence Context. *Journal of Management Information Systems*, 29, 291-325.
- DOSTÁL, M. 2010. User Acceptance of the Microsoft Ribbon User Interface. In: MASTORAKIS, N. E. & MLADENOV, V. (eds.) *Advances in Data Networks, Communications, Computers*. Faro, Portugal: WSEAS Press.
- DREYFUS, H. L. & DREYFUS, S. E. 1986. *Mind over Machine: The Power of Human Intuition and Expertise in the Era of the Computer*, New York, The Free Press.
- DUARTE, N. 2008. *slide:ology: The Art and Science of Creating Great Presentations*, Beijing, O'Reilly.
- DUTKE, S. & REIMER, T. 2000. Evaluation of two types of online help for application software. *Journal of Computer Assisted Learning*, 16, 307-315.
- ECDL / ICDL 2009. ECDL / ICDL Sample Part-Tests. Syllabus Version 5.0. MSXPOpenOffice3.1. *ECDL / ICDL Sample Part-Tests*. ECDL Foundation.
- ECDL FOUNDATION. 2011. *European Computer Driving Licence Foundation* [Online]. Available: <http://www.ecdl.org/> [Accessed].
- EDUCATIONAL TESTING SERVICE. 2011. *ETS* [Online]. Available: <http://www.ets.org/> [Accessed].
- ESCHENBRENNER, B. 2010. *Towards a Model of Information Systems User Competency*. PhD, University of Nebraska - Lincoln.
- EVELAND, J. D., BLANCHARD, A., BROWN, W. & MATTOCKS, J. 1994. The role of “help networks” in facilitating use of CSCW tools. In: SMITH, J. B., SMITH, F. D. & MALONE, T. W. (eds.) *Proceeding CSCW '94*. New York: ACM.
- FINNEGAN, L. 1996. GCN training survey finds what works, what doesn't and why. *Government Computer News*, 43-44.
- FORSYTHE, D. E. 1999. “It’s Just a Matter of Common Sense”: Ethnography as Invisible Work. *Computer Supported Cooperative Work*, 8, 127–145.
- FU, W.-T. & GRAY, W. D. 2004. Resolving the paradox of the active user: stable suboptimal performance in interactive tasks. *Cognitive Science*, 28, 901–935.
- FURNAS, G. W., LANDAUER, T. K., GOMEZ, L. M. & DUMAIS, S. T. 1987. The vocabulary problem in human-system communication. *Communications of the ACM*, 30, 964-971.
- FURUTA, T. 2000. The Impact of Generating Spontaneous Descriptions on Mental Model Development. *Journal of Science Education and Technology*, 9, 247-256.
- GAGNÉ, R. M. & BRIGGS, L. J. 1974. *Principles of Instructional Design*, New York, Holt, Rinehart and Winston.
- GALLAGHER, K. P. & GALLAGHER, V. C. 2012. Organizing for post-implementation ERP: A contingency theory perspective. *Journal of Enterprise Information Management*, 25, 170 - 185.
- GALLETTA, D. F., AHUJA, M., HARTMAN, A., TEO, T. & PEACE, A. G. 1995. Social influence and end-user training. *Communications of the ACM*, 38, 70-79.
- GALLIVAN, M., SPITLER, V. & KOUFARIS 2005. Does Information Technology Training Really Matter? A Social Information Processing Analysis of Coworkers' Influence on IT Usage in the Workplace. *Journal of Management Information Systems*, 22, 153-192.

- GASSER, L. 1986. The Integration of Computing and Routine Work. *ACM Transactions on Office Information Systems*, 4, 205-225.
- GINNS, P. 2006. Integrating information: A meta-analysis of the spatial contiguity and temporal contiguity effects. *Learning and Instruction*, 16, 511-525.
- GOVINDARAJULU, C., REITHEL, B. J. & SETHI, V. 2000. A model of end user attitudes and intentions toward alternative sources of support. *Information & Management*, 37, 77-86.
- GRANT, D. M., MALLOY, A. D. & MURPHY, M. C. 2008. A Comparison of Student Perceptions of their Computer Skills to their Actual Abilities. *Journal of Information Technology Education*, 8, 141-160.
- GRANT, D. M., MALLOY, A. D. & MURPHY, M. C. 2009. A Comparison of Student Perceptions of their Computer Skills to their Actual Abilities. *Journal of Information Technology Education*, 8, 141-160.
- GRAVILL, J. & COMPEAU, D. 2008. Self-regulated learning strategies and software training. *Information & Management*, 45, 288-296.
- GREENBAUM, J. & KYNG, M. 1991. *Design at Work: Cooperative Design of Computer Systems*, Hillsdale, New Jersey, Lawrence Erlbaum.
- GRIGOREANU, V., BURNETT, M., WIEDENBECK, S., CAO, J., RECTOR, K. & KWAN, I. 2012. End-user debugging strategies: A sensemaking perspective. *ACM Trans. Comput.-Hum. Interact.*, 19, 1-28.
- GRONLUND, N. E. 1998. *Assessment of Student Achievement*, Neeham Heights, MA, Allyn and Bacon.
- GROSSMAN, R. & SALAS, E. 2011. The transfer of training: what really matters. *International Journal of Training and Development*, 15, 103-120.
- GROSSMAN, T., FITZMAURICE, G. & ATTAR, R. 2009. A survey of software learnability: metrics, methodologies and guidelines. *CHI '09: Proceedings of the 27th international conference on Human factors in computing systems*. New York: ACM.
- GUGERTY, L. 2007. Cognitive components of troubleshooting strategies. *Thinking and Reasoning*, 13, 134 – 163.
- GUO, P. J., KIM, J. & RUBIN, R. How video production affects student engagement: an empirical study of MOOC videos. Proceedings of the first ACM conference on Learning@ scale conference, 2014. ACM, 41-50.
- HADJERROUIT, S. 2008. Using a Learner-Centered Approach to Teach ICT in Secondary Schools: An Exploratory Study. *Issues in Informing Science and Information Technology*, 5, 233-259.
- HAKKARAINEN, K., ILOMÄKI, L., LIPPONEN, L., MUUKKONEN, H., RAHIKAINEN, M., TUOMINEN, T., LAKKALA, M. & LEHTINEN, E. 2000. Students' skills and practices of using ICT: results of a national assessment in Finland. *Computers & Education*, 34, 103-117.
- HALASZ, F. G. & MORAN, T. P. 1983. Mental models and problem solving in using a calculator. *CHI '83 Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. New York: ACM.
- HALBESLEBEN, J. R. B., WAKEFIELD, D. S., WARD, M. M., BROKEL, J. & CRANDALL, D. 2009. The Relationship Between Super Users' Attitudes and Employee Experiences With Clinical Information Systems. *Medical Care Research and Review*, 66, 82-96.
- HARTWICK, J. & BARKI, H. 1994. Explaining the role of user participation in information system use. *Management Science*, 40, 440-465.
- HATTIE, J. 2009. *Visible Learning: A Synthesis of Over 800 Meta-Analyses Relating to Achievement*, Oxon, UK, Routledge.

- HEARST, M. 2003. Information Visualization: Principles, Promise, and Pragmatics. *CHI 2003 tutorial*.
- HEEKS, R. 1998. Information Age Reform of the Public Sector: The Potential and Problems of IT for India. *Information Systems for Public Sector Management Working Paper*. Manchester: Institute for Development Policy and Management, University of Manchester.
- HERSKIN, B. 2006. *Brugeruddannelse i praksis*, Copenhagen, Nyt Teknisk Forlag.
- HIGNITE, M., MARGAVIO, T. M. & MARGAVIO, G. W. 2009. Information literacy assessment: Moving beyond computer literacy. *College Student Journal*, 43, 812-821.
- HINDS, P. J. 1999. The curse of expertise: The effects of expertise and debiasing methods on prediction of novice performance. *Journal of Experimental Psychology: Applied*, 5, 205-221.
- HMELO-SILVER, C. E., DUNCAN, R. G. & CHINN, C. A. 2007. Scaffolding and Achievement in Problem-Based and Inquiry Learning: A Response to Kirschner, Sweller, and Clark (2006). *Educational Psychologist*, 42, 99-107.
- HOLTON III, E. F. 1996. The Flawed Four-Level Evaluation Model. *Human Resource Development Quarterly*, 7, 5-21.
- HOWELL, J. M. 2005. The right stuff: Identifying and developing effective champions of innovation. *Academy of Management Perspectives*, 19, 108-119.
- IDEN, J. & EIKEBROKK, T. R. 2013. Implementing IT Service Management: A systematic literature review. *International Journal of Information Management*, 33, 512– 523.
- ISAKSEN, H., IVERSEN, M., KAASBØLL, J. & KANJO, C. 2017. Design of Tooltips for Data Fields: A Field Experiment of Logging Use of Tooltips and Data Correctness. In: MARCUS, A. & WANG, W. (eds.) *DUXU 2017*. Springer.
- ISO/IEC 2008. Systems and software engineering - Requirements for designers and developers of user documentation. Geneva: International Organization for Standardization.
- ITIL - AXELOS 2011. ITIL Service Operation. London: TSO.
- KAASBØLL, J. 2014. Suitability of diagrams for IT user learning. *ISDOC '14 Proceedings of the International Conference on Information Systems and Design of Communication* New York: ACM.
- KANAPATHY, K. & KHAN, K. I. 2012. Assessing the Relationship between ITIL Implementation Progress and Firm Size: Evidence from Malaysia. *International Journal of Business and Management*, 7, 194-199.
- KANSTRUP, A. M. & BERTELSEN, P. 2006. Participatory IT-support. In: JACUCCI, G. & KENSING, F. (eds.) *PDC 2006 - Proceedings of the ninth Participatory Design Conference*. Trento, Italy: ACM.
- KARUPPAN, C. & KARUPPAN, M. 2008. Resilience of super users' mental models of enterprise-wide systems. *European Journal of Information Systems*, 17, 29-46.
- KATO, T. 1986. What “question-asking protocols” can say about the user interface. *International Journal of Man-Machine Studies*, 25, 659–673.
- KEHOE, E. J., BEDNALL, T. C., YIN, L., OLSEN, K. N., PITTS, C., HENRY, J. D. & BAILEY, P. E. 2009. Training adult novices to use computers: Effects of different types of illustrations. *Computers in Human Behavior*, 25, 275–283.
- KEITH, N. & FRESE, M. 2008. Effectiveness of error management training: A meta-analysis. *Journal of Applied Psychology*, 93, 59-69.
- KENSING, F. & MUNK-MADSEN, A. 1993. PD: structure in the toolbox. *Communications of the ACM*, 36, 78-8.
- KIILI, K. & KETAMO, H. 2007. Exploring the Learning Mechanism in Educational Games. *Journal of Computing and Information Technology*, 4, 319–324.

- KIRKPATRICK, D. L. 1959. Techniques for evaluating training programs. *Journal of American Society of Training Directors*, 13, 21-26.
- KIRKPATRICK, D. L. 1975. *Evaluating Training Programs*, San Francisco, Berrett-Koehler.
- KIRKPATRICK, D. L. & KIRKPATRICK, J. D. 2006. *Evaluating Training Programs: The Four Levels* San Francisco, Berrett-Koehler.
- KORPELAINEN, E. & KIRA, M. 2010. Employees' choices in learning how to use information and communication technology systems at work: strategies and approaches. *International Journal of Training and Development*, 14, 32–53.
- KUMAR, S. 2010. *DebugMode Wink* [Online]. DebugMode. Available: <http://www.debugmode.com/wink/> [Accessed 5 Jan 2012].
- LARRES, P. M., BALLANTINE, J. & WHITTINGTON, M. 2003. Evaluating the validity of self-assessment: measuring computer literacy among entry-level undergraduates within accounting degree programmes at two UK universities. *Accounting Education: An International Journal* 12, 97-112.
- LAZONDER, A. W. 2005. Do two heads search better than one? Effects of student collaboration on web search behaviour and search outcomes. *British Journal of Educational Technology*, 36, 465–475.
- LI, Y. & RANIERI, M. 2010. Are ‘digital natives’ really digitally competent?—A study on Chinese teenagers. *British Journal of Educational Technology*, 41, 1029–1042.
- LIM, K. H., WARD, L. M. & BENBASAT, I. 1997. An Empirical Study of Computer System Learning: Comparison of Co-Discovery and Self-Discovery Methods. *Information Systems Research*, 8, 254-272.
- LIMAYEM, M., HIRT, S. G. & CHEUNG, C. M. K. 2007. How Habit Limits the Predictive Power of Intention: The Case of Information Systems Continuance. *MIS Quarterly*, 31, 705-737.
- MACKAY, W. E. 1990. Patterns of sharing customizable software. *CSCW '90 Proceedings of the 1990 ACM conference on Computer-supported cooperative work*. New York: ACM.
- MARKUS, M. L. & MAO, J.-Y. 2004. Participation in Development and Implementation - Updating An Old, Tired Concept for Today's IS Contexts. *Journal of the Association for Information Systems*, 5, 514-544.
- MARRONE, M. & KOLBE, L. M. 2010. Uncovering ITIL claims: IT executives' perception on benefits and Business-IT alignment. *Information Systems and E-Business Management*, 9, 363–380.
- MARSH, C. 2007. Strategic Knowledge of Computer Applications: The Key to Efficient Computer Use. *Issues in Informing Science and Information Technology*, 4, 269-276.
- MARTIN, A. P., IVORY, M. Y., MEGRAW, R. & SLABOSKY, B. 2005. Exploring the Persistent Problem of User Assistance. *ResearchWorks*. Seattle, WA, USA: University of Washington.
- MAYER, R. E. 1989. Models for Understanding. *Review of Educational Research*, 59, 43-64.
- MCDOWELL, C., WERNER, L., BULLOCK, H. & FERNALD, J. 2006. Pair programming improves student retention, confidence, and program quality. *Communications of the ACM*, 49, 90-95.
- MCEEDYA 2010. National Assessment Program - ICT Literacy Years 6 & 10 Report 2008. Carlton South, Victoria, Australia: Ministerial Council for Education, Early Childhood Development and Youth Affairs.
- MCINTIRE, S. & CLARK, T. 2009. Essential Steps in Super User Education for Ambulatory Clinic Nurses. *Urologic Nursing*, 29, 337-342.
- MCNEIVE, J. E. 2009. Super Users Have Great Value in Your Organization. *Computers, Informatics, Nursing*, 136-139.

- MERRITT, K., SMITH, K. D. & DI RENZO JR., J. C. 2005. An investigation of self-reported computer literacy: Is it reliable? *Issues in Information Systems*, VI, 289-295.
- MITRA, S. *Kalkaji* [Online]. Available: <http://www.flickr.com/photos/tedconference/8493285132/> [Accessed 6 Nov 2013].
- MITRA, S., DANGWAL, R., CHATTERJEE, S., JHA, S., BISHT, R. S. & KAPUR, P. 2005. Acquisition of computing literacy on shared public computers: children and the “hole in the wall.” *Australasian Journal of Educational Technology* 21, 407-426.
- MUNKVOLD, R. 2003. End User Support Usage. In: GORDON, S. R. (ed.) *Computing information technology: the human side*. Hershey, PA, USA: Idea Group Inc.
- NATHAN, M. J. & KOEDINGER, K. R. 2000. An Investigation of Teachers' Beliefs of Students' Algebra Development. *Cognition and Instruction*, 18, 209-237.
- NGOMA, C., KAASBØLL, J. J. & AANESTAD, M. From User Training to In-Service Support. In: CUNNINGHAM, P. & CUNNINGHAM, M., eds. *IST-Africa 2008 Conference Proceedings, 2008*. International Information Management Corporation Limited.
- NIELSEN, J. 1993. *Usability Engineering*, Boston, AP Professional.
- NIELSEN, J. 1994. Estimating the number of subjects needed for a thinking aloud test. *International Journal of Human-Computer Studies*, 41, 385–397.
- NIELSEN, J., DIRCKINCK-HOLMFELD, L. & DANIELSEN, O. 2003. Dialogue Design-With Mutual Learning as Guiding Principle. *International Journal of Human-Computer Interaction*, 15, 21-40.
- NILSEN, H. & SEIN, M. 2004. What is really important in supporting end-users? *Proceedings of the 2004 SIGMIS conference on Computer personnel research: Careers, culture, and ethics in a networked environment*. ACM.
- NORMAN, D. 1988. *The Psychology Of Everyday Things*, New York, Basic Books.
- NOVICK, D. G., ANDRADE, O. D. & BEAN, N. 2009. The micro-structure of use of help. *SIGDOC '09*. New York: ACM.
- NOVICK, D. G., ELIZALDE, E. & BEAN, N. 2007. Toward a more accurate view of when and how people seek help with computer applications. *SIGDOC '07*. New York: ACM.
- NOVICK, D. G. & WARD, K. 2006. Why don't people read the manual? *SIGDOC '06*. New York: ACM.
- NÜCKLES, M., WINTER, A., WITTEWERT, J., HERBERT, M. & HÜBNER, S. 2006. How do Experts Adapt their Explanations to a Layperson's Knowledge in Asynchronous Communication? An Experimental Study. *User Modeling and User-Adapted Interaction*, 16, 87-127.
- OECD 2011. *PISA 2009 Results: Students on Line: Digital Technologies and Performance (Volume VI)*.
- OLSEN, K. A. & MALIZIA, A. 2011. Automated Personal Assistants. *Computer*.
- ORMROD, J. E. 1995. *Human Learning*, Englewood Cliffs, New Jersey, Merrill.
- ORMROD, J. E. 2012. *Human Learning*, Englewood Cliffs, New Jersey, Merrill.
- PHELPS, R., ELLIS, A. & HASE, S. 2001. The role of metacognitive and reflective learning processes in developing capable computer users. . *Meeting at the crossroads: proceedings of the 18th Annual Conference of ASCILITE*. Melbourne: Southern Cross University.
- POE, S. S., ABBOTT, P. & PRONOVOST, P. 2011. Building Nursing Intellectual Capital for Safe Use of Information Technology: A Before-After Study to Test an Evidence-Based Peer Coach Intervention. *Journal of Nursing Care Quality*, 26, 110-119.
- POLITES, G. L. & KARAHANNA, E. 2012. Shackled to the Status Quo: The Inhibiting Effects of Incumbent System Habit, Switching Costs, and Inertia on New System Acceptance. *MIS Quarterly*, 36, 21-42.

- POOLE, E. S., CHETTY, M., MORGAN, T., GRINTER, R. E. & EDWARDS, W. K. 2009. Computer help at home: methods and motivations for informal technical support. *CHI '09*. New York: ACM.
- PRICE, S. & FALCÃO, T. P. 2011. Where the attention is: Discovery learning in novel tangible environments. *Interacting with Computers*, 23, 499–512.
- PUUSTINEN, M. & ROUET, J.-F. 2009. Learning with new technologies: Help seeking and information searching revisited. *Computers & Education*, 53, 1014–1019.
- RAMSDEN, P. 2003. *Learning to Teach in Higher Education*, London, RoutledgeFalmer.
- REYNOLDS, G. 2010. *Presentation zen design : simple design principles and techniques to enhance your presentations*, Berkeley, New Riders.
- RIEMAN, J. 1996. A field study of exploratory learning strategies. *Transactions on Computer-Human Interaction*, 3, 189-218.
- RODRIGUEZ, M. C. 2005. Three Options Are Optimal for Multiple-Choice Items: A Meta-Analysis of 80 Years of Research. *Educational Measurement: Issues and Practice*, 24, 3-13.
- ROSCOE, R. D. & CHI, M. T. H. 2007. Understanding Tutor Learning: Knowledge-Building and Knowledge-Telling in Peer Tutors' Explanations and Questions. *Review of Educational Research*, 77, 534-574.
- ROSLING, H. 2006. Hans Rosling shows the best stats you've ever seen. New York City and Vancouver: TED Ideas worth spreading.
- ROURKE, L. & KANUKA, H. 2009. Learning in Communities of Inquiry: A Review of the Literature. *Journal of Distance Education*, 23, 19-48.
- SANTHANAM, R., SELIGMAN, L. & KANG, D. 2007. Postimplementation Knowledge Transfers to Users and Information Technology Professionals. *Journal of Management Information Systems*, 24, 171-199.
- SCHOENFELD, A. H. 1992. Learning to think mathematically: Problem solving, metacognition, and sense-making in mathematics. In: GROUWS, D. A. (ed.) *Handbook for Research on Mathematics Teaching and Learning*. New York: Macmillan.
- SCOTT, J. E. 2006. Post-Implementation Usability of Erp Training Manuals:The User's Perspective. *Information Systems Management*, 22, 67-77.
- SEIN, M. K. & BOSTROM, R. P. 1989. Individual differences and conceptual models in training novice users. *Human-Computer Interaction*, 4, 197-229.
- SEIN, M. K., BOSTROM, R. P. & OLFMAN, L. 1998. Conceptualizing IT training for the workforce of the future. *SIGCPR*, 30, 223-241.
- SEIN, M. K., BOSTROM, R. P. & OLFMAN, L. 1999. Rethinking End-User Training Strategy: Applying a Hierarchical Knowledge-Level Model. *Journal of End User Computing*, 11, 32-39.
- SFARD, A. 1991. On the Dual Nature of Mathematical Conception: Reflections on Processes and Objects as Different sides of the Same Coin. *Educational Studies in Mathematics*, 22, 1-36.
- SHARMA, R. & YETTON, P. 2007. The contingent effects of training, technical complexity, and task interdependence on successful information systems implementation. *MIS Quarterly*, 31, 219-238.
- SHAW, N. C., DELONE, W. H. & NIEDERMAN, F. 2002. Sources of dissatisfaction in end-user support: an empirical study. *ACM SIGMIS Database*, 33, 41-55.
- SHNEIDERMAN, B. & PLAISANT, C. 2010. *Designing the User Interface: Strategies for Effective Human-Computer Interaction*, Boston, Pearson.
- SHRAGER, J. & KLAHR, D. 1986. Instructionless learning about a complex device: the paradigm and observations. *International Journal of Man-Machine Studies*, 25, 153-189.

- SIEBER, V. 2009. Diagnostic online assessment of basic IT skills in 1st-year undergraduates in the Medical Sciences Division, University of Oxford. *British Journal of Educational Technology*, 40, 215-226.
- SIMON, S. J. & WERNER, J. M. 1996. Computer training through behavior modeling, self-paced, and instructional approaches: A field experiment. *Journal of applied psychology*, 81, 648-659.
- SIMONSEN, J. & ROBERTSON, T. 2013. *Routledge International Handbook of Participatory Design*, New York, Routledge.
- SINK, C., SINK, M., STOB, J. & TANIGUCHI, K. 2008. Further evidence of gender differences in high school-level computer literacy. *Chance*, 21, 49-53.
- SMART, K. L., WHITING, M. E. & DETIENNE, K. B. 2001. Assessing the Need for Printed and Online Documentation: A Study of Customer Preference and Use. *Journal of Business Communication*, 38, 285-314.
- SPEIER, C. & BROWN, C. V. 1997. Differences in end-user computing support and control across user departments. *Information & Management*, 32, 85-99.
- SPITLER, V. K. 2005. Learning to Use IT in the Workplace: Mechanisms and Masters. *Organizational and End User Computing*, 17, 1-25.
- STAMATOVA, E. & KAASBØLL, J. J. 2007. Users' Learning of Principles of Computer Operations. *Issues in Informing Science and Information Technology*, 4, 291-306.
- STODOLSKY, S. 1988. *The subject matters : classroom activity in math and social studies*, Chicago, University of Chicago Press.
- STUART, L. H., MILLS, A. M. & REMUS, U. 2009. School leaders, ICT competence and championing innovations. *Computers & Education*, 53, 733-741.
- SUBRAHMANYAN, N., BECKWITH, L., GRIGOREANU, V., BURNETT, M., WIEDENBECK, S., NARAYANAN, V., BUCHT, K., DRUMMOND, R. & FERN, X. 2008. Testing vs. Code Inspection vs. ... What Else? Male and Female End Users' Debugging Strategies. In: BURNETT, M. (ed.) *Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*. New York: ACM.
- SYKES, T. A., VENKATESH, V. & GOSAIN, S. 2009. Model of Acceptance with Peer Support: A Social Network Perspective to Understand Employees' System Use. *MIS Quarterly*, 33, 371-393.
- TAYLOR, P. J., RUSS-EFT, D. F. & CHAN, D. W. L. 2005. A Meta-Analytic Review of Behavior Modeling Training. *Journal of Applied Psychology*, 90, 692-709.
- TUFTE, E. 1990. *Envisioning information*, Cheshire, Conn, Graphics Press.
- TUFTE, E. 2011. *The work of Edward Tufte and Graphics Press* [Online]. [Accessed].
- VAN DER SANDEN, J. M. M. & TEURLINGS, C. C. J. 2003. Developing competence during practice periods: The learner's perspective. In: TUOMI-GROHN, T. & ENGESTROM, Y. (eds.) *Between school and work: New perspectives on transfer and boundary crossing*. Oxford, UK: Elsevier Science.
- VAN VELSEN, L. S., STEEHOUDER, M. F. & DE JONG, M. D. T. 2007. Evaluation of User Support: Factors That Affect User Satisfaction With Helpdesks and Helplines. *IEEE Transactions on Professional Communication*, 50, 219-231.
- VAN VLIET, P. J. A. & KLETKE, M. G. 1994. The measurement of computer literacy: a comparison of self-appraisal and objective tests. *International Journal of Human-Computer Studies*, 40, 835-857.
- VAN WIJK, R., JANSEN, J. J. P. & LYLES, M. A. 2008. Inter- and Intra-Organizational Knowledge Transfer: A Meta-Analytic Review and Assessment of its Antecedents and Consequences. *Journal of Management Studies*, 45, 830-853.

- VANLEHN, K., SILER, S., MURRAY, C., YAMAUCHI, T. & BAGGETT, W. B. 2003. Why Do Only Some Events Cause Learning During Human Tutoring? *Cognition and Instruction*, 21, 209-249.
- VENKATESH, V., MORRIS, M. G., DAVIS, G. B. & DAVIS, F. D. 2003. User acceptance of information technology: Toward a unified view. *MIS Quarterly*, 27, 425-478.
- VESSEY, I. & CONGER, S. A. 1994. Requirement Specification: Learning Object, Process, and Data Methodologies. *Communications of the ACM*, 37, 102-113.
- VOLKOFF, O., ELMES, M. B. & STRONG, D. M. 2004. Enterprise systems, knowledge transfer and power users. *The Journal of Strategic Information Systems*, 13, 279-304.
- VON NEUMANN, J. 1945. First Draft of a Report on the EDVAC. Philadelphia: University of Pennsylvania.
- WAGNER, E. L. & PICCOLI, G. 2007. Moving beyond user participation to achieve successful IS design. *Communications of the ACM*, 50, 51-55.
- WALRAVEN, A., BRAND-GRUWELB, S. & BOSHUIZEN, H. P. A. 2010. Fostering transfer of websearchers' evaluation skills: A field test of two transfer theories. *Computers in Human Behavior*, 26, 716-728.
- WAYTZ, A., CACIOPPO, J. & EPLEY, N. 2010. Who Sees Human?: The Stability and Importance of Individual Differences in Anthropomorphism. *Perspectives on Psychological Science*, 5, 219-232.
- WENGER, E. 1998. *Communities of practice : learning, meaning, and identity*, Cambridge, Cambridge University Press.
- WESTBROOK, L. 2006. Mental models: a theoretical overview and preliminary study. *Journal of Information Science*, 32, 563-579.
- WITTEWER, J. & RENKL, A. 2010. How Effective are Instructional Explanations in Example-Based Learning? A Meta-Analytic Review. *Educational Psychology Review*, 22, 393-409.
- ÖSTBY, J. 2012. *KRUT* [Online]. sourceforge. Available: <http://krut.sourceforge.net/> [Accessed 5 Jan 2012].
- ÅSAND, H.-R. H. & MØRCH, A. 2006. Super Users and Local Developers: The Organization of End User Development in an Accounting Company. *Journal of Organizational and End User Computing*, 18, 1-21.

Index

- abstract concepts, 37
- abstract entities, 54
- abstraction, 114
- Adobe Captivate, 24
- adolescents, 115
- adults, 115
- advanced, 21
- age, 114
- algorithms, 134
- analogy model, 49
- anxiety, 77
- architecture, 54
- assessing competence, 127
- assessing IT skills, 25
- assessment, 105
- balloon help, 94
- Behaviour, 125
- behavioural change, 125
- behaviourism, 26
- behaviouristic learning theory, 92
- Bloom's taxonomy, 12
- boundary interaction, 141
- boundary object, 141
- boys, 140
- brevity, 19
- broker, 141
- bug tracking, 159
- Certiport, 135
- champions, 152
- child, 114
- children, 68, 72, 139
- Clippy, 98
- coding, 10
- cognitive load, 16
- command style user interface, 92
- community of practice, 141
- comparing input and output, 37
- competence*, 13
- competence standards, 134
- competence tests, 135
- completeness, 18, 26
- computer science education, 10
- computer supported learning, 10
- conceptualisation, 37
- concrete experience, 37
- concrete operational stage, 115
- confidence, 77
- confronting business misconceptions, 34
- Confronting functional misconceptions, 43
- confronting misconceptions, 34, 108

constructivism, 37

constructivist, 9

context-free help, 97

context-sensitive help, 96

contrast, 119

counterfactual situations, 115

coworkers' influence, 151

critique session, 165

customizing, 150

data model, 53

descendants, 50

direction, 17

directive, 17

discrimination error, 62

distinction between concepts, 63

document structure. *See*

documentation, 19

documentation in training, 108

ease of use, 27

ECDL, 135

educational science, 10

Educational Testing Service, 135

error encouragement, 105

European Computer Driving License, 135

examples, 54

experimentation, 69

exploration, 68

explore, 105

external source for learning, 49

extinctions, 92

facilitating conditions, 28

fantasy phase, 166

feedback, 18

file system, 47

FITness, 134

Fluency with IT, 134

folders, 47

font size, 119

frequently asked questions, 160

functional dependency, 50

functional models, 39

functional understanding, 37

Future workshops, 165

girls, 140

grid, 50

guide, 107

hardware, 123

heuristic evaluation, 99

hidden structures, 46

hierarchy, 50

Hole-In-The-Wall, 68

homonyms, 64

human computer interaction, 9

hypothesis, 72

hypothetical situations, 115

ICDL, 135

icon, 93

illustrations, 118

imitating peers, 77

imitation, 15

immediate reinforcements, 92

impasse, 104

Impress, 117

information literacy, 9

information systems, 9

informative reinforcements, 92

inline help, 94

input state, 39

instruct, 15

instruction, 107

instruction sheet, 16

instruction video, 22

instructions, 16, 105

interference, 41

International Computer Driving License, 135

Internet, 60

Internet Service Provider, 60

Introduction, 108

issue tracking system, 159

IT concepts, 36

IT literacy, 140

IT personnel, 143

IT skills, 14

IT specialists, 8

IT structures, 46

ITIL, 159

Keynote, 117

Kirkpatrick's model of evaluation of training, 125

KRUT, 24

layered architecture, 59

layers, 59

leaders, 151

learn, 36

learnability, 91

learning, 13, 65

Learning, 125

learning competence, 65

Learning objective, 121

learning oriented, 66

learning oriented users, 105

learning styles, 49

lecturer, 8

level of abstraction, 114

level of mastery, 38

local area network, 60

master slides, 116

mental models, 39
metacognition, 65
minimal manual, 19
minimal manuals, 30
minimalism, 19
misconception, 108
misfit, 88
misunderstanding, 108
Mitra, Sugata, 68
motivation, 121
multiple choice, 34
mutual learning, 162
negative reinforcements, 92
network, 50
network protocol, 59
non-trivial assignments, 109
novelty, 41
novice, 21
object first, 76
office assistant, 98
operation first, 76
organisational learning, 10
Outcome, 121, 125
output state, 39
overburdening, 104
own data, 123
pair, 104
pedagogy, 10
perceived ease of use, 27
perceived usefulness, 27
performance oriented, 66
performance oriented users, 105
positive reinforcement, 92
PowerPoint, 117
Practicals, 108
practice, 141
precedents, 50
preschool children, 115
presentation program, 116
Prezi, 117
problem solving approach, 109
problem solving competence, 13, 70
problem solving competences, 105
programming, 134
projector, 24, 107
protocol, 60
Prototypes, 166
proximity, 16
psychology, 10
punishments, 93
qualities of IT support, 158
Question-Suggestion, 100
Reaction, 125
realisation phase, 166

recognizable, 16

relationships between concepts, 63

repetition, 14

research cycle competence, 71

response time, 158

Responsibilities, 166

Result, 125

scaffolds, 13

screeintip, 94

search behaviour, 81

self-efficacy, 77, 106

self-reporting, 138

sequence, 50

sequential, 17

Sequential, 17

service desk, 159

short, 19

short term memory, 16

situated learning, 141

skill levels, 21

skill training, 24

skills, 12

skills acquisition, 12

slide, 115

slide design, 115

slow connections, 123

slow learners, 39

social influence, 28, 121

soft copies of the documentation, 124

software, 123

Software upgrades, 158

structural models, 44

structural semantic understanding, 60

structural understanding, 37

Summary, 108

superusers, 13, 106, 143, 153

supporters, 157

System-initiated help, 98

tasks, 29

teacher, 8

teacher training college, 9

teachers, 143

technology acceptance model, 151

Technology Acceptance Model, 27

terminology, 93

terminology issue, 21

terminology problem, 98

text and illustrations, 117

text flow, 139

thinking aloud, 100

ticket system, 159

tooltip, 94

topology, 50

training environment similar to the
business, 124

training generation and testing of
hypotheses, 106

training material, 158

training module, 102

transfer, 120

troubleshooting, 70

type-instance, 55

types, 54

understanding, 12

Understanding IT in business, 31

usefulness, 27, 29

user group, 54

user interface, 25

user training, 15

verbal abilities, 40

versions, 123

video, 22, 49, 107

Visiting other installations, 165

visual ability, 48

visualising data structure, 51

von Neumann architecture, 58

web host, 60

Wink, 24

wizard, 96

workaround, 88, 149