Human IT

Tidskrift för studier av IT ur ett humanvetenskapligt perspektiv

Futurism and retrospect

User strategies to learning opportunities in cases of surprise or confusion

av Stefan Holmlid

Abstract

Astrid suddenly happens to get the table in the report she is in a hurry finishing up to look like she wanted it to. Surprised she makes her tutor take care of her interaction history all the way back to when she started to fiddle around with the table, saving it for a later learning session. For now it suffices that it looks alright and that the report is soon finished.

Later, in a more comfortable learning environment, she brings up the table design problem with her tutor. She browses through the different steps she made and reads the comments from the tutor, realizing that it all, of course, was very simple. Or at least straightforward. She defines the correct steps as a template, and the whole sequence as a reminder, for later use as a shortcut to the correct look as well as online help.

A week later she gets stuck on quite a different task. She gets confused, and even a little bit annoyed. She asks her colleague for help. She scribbles down what she wants to do, and makes a small sketch of the endresult she wants to achieve. Her colleague suggest a series of steps that should get her at least in the right neighbourhood. She follows the advice given and with only minor adjustments she succeeds with her task.

Later that evening, working on overtime, she once again needs to get the table to look as the other week. She brings up the template and adjusts the number of columns to the number she needs. While at it, she changes the style of the table heading, to be a little bit more subtle looking.

Imagine your wordprocessor being designed and constructed so as to be the tutor

as well as the colleague.

Innehåll

---

1. It’s supposed to be a support, but does not give support

It happens to all of us. The tool we are using does not fully support the task we have at hand quite in the way we expect it to. Generally this is true for the more complex tools we use for professional tasks, and especially for the large amounts of offtheshelf software put on our desks. And this kind of software is meant to support us being productive, effective, contributing to the valueadding process, and the multiloop learning processes at work. Especially when it comes to learning to use them, offtheshelf products of today stand short.

The short scenario with Astrid is an attempt to describe quite a different situation which rests on actual problematic situations and the kinds of knowledge interest that users have in those situations.

(Åter till början av artikeln)

2. A multitude of efforts

There are lots of attempts to produce software more apt for use at work. From a humancomputer interaction point-of-view there are basically four approaches, or perspectives if you will, that are concerned with studies of learning to use software. These perspectives narrow down on the learning-to-use problem quite differently.

The usability engineering approach (see Nielsen 1993, Löwgren 1993) identifies usability, of which learnability is an aspect, as a property of the software product. This approach argues that as long as it is possible to define learning-to-use it is possible to build a system which is easy to learn. By focusing on learnability in the usability-oriented design process, the necessary and sufficient means will be found, for building a system that its users will learn to use. The assumption is twofold. The primary is that there exists a way of pointing out a learning-quality in softwareuse. The secondary, and the most important, is that there exists a *product* property termed learnability, that it is measurable, and that the measures correspond to the actual learning-to-use that takes place in use of the product. The weakness is that it is not possible to define, and formulate requirements for, the learning-to-use problem, decompose and then solve it. In fact we do not know the problem until we suggested a solution.

This is the standpoint which the learner-centered approach (see Soloway, Guzdial & Hay 1994) takes as a starting point, there is a need to learn about the learning-to-use problem from those who are in the process of learning. This positions the solution at the *method* level. The learner-centered approach argues that if we can’t define the learning-to-use problem, we can devise a method that enables us to find a solution. The weakness is that a method relies on the people, the organisation, the motivational factors and the resources with which it is equipped and carried through. It is all too easy to ignore the method, or let the method be an end in itself, without looking at the assumptions behind it, much like engineering methods has been blinded for a long time.

The documentalist approach (see Carroll 1990) pinpoint the dependence on people and focus on finding assumptions that can be used across methods, basic principles which might be regarded as good principles to use to make good learning material. Minimalism, e.g. (see Carroll 1998), focuses on documentation, in a broad sense, including manuals, on-line help, tool-tips, terminology and so on. Documentalists say that even though we try to find better solutions we need to have guiding principles for the written material that will accompany the software. One assumption is that the functions of the software comes first, then comes the documentation. The weakness is that the learning material *accompany* the software and thus makes learning the software easier without making the software easier to learn.

The tutoring approach (see Shute 1994 for an overview) focuses on the software as a technology, having defined the problem to be one of building intelligent systems, that adapt, or implement continuous adaptations to a learner’s strategy, which involves the detection of those strategies. Basically this is all about the modeling of intelligent or human behaviour, either to understand, act on or replicate it. A result of this is the construction of help-systems, agents or companions, striving to anticipate what the users is up to. The weakness is that they are hard to design, implement and sell, other than for very well defined small groups, where we might use knowledge acquisition as a way of finding out what all the parameters of the subject area and the learner population are.

Put in context of off-the-shelf software, as opposed to tailored systems, the weaknesses of these approaches have a large influence on the result. Off-the-shelf software, and especially many of the office systems, have such a broad application that it is impossible to know what the actual worktask for which it will be used is. The very idea behind them is that the task-domain is broad, and not very well defined.

To set usability goals, or devise a new method, will involve some users and work-tasks, but it is hard to say that the chosen subset is representative. Also, the act of decontextualized isolated product requirements is hard to do, without losing the meaning of the learning-quality in-use. To completely fulfill the minimalist heuristics fully the actual work-tasks are needed, although approximations and long-term studies will make certain minimalist designers better than others.

Altogether the assumptions behind these approaches rises new questions; how do we learn that a piece of software is easy to use, how do we learn that it is easy to learn to use. These questions focuses on the learning processes across all efforts to make products easy to use or easy to learn to use.

We have been studying learning within the context of one user and one computer with relatively focused tasks, in order to be able to decide on important ordering and details in the user interface, and what kinds of interaction methods to prefer for different computer focused tasks. To be able to go further we have to do studies of real use, over a longer period of time, focusing on the learning, behaviour and effects of learning to use a piece of software. After that we will have to reconsider the detailed user interface studies.

(Åter till början av artikeln)

3 Effort and focus

The different perspectives also have focused on slightly different aspects of learning and software. The basic division is between those dealing with the use of learning tools, and those dealing with learning to use the tools. There might be studied several others dimensions, such as technology used, the role of the developer, who the user is, etcetera. For the argument made here these are of secondary interest.

The learner-centered design approach as well as the modelling approach primarily focus on the use of learning tools, whereas the documentalist approach and the usability engineering approach focus on learning to use the tool.

The three approaches are not mutually exclusive. For example, you find studies of the learnability of a piece of software which is used as a learning media. In this overview I want to provide a framework within which studies on learning and software can be performed.

This »tool/learning» dichotomy seems fairly easy; dividing the efforts into those dealing with the use of learning tools and those dealing with learning to use the tools. Later, as you will see, this dichotomy will collapse.

(Åter till början av artikeln)

4 Missing out on the learning opportunities

From studies of computer use at two Swedish companies, one at that time medium sized media company, and one large bank. For a detailed account of the studies at the media company see Holmlid (1995, 1997). Fourteen users participated in a study using SUMI (Porteous, Kirakowski & Corbett 1993) and an semistructured interview. At the bank eight users acted as informants. The users answered a questionnaire to assess their computer self-efficiacy (Compeau & Higgins 1995) and eleven qualities-in-use. Also contextual interviews were performed during half a day. This procedure was performed before they received their teacher led training as well as four to six months after the training.

During these studies there has been identified opportunities for learning which today is supported only by informal learning environments, and to some extent by the organization itself.

(Åter till början av artikeln)

4.1 Environments of use and learning

It is fairly easy to study and pinpoint one of the most common qualities of environments of use and learning, their diversity. Here only a few will be mentioned (see also Holmberg 1996, and Busch 1993 for other overviews)

Software diversity has several dimensions. Of minor importance is the within-application version problem. Most of the time there are several different software families run in parallel with more complicated across application or across families problems, sometimes due to version differences. It is also fairly common that there is a proprietary system, especially at large companies, with quite a bit of installed base. There might be different versions of the same operating system running, or a user switches between more than one operating system. Or different users might use different operating systems, or even different computer platforms. It is hard to see any changes in software diversity at workplaces unless there is a drastic change, such as a shakeout or technological stagnation, in the information technology field.

Then there is the learning environment diversity. Focusing only om off-the-shelf software there is teacher-led training, self paced computer or non-computer supported instruction, third-party books, written manuals, on-line help, etc. For different software one person uses at work a few of these might be chosen. There are informal and formal learning environments, task-focused or task-free exploration, etc. It is hard to see any changes in learning-environment diversity due to technological development pressure, the introduction of new learning media as well as learning philosophies.

Task-diversity is manifold. Users are experts at doing their job, but to a varying degree experts at using certain pieces of software for different tasks. To be able to do their work smoothly they depend on others being experts at, e.g., operating systems and mail servers. Most work-tasks require the use of several pieces of software. The frequency of the work-tasks vary as does their complexity, and changes over time. And, still, the individual differences and preferences has only been touched upon.

(Åter till början av artikeln)

4.2 Help strategies

Users at the media company showed a clear picture of their preferences and their use of different help-strategies (see Holmlid 1997). Most commonly people ask their colleagues for help. Other strategies used were trial and error, use of personal notes or learning material, the on-line help,as well as the manual.

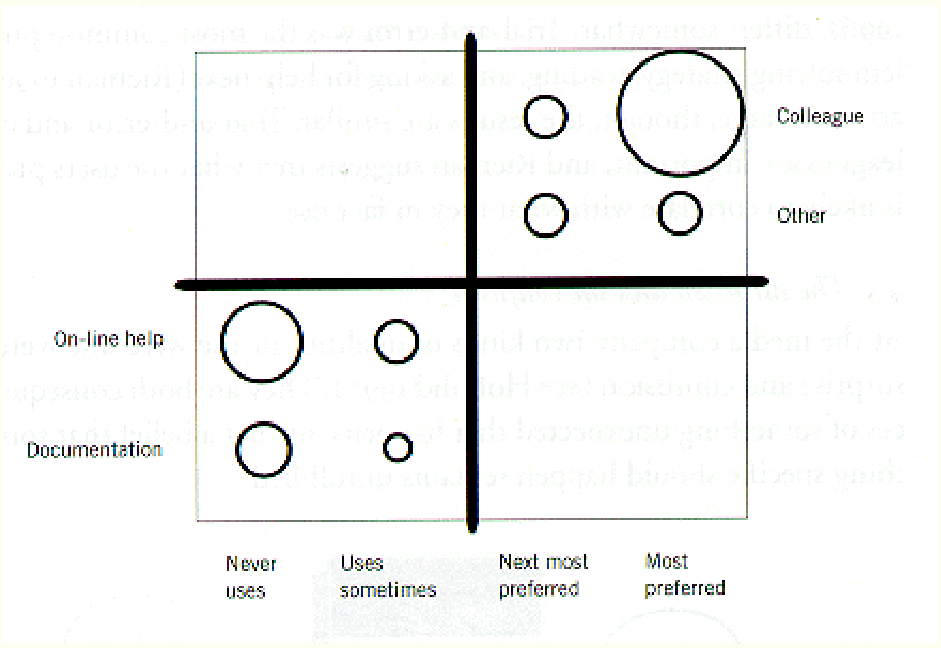

FIGURE 1. *Proportions of number of users' preferences and frequency of use of differentmethods for getting help at the media company.*

The methods of help provided for by the software producer are used inversely to their power of contextualization. At the same time the methods favoured are those which are the most personalized; colleagues, personal notes and training material. The expertise of the colleagues are appreciated, both in relation to the software and to the work tasks; it is easier with contextualized and personalized help. In the personal notes, and the training material, the user himself have made some personal investment, and partaken in activities around the material.

A similar pattern was observed at the bank. Trial-and-error strategies and colleagues were used most frequently, whereas the on-line help and the manual was used only as a result of the observer being present. For some work-tasks, characterized by being customized for a certain campaign involving specially programmed steps in a process of registering a loan, e.g., telephone support was common.

The pattern in a study of exploratory learning strategies (Rieman 1996), differs somewhat. Trial-and-error was the most common problem solving strategy, reading, and asking for help next (Rieman 1996, p 207). At large, though, the results are similar. Trial-and-error and colleagues are important, and Rieman suggests that what the users prefer is likely to correlate with what they in fact use.

(Åter till början av artikeln)

4.3 The surprised and the confused

At the media company two kinds of qualities-in-use were uncovered; surprise and confusion (see Holmlid 1997). They are both consequences of something unexpected that happens, or that a belief that something specific should happen remains unfulfilled.

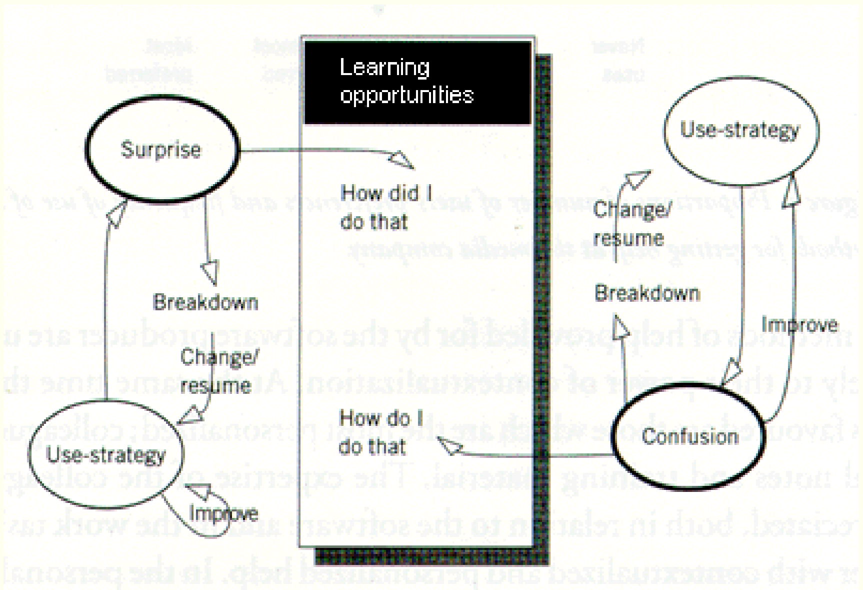

FIGURE 2. *Surprise and confusion in relationship to use-strategies and learning oppor-tunities.*

surprise as well as confusion are closely linked to breakdowns in usage. In these states lies two different, sincere knowledge interests. For the surprised user the interest is what he actually did. For the confused user the interest is what he should be doing instead of what he is doing. The surprised user has a historical interest for future use, whereas the confused user has a futurist interest for immediate use. Both users are in a situation where learning is possible, needed and to some extent wanted.

At the bank cases of surprise were rare, and did not lead to any direct consequential actions, only expressions of surprise. Cases of confusion were resolved either by trial-and-error, with the help of a colleague, through telephone support, through a work-around, or by looking at prepared procedural notes.

Carroll and Aaronson (1988) describes a study of two different kinds of help, how it works help, and how to do it help. Their how-to-do-it help would possibly help the confused user, but they have no way of catching the surprised user. The how-it-works help primarily focuses on the user who prepares himself.

Thomas, in his study of long term exploration (Thomas, 1996), identifies a retrospective knowledge interest which he calls serendipity. As a concept serendipity is a fairly broad concept. Surprise as it is described here is a kind of serendipity. As such it is interesting in itself, because it is a sudden accomplishment without prior learning, and with only a partial understanding of how the result was achieved. The surprised user, as opposed to other cases of serendipity, is experiencing first-time serendipity, and might experience another kind of serendipity in the same situation, if no learning takes place. On the other hand, this other kind of serendipity could be a signal that the user in fact learned something while being surprised.

In a study of exploratory learning strategies (Rieman 1996) only in very few occasions the category »stumbled onto by accident» was used. The system studied did not provide any specific support to learn from the surprises, simply because systems of today do not account for those kind of situations. The most used strategy was trial and error, which in some cases would be the same as the confused user trying to find out what to do.

Thomas also points out, as a variation to the Zone of Proximal Development, ZPD (Vygotsky 1986), the Zone of Exploration Model, ZEM (Thomas 1996). The surprised and the confused user both are in state of the ZEM. Possibly, if there as a breakdown which allows the user to reflect on his actions, the surprised users enters his ZPD where the piece of software has accidentally acted as the more capable peer. The confused user, on the other hand is in need of the more capable peer in order to be able to enter his ZPD.

It is evident that surprise as well as confusion are the basis for micro level learning environments that deserves to be further explored.

(Åter till början av artikeln)

5 Supporting individuals learning from doing

Many interpretations might be made of what has been said. In fact data might be used by any of the approaches to support their cause; we need to design better and usable documentation and on-line help, we need to include the technical writers in the development process, we need to use more intelligent techniques for delivering help on-line, we need to model the user so that the system might predict the help needed in a more appropriate manner.

More reasonable is to propose a combination of those arguments. We need to develop better designs, not of the on-line help in isolation, but of the *use of the system*. We need to include learning aspects, not only the technical writers, in the development process. We need to develop the technical possibilities, not only for the learning parts of a system, but for the system as a whole, in order to be able to provide better learning environments *including* the system. We need to find ways of modelling, not the user as much as the use of the system, and relate that to learning. That is, there is a need for a basic shift both in approaches to systems development as well as the use of the possibilities computer science provides us with. The limitations of what is possible to do is basically what is being done when we design and construct software.

As it stands today most pieces of software is scaffolding for a completing a task as well as for learning how to use the software for solving that task. For better or worse this points out precisely the software as a means for learning. Without adequate software solutions and support for the kinds of learning that is going on, there will be a continuing growth of non-read manuals, CD-ROM packages and formal training which struggle against a product which do not want to be learned about. The learning/tool dichotomy ends in a unity. We learn to use the tools using themselves as tools to learn; the dichotomy falls apart.

We know quite a lot about user learning. Rieman (1996) argues that users postpone their learning until driven by a real task, that task-free 9 exploration is unlikely to occur and that users use the most effective learning strategy provided. From studies at the media company and the bank we know that surprise and confusion occur, that personalization and contextualization is important when learning to use a piece of software.

It is possible to vision types of solutions that takes its stance before the artifact, and might be involved in the making of the artifact, and will eventually become a part of the artifact. That kind of solutions will inevitably transform the way we look at software, as well as the way we learn and use software. Some such visions will be presented here.

(Åter till början av artikeln)

5.1 Assessing recorded interaction history

In cases when we do not have the time or the motivation to learn at the very moment of activity, there is a need for a simple and easy technique of recording and later being able to assess histories of interaction which one has performed. This would provide the surprised user, or the confused user who were helped solving his problem, with a tool for creating the basis for a learning session.

(Åter till början av artikeln)

5.2 Temporary collaborative learning

The activity of asking a colleague for help is a social activity. As long as we construct and utilize complex tools for complex tasks in a complex and changing environment, this kind of activity will persist precisely in these tasks and with these tools. There is a need for simple and easy techniques which might be used in those collaborative learning situations, during ordinary work activity or in a more formal learning situation. To the formal session one might bring problems and activities to learn from, for the learner/mentor communication and activity to become a learning situation which might produce changes in behaviour as well as learning effects.

One such technique to build upon is the recorded interaction histories.

(Åter till början av artikeln)

5.3 Self made learning templates, retrospect

Once out of work, at home, in a formal learning situation or at the end of the day, trying to recapture the task, the situation and all the other task related things is hard. Better learning situations would be able to be created through using prerecorded interaction histories as templates for a learning session.

(Åter till början av artikeln)

5.4 Self made exercises

From the interaction histories and the use of them in learning situations it would be possible for the user to create his own exercises. As a teacher a tool like this would be useful to collect and refine exercises for a specific company.

(Åter till början av artikeln)

5.5 Self made help templates, recaps

Once used in a learning session a help template for task specific activities and situations might be created from the recaptured session for later use. It might be called upon by using keywords, sketches or whatever is the best way of describing what the user wants to do. In some cases the software itself might be able to anticipate the kind of help needed, through close assessment of an interaction log, thus being able to adapt the search facility to suit the task at hand.

(Åter till början av artikeln)

5.6 Self made application templates, recycling

Once used in a learning session a template for task specific activities and situations might be created from the recaptured session for later use. It might be called upon by using keywords, sketches or whatever is the best way of describing what the user wants to do. In some cases the software itself might anticipate, through close assessment of an interaction log, the possible templates that might be used at one specific point in time.

(Åter till början av artikeln)

6 A framework for continuations

A framework for research and development that would be a step on the way to software that from the beginning is developed as learning as well as use environments include system development methods, basic datastructures, dataformats and -storage, interaction techniques, software functionality, knowledge based support structures, and possibly even more.

The possibilities of software being tools we learn to use by using themselves to learn is limited by how we approach the learning-to-use problem, and thus what the approach enables us to count before the design and construction of a piece of software.

(Åter till början av artikeln)

About the author

Stefan Holmlid is a postgraduate student at the Department of Computer and Information Science at Linköping Universitet in Sweden where he works in the Usability Matters group at the Application Systems Laboratory. The focal point of his research is circling around issues of Usability, Utility, Evaluation and Interaction Design. Teaching undergraduate courses in HCI intended for students at the Institute of Technology as well as the Faculty of Arts and Sciences. Currently developing and teaching a Masters Programme in Interaction Design.

References

Busch, T. (1993). *Overføring av laering*. Avhandling for graden dr. oecon, TØH-serien 1993:4, Trondheims Økonomiske Høgskole, Trondheim, Norge.
Carroll, J., M. ed (1998). *Minimalism beyond the Nurnberg funnel*. MIT Press, Cambridge, MA.
Carroll, J., M. (1990). *The Nurnberg funnel: Designing minimalist instruction for practical computer skill*. MIT Press, Cambridge, MA.
Carroll, J., M., Aaronson, A., P. (1988). Learning by doing with simulated intelligent help. *Communications of the ACM*, 31(9):1064-1079.
Compeau, D., R., Higgins, C., A. (1995). Computer selfefficacy: Development of a measure and initial test. *MIS Quarterly*, June, 189-211.
Holmberg, L. (1996). *Design of learning environments for IT-users*. Report no 1996:02 (and 01). PHd thesis, Dept.Education and Educational Research, Göteborg University, Sweden.
Holmlid, S. (1997). User perceptions of effects of training: In search for qualities in use. *Linköping electronic articles in computer and information science*, Vol 2(1997): nr 8. http://www.ep.liu.se/ea/cis/1997/008/ August 29, 1997.
Holmlid, S. (1995). *The effect of end-user instruction on usability: Usability as a key quality of instruction*. Master Thesis, LiTH-IDA-Ex-9571, Linköpings universitet, Sweden.
Löwgren, J. (1993). *Human computer interaction: What every systems developer should know*. Studentlitteratur, Lund. **Nielsen**, J. (1993). *Usability engineering*. Academic Press, San Diego, CA.
Porteous, M., Kirakowski, J. and Corbett, M. (1993). *SUMI user hand-book*. Human Factors Research Group, University College Cork, Ireland.
Rieman, J. (1996). A field study of exploratory learning strategies. *ACM Transactions on Computer-Human Interaction*, 3(3):189-218.
Shute, V. (1994). Intelligent tutoring systems: Past, present and future. In D. Jonassen (ed), *Handbook of educational communications and technology*. Scholastic Publications.
Soloway, E., Guzdial, M., Hay, K., E. (1994) Learner-centered design : the challenge for HCI in the 21st century. *interactions*, 1(2):36-48.
Thomas, R., C. (1996). *Long term exploration and use of a text editor*. PHd Thesis, University of Western Australia, Nedlands, 6907, Australia, November 22, 1996.

Vygotsky, L. (1986). *Thought and language*. MIT Press, Cambridge, MA,

---

Åter till Human IT 3/1998