

Two cases of bad Web usability: banking and employee self service

Kai A. Olsen
Molde University College and University of Bergen

Abstract

Two cases are presented, an Internet banking system, used by 40 % of Norwegian banks, and an Employee Self Service System (ESS) developed for Norwegian State organizations and used by all state colleges. Neither system is designed for their users. The banking system has an error prone user interface never subjected to usability testing. This caused a user to transfer 500,000 Norwegian kroner to the wrong account. The ESS system requires users to spend significantly more time on simple administrative tasks than by using the previous paper based system.

The breakthrough technology and applications that we see today, of simple to use systems that use touch screens, wireless technology, and eInk, can give us the impression that all systems are up to this standard. While one cannot generalize from two cases, they are both examples of the low dissemination in society of basic concepts of human computer interaction, or of the unprofessionalism of many organizations.

1. Introduction

Twenty years ago the first efforts to develop a universal system of information sharing were started at CERN in Switzerland. This gives us nearly twenty years of user interface experience with the Web. However, the field of human computer interaction started many years before. The work became really important in the early eighties, when the development of the PC made computer technology available to new user groups, most with no computing experience. The basic rules of creating good user interfaces were then, as now:

- Simplicity and efficiency
- Support the user
- The user should be in control
- Avoid errors

Today most users have many years of computer experience. They may use a PC several hours a day, in addition to other electronic devices such as cell phones, car navigators, TVs, movie systems and digital cameras. While the basic rules are still the same their interpretation may have changed. Today it is as important to accommodate the experienced users as well as the novice. We may want user interfaces that reduce the time to perform a task, for example by avoiding unnecessary input. It may be just a click, but when we perform hundreds, maybe thousands of these a day, avoiding unnecessary clicks become important.

A novice may make an error due to a misunderstanding of the user interface, while an experienced player may perform as many just by trying to be efficient. In both cases it is important that the interface protects the user. Not by generating a long sequence of warnings, but by collecting all the available information and intelligently warn the user whenever there is a possibility of an error.

In many cases the best interface may be no interface. A hundred years ago it was necessary to open the hood of the car just to get it started, and the driver had to go through a set of steps to get the engine running. Today we turn a key or push a button. Connoisseurs may feel that something is lost, but the automation makes driving simpler for most of us. However, the danger of automation is that the user may lose control. The system may perform actions that go contrary to the objective of the user, that leaves him perplexed and that may be the source of errors.

All these aspects of user interfaces are well understood. They have been in the user interface textbooks for many years (Schneiderman, 1987; Morville and Rosenfeld 1998; Nielsen, 1999). With the PC and later, the World Wide Web, the interfaces have been “tested” on millions of users. The advantages of good user interfaces have been firmly established, not only to get satisfied users, but also to create revenue. Apple Corporation is a good example. Instead of developing new technology they have utilized ideas and products that are already in the marketplace to offer excellent user interfaces. Amazon with their “one-click” philosophy is another best-practices case.

Today we should therefore demand that user interfaces all keep up to state-of-the-art, that they offer simple solutions to tasks, automate what can be automated and let the users be in control. Alas, this is not always the case. While there are some good user interfaces out there, many if not most, have faults. In some cases these only require the users to give unnecessary input, lead them down back alleys, select the wrong options, give warnings that cannot be understood, that is, waste the user’s time and be a source of irritation. In the more serious cases errors are introduced.

In this paper two cases of bad user interfaces are presented, together with an analysis of each case based on the basic rules of usability. The paper is anecdotic in its form, but the cases selected will present fundamental problems with the way we address the users.

2. A banking system to default

In 2007 Grete Fossbakk offered to lend her daughter 500,000 NOK, approx 60,000 euros. She used her internet bank to transfer the amount, but while keying her daughter’s account number she typed a digit to many. Instead of typing 71581555022 she typed 715815555022, i.e., an additional 5. Since bank account numbers in Norway have eleven digits this is an illegal number. But instead of giving an error message, the system simply ignored the last digit. There is a checksum control in the account number, but this did not catch the error (in the cases, as here, where more than two digits are wrong the efficiency of the checksum algorithm is only 92 %). Unfortunately for her someone had the account 71581555502. With ten digits for the account number (one digit is used for the checksum), only a fraction of the available numbers is used for actual accounts. Still unlucky, this account was owned by an immigrant living on social security that decided to use the funds (in Norway, most account owners would have returned the amount). In just a few days he managed to gamble away most of the 500,000. When he was caught by the police there were only 100,000 left. He got a nine month prison sentence for using funds that were not his own, but this did not help Grete Fossbakk.

She tried to get the money back from the bank but they told her that this was her mistake. She had not only typed an incorrect number but had also confirmed the transaction. Fossbakk took the case to the bank complaint board, which has two consumer representatives, two from the banks and a chairman, a professor of law. Here she lost by a three to two vote. She got the support of the consumer representatives but the chairman voted with the bank representatives. The only option left was to go to court. This may be expensive in Norway; if one loses there is a chance that one may have to cover the legal expenses also for the other part. But Norway

has an active state supported consumer organization (the Forbrukerrådet) that offered to finance the case. At this point the author was invited by them to be an expert witness. From my point of view the case was simple, if she had typed a digit too many the fault was in the user interface, if she had typed only eleven digits it was her mistake and responsibility. The problem being that we did not have any information on what happened. The user interface only accepted eleven digits in the account number field, thus there was no log information.

From a logical standpoint she had only made one error if she inserted an extra 5, but two if she also had forgotten a 2 at the end. However, the bank refuted this argument. They argued that there was no proof of her having typed twelve digits (somewhat bizarre, since it was their interface that did not collect the log data), but – in any case – she had confirmed the transfer to the wrong account.

In order to recreate what had happened we developed a simulator that mimicked the bank interface. 69 students were then asked to enter up to 30 transactions each; a total of 1800 after some outliers had been removed. Contrary to the real system the simulator collected all data. Analyzing these we found that Fossbakk's error, that of inserting an extra digit in a sequence of similar digits, was the most common error, followed by the error of leaving out a digit in such a sequence (for details see Olsen, 2008a, 2008b). In not one of the 1800 transactions had someone managed to type an additional digit in a sequence and at the same time overlooked a digit at the end. We could therefore prove, with a high probability, that Fossbakk had typed twelve digits. In three cases from the test, the checksum algorithm that the bank relied on did not catch the error of inserting an extra digit – just as in Fossbakk's case.

What then about the confirm phase? The bank interface, as well as the simulator, presented the final transaction on a form, with all the data that the user had typed. In Fossbakk's case she now had the chance to correct her error. She did not, but neither did the students! Very few found typing errors in this phase. To stress this point the simulator, on every tenth transaction, changed the correct typed account number to a similar, but incorrect number. This was done in 178 cases; in only five of these did a student catch the error. That is, we had proved that the confirm phase offered very little protection against errors. We check when we type, one digit at a time, and if the number *looks* right we confirm the final eleven digit number.

When we asked the bank lawyers about data on usability testing, they returned with data on a market research survey. Sure, if we go to Wikipedia to learn more about usability testing we will find a paragraph on market research, but under the section “What usability testing is not.” That is, not only had the bank skipped usability testing, but they also lacked an understanding of what this concept implied. Note that this was an interface that was used by 40 % of Norwegian banks since year 2000, and many foreign banks have been using similar systems. This gave ammunition to a case of gross negligence. Probably the bank also saw that this could be the outcome. They gave up two days before the case was to be tried in court, returned Fossbakk all the money that she had lost, with interest.

What can we learn? Apparently even large banks are satisfied when the user interface *seems* to work. Skipping usability testing of an interface is as if car makers should produce cars without checking the brakes. They may have skipped testing in order to save time and money, but it is more probable that they never even considered testing their interface on real users. Perhaps it takes more than fifty years for a profession to mature? While we had hoped to take this case to the courts to make a legal precedence, the fact that the bank gave in is also of legal significance. Hopefully it will tell institutions that they are responsible for running systems that work in practice, that is, when we consider the complete system consisting of both the

technical parts and the user. Today, with new laws and regulations for universal access, it should be even easier to prove that such an interface lacked the required tolerance for errors.

3. The worst system ever?

Many systems have millions of users; they may be commercial and are used because they offer good services, such as banking or booking, or because they offer pleasure, such as YouTube, Facebook and online games. As we have seen these systems may have faults in the user interface, however, overall they must satisfy users. If not, the user would go to the competition. However, there exists another type of systems where the user cannot go away. This is systems that are used internally in organizations; the users are employees and are required to utilize the systems. Then there is no limit to how terrible the system can be.

The case presented here is a fairly new system, Employee Self Service (ESS), which has been developed by the Norwegian Government Agency for Financial Management (Senter for statlig økonomistyring) to be used in all state organizations. The idea is that the system should encompass all administrative and personal functions needed by an employee, such travel allowance, salary, time registration, leave and vacation. We shall focus on travel allowance.

Previously this was paper based. All travel expenses were registered on a special form, receipts were attached, and the form was signed and delivered to the administration. It was then checked for consistency with travel regulations, and eventually the amount was returned to the employee. For a national flight, with some buses and taxis, some nights at a hotel, etc. one would use between five and ten minutes to fill out the form. But the paper version had some drawbacks. You would have to get the printed form, do the adding manually and find the correct amount for some regulation defined values, for example allowance for food in the cases were one used the standard amounts instead of presenting receipts. A spreadsheet based system automated the adding part, could also save general data (such as name, institution, account number) and in later versions would also be updated with the standard allowances. Then the time used to fill out the form would be closer to five than to ten minutes, somewhat more for international travel.

Now the ESS system has been introduced in all state owned colleges in Norway. All employees had to participate in a two hour user course. There was no mention of what one wanted to achieve with the system, but according to the Web site the idea is to give a better “overview of the work situation”, and that the system can liberate administrative personnel to do other tasks. Well, as a professor my work situation consists of research and teaching and ESS does not cover these parts, thus the “overview” goal cannot be achieved. I would be delighted if the administration could be reduced by introducing more efficient systems, but I do not accept if this just implies that the work load is moved to the professors.

In my college the system has been used for two years. The data shows that the time used to fill out the travel form has increased to at least thirty minutes for a simple trip, much more for complex trips. We now have more personnel involved in tasks connected with travel allowance in the administration than at any time before. One reason is that many employees need help in order to fill out the forms, even after two years of experience.

This raises two interesting questions. Why did the system fail and why do we still use it?

3.1. Why did the system fail?

ESS is based on a SAP platform (the most used enterprise resource planning system). It is run on central servers and is accessed using a Web browser. It works best with Internet Explorer, but also handles Firefox, even if commands and menus are then placed in strange places. The

employee meets problems from the very start. ESS uses a proprietary user name, different from the one that we use on the college system. It also has a proprietary password. This must include digits, letters, special characters, must be eight characters long and be changed every three months. That is, a password that no one can remember. Thus, most users have the password on a yellow tag stuck to their computer display. It is interesting to note that if you forget the password they will send you a new using email. Thus the security level is not higher than what we have on the college computer system (even lower, as the college avoids sending passwords by email).

If one is lucky the login process will not take more than a couple of minutes, but the system is often not running (by far the most frequent emails I get is the one that tells me that ESS down). Inside ESS we will find the travel allowance form, which looks just as the old spreadsheet form. However, it has several peculiarities. Terminology in the system is strange, instead of salary (“lønn”) they use the term pay (“Betaling”). Instead of asking for the reason for a trip (“formål”) they name the field travel-cause (“årsak”). This makes it difficult for novice or low-frequent users to navigate in the system. Error messages appear in a small line at the bottom of the screen, and are ignored by most users. The reason for such a modest presentation may be that most error messages will, in any case, be incomprehensible for the ordinary user.

Information architecture in ESS is terrible. As with the paper form it is based on the idea that the user should fill in information on all expenses, dates, type of expense and amount. Normally, when a full form is presented on the screen, a user would think that she can fill out the form as she chooses. Not so with ESS, you have to start at the top and work your way down. If not, the system will seemingly break down. Then, for some reason or other, you have to wait several hours before you are allowed to log in again. That is, they make an error and punish you for it!

ESS presents the overall menu at all times. This implies that you, while filling out the travel form, can move to other parts of the system just by clicking a button. This is functionality that we expect in a modern application, noting that the users may perform more than one task at the same time - an implementation of the idea of universal access. However, in ESS you can never get back to the travel form again if you move away. No warning is given, it just assumes that you have changed your mind and want to disregard everything that has been entered until now.

ESS implements efficiency in a strange way, forcing the user to work at its phase. In the first version the timeout was set to two minutes! This has been changed to ten in later versions. If you are passive for a longer period, the system will automatically terminate the session or log you off without saving anything! That is, when working with ESS we have to close doors and never take the phone, barring out colleagues and students. That prioritizing – administrative task come first.

It now becomes quite clear that the developers of this system have no idea of how business trips are performed. These may come in all variants. For example, one may combine private and business travel, perform tasks for more than one institution, and recursively, perform a new trip while being on a trip. To handle such a degree of flexibility one needs a very open system where the user is in full control. For example, the user may want to divide the final amount of compensation in two where the expenses are to be split between two institutions. But ESS tries to formalize and control the task, and no user manipulation is allowed – not even reducing amounts. The time you are away are computed automatically, with no possibility of removing additional days used for private purposes (tempting). Amounts in other currencies are computed automatically into Norwegian kroner based on the current

exchange rate, ignoring the possibility that the expense have occurred at quite different times at quite different exchange rates, neither do they include the overhead added by the credit card company. In one situation, where I tried to set a different exchange rate for a conference fee paid in dollars several months in advance, I got the error “More than 10 % difference in exchange rate.” It may be a problem that the value of the dollar decreases, but I am not responsible.

As with all other systems where the formalization does not cover all situations one has to bend the system into describing reality. For example, instead of dividing the total amount by two and send a copy to each institution, we have to make up two different trips and try to adjust expenses so that each institution pays what’s due. This implies that the dates must be fixed, if not one will get reimbursed twice for some expenses (also tempting). A friend that had a one month sabbatical at another institution went of to a conference while saying there and also attended a fully paid overnight institutional seminar had to make this into six different trips to get it through ESS. On paper, one form would have been enough.

The finished form is sent electronically to the administration. So, this is where we get an advantage? No, all receipts have to be glued to A4-pages and put in a mail box. The administration then have to connect the electronic form with the paper receipts, so very little is won at this side. In fact, since users have problems understanding the system most forms will have to be returned at least once to the user to be corrected. Many users will also require expert help. Due to the timeout the users that need help has learned to ask for this before they start filling out the form, requiring access to the expert until finished After two years we should expect users to be more experienced, but many travel rarely or may have very few international trips. With months or years since one used the system the last time, the tricks we learnt the first time are not remembered.

Thus ESS fails because the developers have not understood the task involved, neither the user’s work situation. In addition, it has a very poor user interface. While there have been some attempts to remove the worst faults, a system such as ESS can never work. The reason is that it is based on the fill-out-a-form idea. If users have to type in information on all expenses ESS can never beat the paper form or the simple spreadsheet variant.

3.2. Why do we still use it?

This is a very interesting question, especially as private businesses around us are now moving to better systems, i.e. to systems where much of the data entry is avoided by retrieving expense data directly from the credit card companies. Then it becomes possible to create a “one click” expense form, perhaps needing some editing in a few cases. But we use ESS even if it has a performance record that cannot even be compared to a simple paper based system. ESS violates the most basic rules for universal access. It does not work with screen readers; it does not support the users in performing the tasks, it is inefficient and are not robust with regard to errors.

At best this is due to incompetence on all levels, from the persons that defined the needs for the system to the actual programmers and the college administration that enforce us to use a system that does not work. At worst ESS may be an example of increased bureaucracy in colleges, universities and other state organizations. That is, the administrative parts are seen as the most important, other functions are not prioritized. It becomes more vital to follow rules and regulations than to perform quality teaching and research. And, of course, it is not easy to admit failure and scrap a million dollar system. However, the bad user interface cannot be blamed on anybody but the developers, and may therefore be an indirect criticism

of the institutions that have educated the programmers (us!), that is, assuming that they have been educated in the first place.

When calling ESS the worst system ever I have been accused of exaggerating. There are, the critics say, many systems that don't even run due to technical errors. But that would have been so much better!

4. Conclusion

Two cases have been presented:

- a banking system that has not been submitted to usability testing, causing a serious error where one customer risked the loss of a large amount of money and where another got a nine month prison sentence.
- an employee self service system (ESS) where the basic tasks have been misunderstood, and that in addition, has an unnecessary complex error-prone interface.

Both systems have large user groups and both show disrespect for their users. We expect that books are proofread before publication, that industrial products are tested, that is, as consumers we at least expect that all organizations have good quality assuring routines. But software seems to be different. Many systems, such as the cases presented here, have faults that could have been found by very simple usability testing. One can assume that the problem is lack of professionalism at all levels. This again may be caused by low quality educational programs in informatics, or, perhaps more to the point, by uneducated system developers. But we also find examples of systems, such as ESS, where the basic model is wrong. As we have seen, this may again be due to a lack of understanding of the problems to solve, i.e., by a lack of professionalism. But there may also be the possibility that the system is based on a very different model to what the users have of their work, for example, that administrative functions are the most important.

References

Morville, P., Rosenfeld, L. (1998) *Information Architecture for the World Wide Web*, O'Reilly.

Nielsen, J. (1999) *Designing Web Usability*, New Riders Publishing

Olsen, K.A. (2008a). Customer Errors in Internet Banking, Proceedings Norsk Informatikkonferanse, Tapir forlag

Olsen, K.A. (2008b). A \$100,000 keying error, *IEEE Computer*, April, vol. 41, No. 4.

Schneiderman, B. (1987) *Designing the User Interface: Strategies for Effective Human-Computer Interaction*, Addison-Wesley Publishers.