The $100,000 Keying Error

Kai A. Olsen, Molde University College and University of Bergen

Losing $100K hurts, but other input mistakes can cost much more.

Although it sounds disastrous, making a $100,000 mistake seems relatively minor when stockbrokers have lost millions by hitting the wrong key. The following case proved to be different, however.

An ordinary bank customer, Grete Fossbakk, used Internet banking to transfer a large amount to her daughter. She keyed one digit too many into the account number field, however, inadvertently sending the money to an unknown person. This individual managed to gamble away much of the sum before police confiscated the remainder.

Subsequently, the case received extensive media coverage in Norway. The Minister of Finance criticized the bank's user interface and requested new and improved Internet banking regulations. Suddenly, the risk to Internet banking had become apparent to both the government and ordinary citizens.

Clearly, the user made a slip. She also had the chance to correct the typo before she hit the confirm button. However, as we shall see, the system also had every opportunity to catch her mistake. Yet this did not happen. The system's developers had neglected to build in a simple check that would detect if the correct input were missing.

This case raises questions about what the minimum validation procedures from a banking system developed for ordinary users should be. It also challenges us, as system designers, to help users avoid such errors.

Today's users operate alone in front of a computer, with intermediates and colleagues replaced by computer systems. This new reality makes it important to have interfaces that can offer as good as—or even better—error detection than that found in previous manual systems.

FOSSBAKK CASE

The Fossbakk case provides an illuminating example. The Internet system she employed when making her fatal mistake was one common to a large group of Norwegian banks. Reviewing this case provides insight into the types of typos that users make, the psychology behind "confirmation," and the pitfalls inherent in many Web-based transactions systems.

Fossbakk's daughter's account number was 71581555022, but she inserted an extra 5 and keyed in 715815555022. The user interface accepted only 11 digits in this field (the standard length of a Norwegian account number), thus truncating the number to 71581555502. The last digit is a checksum based on a modulo-11 formula. This will detect all single keying errors and errors where two consecutive digits are interchanged. Inserting an extra 5 changed both the ninth and tenth digits.

The average checksum control will catch only 93 percent of the cases in which such errors occur. For Fossbakk, the final eleven-digit number was a legal account number. However, only a small fraction of all legal account numbers are in use. Further, the chance of mistyping the account number so that it benefits a dishonest person without income or assets is overwhelmingly low in a homogeneous country such as Norway. Our user was thus extremely unlucky. The person who received her $100,000 transaction and kept the proceeds has been sentenced to prison, but this does little to help Fossbakk get her money back.

LITIGATION

Fossbakk took the case to the Norwegian Complaints Board for Consumers in Banking. This board deals with disputes between consumers and banks. The board has two representatives for the consumers and two from the banks, with a law professor as chair. In a three-to-two vote, Fossbakk lost. The chair voted for the bank, arguing that "she made an error and has to take responsibility." He also regretted that Norwegian regulations set no limit for a consumer's loss in these cases, as there would have been if Fossbakk had lost her debit card.

Fossbakk is now taking the case to court, backed by the Norwegian Consumer Council. She argues that she typed 12 digits and that the bank

Continued on page 106

Continued from page 108

system should have given an error message in this case, instead of ignoring all typed digits after the first 11. She has acknowledged she would have no case if only 11 digits had been typed. The bank argues that she cannot prove by any measure of probability that she keyed 12 digits. They further state that there cannot be different rules of responsibility depending on the number of digits given. Finally, they stress that she confirmed the $100,000 transaction.

At this point, I was called in as an expert witness for Fossbakk. In my opinion, and I should expect that of most other computing professionals, a system should give an error message when the customer types a too-long number. Clearly, such a test can be inserted with a minimum of effort. In fact, the Financial Supervisory Authority of Norway has required all banks to implement this functionality based on the Fossbakk case.

Reasonably, we could argue that the bank showed negligence when developing the user interface in question. However, can we prove, beyond doubt, that Fossbakk keyed 12 digits? Since any digits beyond 11 were stripped from the HTML form, no information log exists that can tell us what happened.

BANK SIMULATOR

The answer to this case cannot be found in the literature. It seems that researchers lost interest in studying keying errors when keypunches disappeared. We therefore decided to get our own data by implementing an "Internet bank simulator," a simple interface that works similarly to the system Fossbakk used.

Our system consists of two forms. In the first form we entered the data, date, customer identification number, message text, amount, and account number. After hitting the "pay" button on this form, the data appeared in a new form for confirmation, allowing the user to either confirm or edit the displayed information. Students from a college and some high schools, 69 testers altogether, engaged in entered 30 transactions each from a predetermined test set. After removing some outliers, this gave data on 1,778 transactions.

Results

Our student testers got 124 account numbers wrong, 7 percent of the transactions. This error rate is higher than we would expect in a real system. First, since analyzing faults is our initial task, the simulator does not offer any error messages. However, the user must confirm the transaction, just as in the real system, and the count will not include any errors corrected before confirmation.

It seems nonchalant to code software that lacks a detect-too-long number.

Second, the testers enter a large set of transactions. In some cases, the account number for the preceding or following transaction has been used instead. This could happen in real life, but will be much more frequent here since testers enter the transactions from a list. Third, the test situation does not involve any real money. We suspect that users would verify transactions more carefully when using a live system.

While the overall error rate might be higher, there seems to be no reason why the distribution of different error types should be any different from what we would find in a real system—an exception being the case in which an account number is replaced by another from the data set.

In 29 percent of the cases with a wrong account number, the number ran too long. In half the cases where this happened, students made the same error as Fossbakk, inserting an extra digit in a sequence of two or more identical digits. The bank interface's strategy of skipping digits beyond 11 would have given the correct number in 64 percent of the cases. Of the remaining abbreviated numbers, the modulo 11 test captured all but three. That is, of the nearly 1,800 transactions, three (0.2 percent) would have passed the banking interface's error-detection routines. Multiply this by the nearly 200 million Internet transactions performed each year in Norway, and we see that this small percentage hides a massive problem.

In an improved interface, with a "too long" check, along with the modulo 11 routine, all errors made in our test would have been captured—except in cases where someone entered an account number from another transaction in the set.

Analysis of customer identification numbers, also a part of the transaction, showed the same result. Adding an extra digit in a sequence is a normal mistake. Missing a digit in a sequence is also easy. This test found these errors most commonly. If we ignore the error of typing another account number from the set, "too long" errors occur in 41 percent of the cases, "too short" errors in 35 percent, and wrong 11-digit numbers in 24 percent.

Given these statistics, it seems nonchalant to code software that lacks a detect-too-long number. Since none of the people who entered an extra digit or missed one managed to finish with an 11-digit number by making yet another error, we can state with high probability that Fossbakk entered a 12-digit account number.

Confirmation

This leaves us with the argument that she confirmed a $100,000 transfer to the wrong account—as did the students in 124 of our test cases. In addition, for every tenth transaction, the simulator replaced the typed number with a similar-looking number before confirmation. For example, it replaced the number 70581555022 with 70581555502. In only five out of the 178 cases, 2.7 percent, where this was done did the users recognize the error and correct the number.

It appears that most people perform the inspection while keying,

not when the whole number is displayed onscreen. In many ways, this is efficient. While keying, we can concentrate on one digit at a time, and after keying we have a large number. If this *seems* correct, we hit the "confirm" button.

Psychologist Donald A. Norman explains this behavior in his book, *Psychology of Everyday Things* (Basic Books, 1988). Here, a user confirmed deletion of his "most important work." According to Norman, the user confirms the action, not the file name. Thus, the "confirm" part of the transaction, while having some legal implications, has a minimal effect on detecting errors.

ACCOUNTABILITY

Like many new IT applications, Internet banking is effective. As users, we enjoy reduced costs and 24/7 availability. However, transferring real money based on instructions from possibly inexperienced humans who might slip up means we must look to the system for help. Developers must require that it intercept as many errors as possible. If Fossbakk had used the manual system instead—by writing a letter to her bank requesting the transaction—no responsible employee would have removed the 12th digit of the account number in hopes this would correct the error.

We should expect more. The banking system could offer the account owner's name as confirmation when an account number is entered. In cases where this conflicts with privacy issues, a first name or alias could be used. The system could give a warning message whenever a previous pattern is violated. For example, if we pay a utility bill of $100 to $300 each month, we should get a warning if the reported amount is way off in either direction. Further, e-invoices and other automatic procedures can limit the number of transactions that must be keyed in, reducing the overall error rate.

In Fossbakk's case, the banking system erred. Next time, it might be a weapons system or medical information system that fails. Examples from these areas have already revealed misinterpretations between systems and users that caused serious consequences. For all systems, we must, as computer professionals, protect users from their own errors, intercept all detectable errors, and give informative warnings when we believe the user might have made an error. The "she made an error and must take responsibility" defense is too simple. We need systems that work in collaboration with the user such that the overall error rate drops to a minimum. Yes, we need responsible users, but a good system can handle most slips and typos they make, as this case shows.

Kai A. Olsen is a professor at Molde College and the University of Bergen, and an adjunct professor at the University of Pittsburgh. Contact him at Kai.Olsen@hiMolde.no.

Editor: Neville Holmes, School of Computing and Information Systems, University of Tasmania; neville.holmes@utas.edu.au.