Using Checksums to Detect Number Entry Error

Sarah Wiseman, Anna Cox, Duncan Brumby, Sandy Gould & Sarah O’Carroll


PP release date: 5 February 2013

file: WP107.pdf
Using Checksums to Detect Number Entry Error
Sarah Wiseman, Anna L. Cox, Duncan P. Brumby, Sandy J.J. Gould, Sarah O’Carroll

UCL Interaction Centre,
University College London
{s.wiseman, brumby, s.gould}@cs.ucl.ac.uk, anna.cox@ucl.ac.uk, sarah.o’carroll.10@ucl.ac.uk

ABSTRACT
Number entry is a common task in many domains. In safety-critical environments such as air traffic control or on hospital wards, incorrect number entry can have serious harmful consequences. Research has investigated how interface designs can help prevent users from making number entry errors. In this paper, we present an experimental evaluation of two possible interface designs aimed at helping users detect number entry errors using the idea of a checksum: an additional (redundant) number that is related to the to-be-entered numbers in such a way that it is sufficient to verify the correctness of the checksum, as opposed to checking each of the entered numbers. The first interface requires users to check their own work with the help of the checksum; the second requires the user to enter the checksum along with the other numbers so that the system can do the checking. In each case, two numbers needed to be entered, while the third number served as a checksum. With the first interface, users caught only 36% of their errors. The second interface resulted in all errors being caught, but the need to enter the checksum increased entry time by 46%. When participants were allowed to choose between the two interfaces, they chose the second interface in only 12% of the cases. Although these results cannot be generalized to other specific contexts, the results illustrate the strengths and weaknesses of each way of using checksums to catch number entry errors. Hence our study can serve as a starting point for efforts to improve each method.

Author Keywords
Number entry; Checksums; Error detection; Redundancy

ACM Classification Keywords
H.5.2. Information interfaces and presentation (e.g., HCI):
User interfaces – input devices and strategies.

INTRODUCTION
Numbers are entered into devices on a regular basis in many environments, including safety-critical environments. Entry can be done either from memory (as with entering a PIN) or from an external source (such as a medical prescription) – the case we will consider in this paper. Occasionally, harmful errors are made: In one case reported in the medical domain, a patient was given drugs at a rate of 68 mL/h rather than the prescribed 6.8 mL/h, an error which led to the patient’s death [4].

Thimbleby and Cairns [8] have made suggestions about the types of design interventions that can enable incorrectly entered numbers to be recognized as such by the computer into which they are entered (e.g., recognizing common syntactic errors). Other suggestions aimed at reducing number entry errors include changing the way that numbers are input into devices, for instance, by encouraging users to check the numbers they are entering [2,5]. Since none of the available approaches can prevent all types of error, it is worthwhile to explore new approaches.

Any attempt to reduce problems arising from number entry errors in a given practically important domain needs to take into account the specific conditions that prevail when numbers are entered and the other related measures that are taken in that context (see, e.g.,[7] for an extensive survey of these issues in the context of prescribing medication). In this paper, we aim not to solve a particular practical problem but to initiate research on an approach that may ultimately prove to be applicable in a variety of contexts.

POSSIBILITIES FOR USING CHECKSUMS
The process of users entering data into a machine can be likened to the process of sending a signal over a noisy channel. Both involve the transfer of data between two locations, and both are potentially subject to errors occurring during that process. To catch errors in the case of signal processing, error-correcting codes are used. This involves the sender providing redundant information, along with their message. This redundant information can be used to detect if any errors have been made during transmission and can allow for error correction. We can apply this method of error detection to cases where a user has to enter data.
More specifically, the key idea is that of a **checksum**: an additional (redundant) number that is related to the to-be-entered numbers in such a way that it is sufficient to verify the correctness of the checksum, as opposed to checking each of the entered numbers, which has been shown not to be a successful strategy [6]. As was suggested by Thimbleby [9] if a checksum is presented along with the numbers to be entered and if the checksum is also computed on the basis of the numbers that the user has actually entered, it is possible to check the correctness of the numbers simply by comparing the two checksums.

Sometimes, checksums occur naturally, as with infusion pump programming in the medical domain. Infusion pumps are used to administer a drug to a patient intravenously over a period of time. Three values specify each infusion: time, volume to be infused (VTBI) and rate of infusion. The relationship between these values is such that knowing two of them allows the third to be calculated:

\[
\text{VTBI} = \text{Time} \times \text{Rate}
\]

As a result, infusion pumps often require only two of these values to be entered; some may compute and display the third value. Hence the third value can serve as a naturally occurring checksum.

In other cases, it would be necessary to agree on a general procedure for computing an artificial checksum, which must be applied both when the numbers and their checksum are originally prepared for presentation and after the user has entered the numbers.

There are two ways in which a checksum can be used for verification: (a) The user is prompted to compare the original checksum with the computed checksum; or (b) the user is required to enter the checksum along with the other numbers, and the computer does the comparison. (This latter method is somewhat similar to the use of cross-validation in form design where the same piece of information is requested in two different questions to ensure it is correct [3], but it requires less data entry.) In our study, we test both of these methods, with interfaces referred to below as the “2-number interface” and the “3-number interface”, respectively.

We hypothesise that participants will complete number entry tasks with the 2-number interface more quickly (because fewer key presses are required); but that entry will be more accurate with the 3-number interface, which makes it almost impossible not to notice an error that has been made and which does not rely on the correctness of visual comparison of numbers by the participants.

**METHOD**

**Participants:** Twenty-four participants (13 male) were recruited from a university subject pool. The participants had no previous experience with medical devices. The mean age of participants was 34.7 years (range 21-59, SD=10.8). All were experienced computer users who either had normal or corrected to normal vision.

**Design:** The experiment was a within-subjects design and the order in which the interfaces were presented was counterbalanced. The independent variables were the interface type and the instruction. There were two levels of interface type: the 2-number and the 3-number interface.

The 2-number interface required participants to enter two numbers (e.g., VTBI and rate). It then calculated the third number (e.g., time) and prompted the participant to compare this “checksum” with the third of the presented numbers.

The 3-number interface asked the participant to enter all three numbers; the computer then checked whether the third number was related to the first two in the expected way and displayed the feedback “correct” or “incorrect”.

With both interfaces, if a participant noticed that they had made an error, they could correct it before moving on.

The instructions given to the participants were either to perform the task as quickly as possible or as accurately as possible. The dependent variables for this experiment were the number of errors and the time taken to complete trials.

**Materials:** The experiment was conducted using a desktop PC. The participants were provided with a standard computer keyboard to type on, and were asked to use the numeric keypad at the right of the keyboard. Navigation through the interface was done using the tab and arrow keys.

**Procedure:** The task was to enter a series of numbers, either as quickly or as accurately as possible (this variation was counterbalanced throughout, with participants using each interface with each condition). Data to be input was presented in columns of VTBI, Rate and Time. Each row represented one ‘prescription’ that would have to be entered before submitting (see Figure 1).

The experiment was split into two parts: a no-choice phase and a choice phase [1]. In the no-choice phase of the experiment, participants were given six sets of 20 numbers to enter on each of the two interfaces; the same number set was given to all participants of the experiment. The choice phase of the experiment aimed to assess which of the interfaces the participants preferred, depending on whether they were told to enter numbers quickly or accurately: Participants were given the choice of interfaces for entering a single set of 20 prescriptions in the “fast” instruction and were again given a choice when entering another set of 20 prescriptions in the “accurate” condition.
We next consider whether there was an effect of interface type on the likelihood that participants would correct number entry errors after being prompted by the system. To do this we calculate the percentage of corrected errors, which is the number of entry errors that were corrected after they were made, divided by the total number of entry errors made. With the 2-number interface, participants noticed only 36.18% (SD=34.58%) of the entry errors when prompted to check for them. With the 3-number interface, 100% of all entry errors were flagged by the system after its checksum computation, and the participants corrected all of these.

**DISCUSSION**

This experiment illustrates the strengths and limitations of two ways in which number entry errors can be caught with the help of a verification process based on a checksum.

In our study, participants using the 2-number interface failed to detect 64% of the entry errors that they had made, even though the checksum presentation reduced the verification step to a simple comparison of two new and not previously typed numbers.

The 3-number interface led to all entry errors being corrected, but it required 46% more time; and when given a choice the participants almost always preferred the faster 2-number interface.

---

**RESULTS**

The measures taken during this experiment were the time taken to enter each prescription and the proportion of initial number entry errors made. For statistical analysis we use a dependent t-test with a .05 significance level. Data from one participant was corrupted and so is removed from the analysis. Three participants did not complete the choice section of the experiment. It was found that the instruction given to the participants to either perform the task as quickly or as accurately as possible during the no-choice sections had no significant effect on error rate or time.

**No-Choice Phase: Performance Profiles**

**Time**

The mean time taken for each participant to complete the task was significantly shorter for the 2-number interface (M=7.6 s, SD=2.6 s) than for the 3-number interface (M=11.1 s, SD=4.1 s), t(22)=5.3, p<.001. The increase in time of 46% for the 3-number interface is understandable in that it requires the participant to enter 3 rather than 2 numbers (though the 2-number interface calls for an additional visual comparison operation).

**Number Entry Errors**

A single entry error was counted when the participant entered at least one of the (two or three) numbers of a prescription incorrectly. The percentages of entry errors in the 2-number interface condition (M=5.14%, SD=3.98%) and the 3-number interface condition (M=6.67%, SD=2.93%) were not significantly different, t(22)=1.565, p =.132.

We next consider whether there was an effect of interface type on the likelihood that participants would correct number incorrectly entered at least one of the (two or three) numbers of a prescription incorrectly.

A single number (though the 2-number interface calls for an additional visual comparison operation).

**Choice Phase: Participant Preference**

For the final two sets of numbers, which were presented under the “fast” and “accurate” conditions, respectively, in that order, participants were given a choice of interface to use. In 37 of the total of 42 choice trials that were completed participants, preferred the 2-number interface; there was no reliable difference in choices between the “fast” and “accurate” conditions.
One factor affecting these results might be the experimental setting: participants were not penalised in any meaningful way for entering inaccurate information. In situations where the results of incorrect number entry were more costly, it is possible that users may pay more attention to the information provided by the checksum in the 2-number interface – or be more willing to use the slower 3-number interface. Another feature of the experiment that may not generalise to some workplace situations is that participants were asked to enter many sets of numbers. For example, when infusion pumps are programmed on the hospital ward, only one set of numbers for a single prescription is entered at a single time. Finally, the computer’s numeric keypad used in this study is different from some commonly used input devices, such as an infusion pump keypad.

CONCLUSIONS

Despite these considerations, it is reasonable to expect that the basic drawbacks of each method will be found to some degree in realistic contexts. It is therefore worthwhile to consider how these drawbacks can be minimized.

When checksums are to be compared by the user (as in our 2-number interface), it should be possible to make this comparison easier than it was in our interface. For example, the checksum could be displayed in a consistent, distinctive way both in the original prescription and in the computer’s feedback.

When the user is required to enter the checksum (as in our 3-number interface), the goal is to minimize the extra effort that is required. In contrast to our naturally occurring checksums, an artificially computed checksum does not have to have as many digits (or characters) as the numbers that are being entered; it might even be a word rather than a number.

The relationship of the benefits to the costs of using a checksum, with either of these two methods, should be greater when more than two numbers and/or longer numbers are to be entered. Even for a set of 10 multidigit numbers, only a single checksum is needed for verification. Therefore, the checksum approach may be more attractive in contexts where more complex sets of numbers are to be entered.

ACKNOWLEDGEMENTS

Anna Cox, Duncan Brumby and Sandy Gould are funded, by the CHI+MED project EPSRC grant EP/G059063/1. In particular we thank Anthony Jameson for his significant contribution to improving this work. We also thank members of the UCL Interaction Centre for useful comments on an earlier draft.

REFERENCES


