Issues in number entry user interface styles: Recommendations for mitigation

[Invited Paper]

Patrick Oladimeji, Harold Thimbleby
Swansea University
College of Science
SA2 8PP, Wales
p.oladimeji@swansea.ac.uk,
harold@thimbleby.net

Paolo Masci, Paul Curzon
Queen Mary University of London
Mile End Road
E1 4NS, London
p.m.masci, p.curzon@qmul.ac.uk

ABSTRACT

Interacting with numbers is a core part of using many interactive computer systems from the remote controls of electronic media appliances to user interfaces of high-integrity systems such as medical devices. Number entry systems are widely used on mobile devices. A wide variety of different user interface designs exist for interacting with numbers. The intricacies of the different styles are not well understood by designers and developers, especially for handling use error. This paper reviews these issues and provides recommendations for mitigating them.

Categories and Subject Descriptors

H.5.2 [**User Interfaces**]: Input devices

1. INTRODUCTION

There are a variety of ways to interact with numbers, not least because of the wide variety of input devices available (e.g., Buxton [1], Card *et al.* [2] and Mackinlay *et al.* [7]). The ease with which the different properties, movement, pressure, etc, sensed by these input devices can be mapped to numeric quantities (e.g., a slider specifies position while a dial specifies angle) also adds to the diversity of design options. To structure the design space of number entry interfaces, we use the *Questions, Options and Criteria* (QOC) [8] method.

2. THE DESIGN SPACE

QOC facilitates a structured exploration of the design space, which can be expanded to an arbitrary level of detail. QOC enables reasoning about design options at many levels of abstraction, which is useful for people with different roles in the design process.

The first question that helps structure the design space addresses *what part of a number the interface controls*. The user interface might be used to control individual digits of the number or to control a selection mechanism for choosing the intended number from a set of valid values. When the interface controls individual digits, the question of *the order of digit specification* can be asked. For this, there are three options. The digits may be specified in an *unrestricted* order or in a restricted order such as from *left to right*, or from *right to left*.

For each of these options, it is possible to explore methods for selecting digits in the number in order to further refine the design space. Hence we address the question of *how the digits are specified*. The digits can be specified directly using, for example, handwriting, widgets with a 1:1 mapping to each numeral (e.g., a digit key) or by using widgets that have a 1:many mapping to numerals (e.g., a knob with 10 position).

When the interface controls how a number is selected directly, we can address the question of *how the numbers are selected*. Similar to digit selection, the number can be selected directly, using widgets with a 1:1 mapping to the intended number (e.g., as seen on elevator panels) or the number might be selected by using widgets that helps the user to navigate a much larger set of valid values, e.g., using a dial or a pair of buttons for traveling up and down the number line.

Figure 1 shows a decomposition of the design space. We describe four groups of interface styles from this classification with some examples from day to day use.

2.1 Serial digit entry

This allows the user to enter the digits of a number in a restricted order (either from left to right or from right to left). The most common example of this is the numeric keypad which allows direct digit entry in left to right order. This can be found in calculators and telephones. Note that digit selection on a serial interface need not be by direct selection. Digits can be set incrementally using up-down arrows or by chording (pressing multiple keys) as found on old mechanical calculators like the Kollektor [9].

2.2 Independent digit entry

Independent digit entry interfaces allow the user to specify the digits in a number in unrestricted order. Examples include the directional-pad (D-pad) interface (figure 2c), a four way navigation style keypad where the up-down keys are used to increase or decrease the digits and the left-right keys are used to select a digit to modify. Another example,

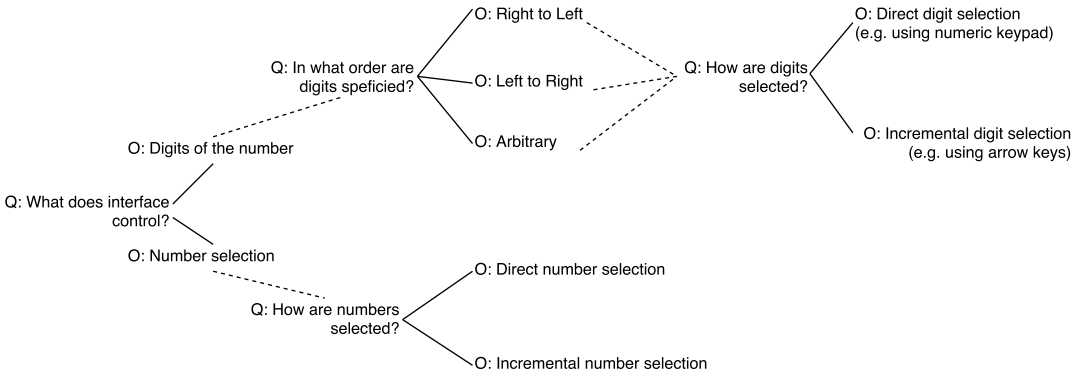


Figure 1: The QOC analysis segments the design space of number entry systems into two main dimensions. Some interfaces allow the explicit specification of digits in a number, while others allow selection of a number from a list of valid values. Digit based interfaces are further distinguished based on the order of digit specification and how digits are selected (e.g., direct selection or incremental selection), while number selection based interfaces are structured based on how numbers are selected (e.g., direct selection or incremental selection).

the up-down interface, has a pair of widgets for increasing and decreasing each digit that can be displayed by the host application. A variation on the up-down interface is shown in figure 2b where each digit is modified independently by swiping up or down on a digit.

2.3 Incremental number entry

Here the user navigates forward or backwards through a list of numbers. The interface may allow the user to control the speed of navigating through the list. In many designs, tapping a button goes to the next number, and holding it gradually (or after a pause) increases the speed of search.

2.4 Direct number selection

Here the user makes a selection from a valid set of options presented by the application. This is feasible in situations where there is a small and stable set of numeric values to choose from, e.g., as seen in lift floor selection interfaces. If the range of numeric values required by the host application is large enough to require scrolling between screens, then this interface essentially becomes an *incremental* interface.

Any of the interfaces described above may be implemented using buttons, sliders, dials, gestures or any widgets found in the design space of input devices [7]. The smallest unit of a number is a digit, which itself is a number. A sequence of digits that are a part of a number are also numbers. This means, for instance, that the controlling aspect of the interfaces described here is not limited to single digits and entire numbers. The controls may be applied to digits that are grouped together in twos, threes or more clusters and these controls may be performed incrementally or directly.

3. CRITERIA

Below we present criteria that can be used as part of the decision process of selecting an appropriate interface from the design space for a given context.

3.1 Error correction rate

This refers to the proportion of all errors that occur on an interface that are corrected by the user. It gives an indication of how well errors are detected on each interface. For instance, results from our lab studies [10, 11] showed that 83% of errors on the serial interface tested were corrected. 94% of errors were corrected on both version of the incremental interface tested. On the independent digit entry interfaces, the *up-down* interface had 84% of errors corrected while the *d-pad* had 100% of errors corrected.

3.2 Error severity

Error severity provides a quantitative measure for assessing the level of risk that can be attributed to an undetected error on an interface. More formally, it is defined as the ratio of the intended value to the transcribed value in a number entry task [3, 10, 14]. This criterion can be used to explore the cost of undetected overrun, substitution, deletion or insertion errors in a number entry task.

3.3 User interface footprint

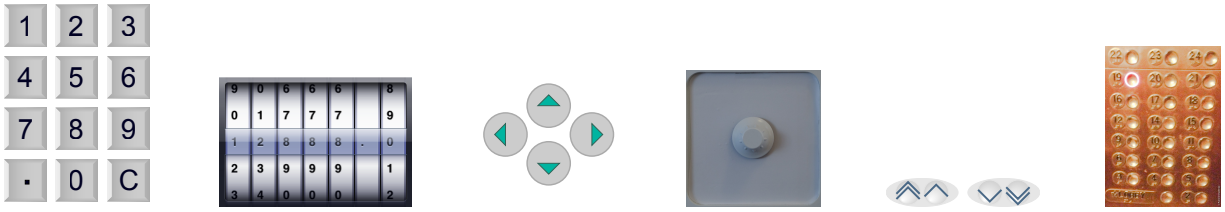
This is the amount of space needed to implement an interface. This includes issues such as the number of widgets required, the minimum size of each widget in order to facilitate optimal target acquisitions and the amount of space needed to meaningfully implement and execute interaction modalities such as gestures if using a touch screen interface.

3.4 Speed of entry

This refers to how quickly a user can enter or select an intended number on a given interface. The speed of entry would depend on the range of values and the precision (i.e., the number of decimal places) of the values allowed in the application. Where numbers of unbounded length and precision need to be entered, a reasonable option might be to implement a serial digit entry interface. This same assumption should not be carried over to contexts where the type of numbers allowed in the application are bounded, with some values more common than others. An example of this is in numbers used in infusion pumps and other medical devices. Our research shows that on average, the up-down independent digit entry interface is faster than the numeric keypad found on many infusion pumps [9]. Based on the analyses of numbers used in infusion pumps in hospitals, research by Wiseman et al. shows that optimising the user interface to address more frequently used digits in the required context would improve the speed of entry and could reduce number entry error [16].

4. ISSUES

Each group in the classification comes with choices that



(a) Number pad (b) Numeric wheels (c) D-pad (d) Dial (e) Chevrons (f) Lift

Figure 2: Serial digit entry (a), independent digit entry (b) and (c), incremental number entry (d) and (e) and direct number entry (f).

designers have to make when implementing user interfaces from that group. These have an impact on the correctness of the overall application. We highlight some common issues below and give recommendations to mitigate them:

4.1 Serial digit entry

The nature of interfaces based on serial digit entry means they are prone to syntax errors that violate the rules of a properly formed number. For instance, what happens when a person keys in multiple decimal points, or a number too long to be displayed in the application? How are detected errors corrected? and how are users alerted to error?

4.1.1 Blocking and alerting users to error

Many devices, such as calculators and infusion pumps, just ignore the user pressing the decimal point more than once. This behaviour means that if a person tries to correct the key presses [0 • • 7 5], by trying to delete a decimal point [0 • • DEL 7 5], they could turn it into 75 rather than 0 • 75. If the second decimal point was just silently ignored then the attempted correction would delete only the one registered instead. Does this really make a difference? We studied a variety of designs with computer simulated user key pressing to see what effect decimal point handling has on the accuracy of numbers. We found that correctly handling decimal points can at least halve the rate of 'out by ten' errors, where a person enters a number that is ten or more times larger or smaller than that intended [14]. This can be achieved by registering all keys the user presses (including multiple decimal points) on the display and by blocking and alerting the user to the entry of numbers that violate the international guidelines for formatting numbers of the Institute for Safe Medication Practices. Few medical devices do this, yet it would be easy to fix them — and normal (error free) use would be unaffected.

There is a related problem with some numeric keypads, like calculators, such as those which have one key for each digit. Commonly, the devices only have space to display numbers with at most 8 digits. The delete key behaves unexpectedly because applications often 'ignore' digit keys pressed after the display is full. Trying to correct the 9 digit number 123456786 to 123456789 with the sequence of keys [1 2 3 4 5 6 7 8 6 DEL 9] could delete the 8 if the last 6 had just been ignored. This would turn it into 12345679.

Overall the important design lesson here is that devices should always alert users when they have made a mistake, like entering too many digits or multiple decimal points [12], rather than silently ignoring the flawed key press. Otherwise the person may not notice their mistake, or not notice that it has been silently corrected and then mistakenly try to correct it.

4.2 Independent digit entry

4.2.1 Controlling digits

Modification of digits on an interface allowing direct control of digits is usually done using UP and DOWN arrow keys. How should these keys work? For example, pressing UP will increase 1 to 2, 2 to 3, 3 to 4 and so on, but what happens when 9 is increased? Does it go to 0 or does it go both to 0 and increase the digit to its left (does 9 "increase" to 0 or to 10?). Perhaps it stays at 9? What happens when users attempt to change numbers beyond the maximum or minimum values the device can display?

4.2.2 Controlling cursors

Instead of having UP and DOWN keys that correspond to each digit on the display, some interfaces use LEFT and RIGHT arrow keys to move a cursor over the digit that should be modified. What happens if you press LEFT when the cursor is in the leftmost position, or press RIGHT when in the rightmost position? Where should the cursor be initially placed when entering a number?

We studied combinations of design choices from a set of common features like these found in designs [4]. We inserted simple, common keying errors such as pressing a key twice or missing a key press at random. We looked at each design decision separately in terms of whether it was included or not. We found the safest choices in terms of which is least likely to lead to a large error in the sense of being least sensitive to simple keying slips are:

- when users attempt to increase or decrease numbers beyond the maximum or minimum values allowed on the device they are stopped from doing so;
- the cursor starts on the leftmost digit, and
- when moving left or right, the cursor does not jump from one end of the display to the other, but instead the cursor stays where it is.

4.3 Incremental number entry

Incremental number entry interfaces can be slow if they are used to address numbers over a very large range. Designers try to improve the speed by changing the order of control of the widget from a simple position control system to a velocity or acceleration control system — where user action determines the speed at which changes to the number is made. These design decisions mean users are likely to overshoot and undershoot their target values when selecting numbers.

4.3.1 Gain and time-delay

Improving the stability of incremental number entry interfaces requires understanding the roles played by *gain* and *time-delay*. Gain affects the speed at which the user ap-

proaches the target number and time-delay affects the time taken for the system to react to changes made by the user (also referred to as *latency*). When *gain* is low, the speed of entry is slow. When gain is high, the user is likely to oscillate about the target value. There is a similar effect when time-delay is high [5, 6].

Since users tend to look at the displays of this type of interface during interaction, appropriate information should be fed back to the user regarding how much change would happen to a number as a result of interaction. This can be achieved by highlighting digits on the display or using auditory feedback to encode the amount of change occurring on a number. Stability of the system would also be improved if a widget that allows the user to control the gain of the system is used. This way users are able to control exactly how fast they approach the target value.

4.4 Bugs

Numeric user interfaces seem easy to implement, so it is surprising that many have bugs [13–15], perhaps a consequence of programmers thinking number entry is so easy to program they do not think it is worth adopting best practice they would use for problems recognised as being hard.

For example, the delete key on serial number entry user interfaces is often incorrectly implemented [15], and on some independent digit entry systems there are modes where previous numbers interfere with the number the user is entering [4].

An unlimited number of bugs is possible, and any can interact badly with use error, so the recommendations for mitigation are to use formal methods [13] and testing [15], as well as development environments that support tools for rigorous development.

5. SUMMARY

Number entry user interfaces are ubiquitous and their current design and implementation are often not performed in a dependable way. We have shown there is variety in the design of user interfaces for manipulating numbers. With different contexts of use, it is important to choose an interface style that is appropriate. Various constraints might determine this choice — for instance *speed* might be important in gaming user interfaces, but in a medical application *error severity* might be most important, and *user interface footprint* might be important in a mobile application. We have therefore highlighted common issues that arise as a result of the choices designers and developers have to make when implementing number entry user interfaces. We have provided some recommendations on how to mitigate common issues. In general errors should be caught and alerted to the user, and users should be able to accurately predict the effects of their actions on the interface.

Acknowledgements

This work was funded by EPSRC [grant EP/G059063/1].

6. REFERENCES

[1] William Buxton. Lexical and pragmatic considerations of input structures. *SIGGRAPH Comput. Graph.*, 17(1):31–37, January 1983.

[2] Stuart K. Card, Jock D. Mackinlay, and George G. Robertson. The design space of input devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '90, pages 117–124, New York, NY, USA, 1990. ACM.

[3] Abigail Cauchi. Differential formal analysis: evaluating safer 5-key number entry user interface designs. In *Proceedings of the 4th ACM SIGCHI symposium on Engineering interactive computing systems*, EICS '12, pages 317–320, New York, NY, USA, 2012. ACM.

[4] Abigail Cauchi, Andy Gimblett, Harold Thimbleby, Paul Curzon, and Paolo Masci. Safer "5-key" number entry user interfaces using differential formal analysis. In *Proceedings of HCI 2012 The 26th BCS Conference on Human Computer Interaction*, volume 26, pages 29–38, Birmingham, UK, September 2012.

[5] Gavin Doherty and Mieke Massink. Continuous interaction and human control. In *Proceedings of 18th European Conference on Human Decision Making and Manual Control*, pages 80–96, Loughborough, 1999.

[6] Richard J. Jagacinski and John Flach. *Control theory for humans: quantitative approaches to modeling performance*. Routledge, 2003.

[7] Jock Mackinlay, Stuart K. Card, and George G. Robertson. A semantic analysis of the design space of input devices. *Hum.-Comput. Interact.*, 5(2):145–190, June 1990.

[8] Allan MacLean, Richard M. Young, Victoria M.E. Bellotti, and Thomas P. Moran. Questions, options, and criteria: Elements of design space analysis. *Human-Computer Interaction*, 6(3-4):201–250, 1991.

[9] Patrick Oladimeji. Designing number entry user interfaces: a focus on interactive medical devices. PhD Thesis, January 2014.

[10] Patrick Oladimeji, Harold Thimbleby, and Anna Cox. Number entry interfaces and their effects on error detection. In *Proceedings of the 13th IFIP TC 13 international conference on Human-computer interaction*, pages 178 – 185, Berlin, 2011.

[11] Patrick Oladimeji, Harold Thimbleby, and Anna Cox. A performance review of number entry interfaces. In *Proceedings of the 14th IFIP TC13 Conference on Human-Computer Interaction*, pages 365–382, Cape Town, September 2013.

[12] Harold Thimbleby. Contributing to safety and due diligence in safety-critical interactive systems development by generating and analyzing finite state models. In *Proceedings of the 1st ACM SIGCHI symposium on Engineering interactive computing systems*, EICS '09, pages 221–230, New York, 2009.

[13] Harold Thimbleby. Safer user interfaces: A case study in improving number entry. *IEEE Transactions on Software Engineering*, 41:711–729, 2015.

[14] Harold Thimbleby and Paul Cairns. Reducing number entry errors: solving a widespread, serious problem. *Journal of the Royal Society Interface*, 7(51):1429–1439, October 2010.

[15] Harold Thimbleby, Paul Cairns, and Patrick Oladimeji. Unreliable numbers: Error and harm induced by bad design can be reduced by better design. *Journal Royal Society Interface*, in press.

[16] Sarah Wiseman, Anna L Cox, and Duncan P Brumby. Designing devices with the task in mind: which numbers are really used in hospitals? *Human factors*, 55(1):61–74, February 2013.