Designing number entry user interfaces: A focus on interactive medical devices.

Oladimeji, Patrick

Designing Number Entry User Interfaces

a focus on interactive medical devices

Patrick Oladimeji

Submitted to Swansea University in fulfilment
of the requirements for the Degree of Doctor of Philosophy

Prifysgol Abertawe
Swansea University

College of Science
Swansea University

2014

ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 – 1346

To my parents

Contents

Abbreviations

Abbreviation	Description	Definition
ADE	Adverse Drug Event	page 3
ANOVA	Analysis of Variance	page 96
API	Application Programming Interface	page 76
ATM	Automated Teller Machine	page 2
CxC	Correct by Construction	page 31
DERS	Dose Error Reduction Systems	page 33
EEG	Electroencephalography	page 32
FDA	Food and Drug Administration	page 29
ISMP	Institute for Safe Medication Practices	page 31
KLM	Key-stroke Level Model	page 71
KSPC	Key strokes per character	page 20
KVO	Keep Vein Open	page 56
MAUDE	Manufacturer and User Facility Device Experience	page 5
MSD	Minimum String Distance	page 20
NPSA	National Patient Safety Agency	page 4
QOC	Questions Options Criteria	page 38
RFID	Radio Frequency Identification	page 33
SKSS	Stochastic Key Slip Simulation	page 31
VTBI	Volume to be infused	page 33
wpm	words per minute	page 20

Declaration

This work has not been previously accepted in substance for any degree and is not being concurrently submitted in candidature for any degree.

Signed .... (candidate)

Date 02 April 2014

Statement 1

This thesis is the result of my own investigations, except where otherwise stated. Other sources are acknowledged by footnotes giving explicit references. A bibliography is appended.

Signed .. (candidate)

Date 02 April 2014

Statement 2

I hereby give my consent for my thesis, if accepted, to be available for photocopying and for inter-library loan, and for the title and summary to be made available to outside organisations.

Signed .. (candidate)

Date 02 April 2014

Abstract

Number entry is a crucial aspect of using many interactive systems. Tasks such as withdrawing money from an ATM, selecting a TV channel, manually tuning into a radio station, or setting up an infusion pump for drug delivery, all involve entering or selecting numbers. The number entry aspects of these tasks are usually secondary to the user's goal. Users typically have higher level goals which might involve a sub-task related to entering numbers. As a result, number entry is assumed to be simple, straight forward and uninteresting.

The design of number entry interfaces dates back as early as the use of tally sticks and counting boards although modern interfaces did not emerge until the design of the first mechanical calculator in the 17th century. The nature of numbers allows interfaces to be designed that exploit the specification of individual digits of the number as well as making incremental changes to the entire number. The diversity in interface design is not evident in current research which is dominated by various forms of evaluations of the numeric keypad interface.

This thesis undertakes a historical review of the design of number entry interfaces and then explores the design space within which they lie while proposing a classification for the different styles of interfaces. It then evaluates several example alternatives to the numeric keypad, specifically those in use on infusion pumps in hospitals using both exhaustive simulations and usability studies. These evaluations explore the effects of interface styles on error detection, speed, error severity and error type. This research concludes by identifying properties for performing relative comparisons of interfaces and uncovers design trade-offs that will help inform decisions about the safety and dependability of number entry interfaces.

Acknowledgements

I would like to thank Harold, my supervisor for the very helpful and extensive suggestions made through this research. Your seemingly tangential thought processes were an incredible source of inspiration for a great deal of work reported in this thesis. Thanks also to Parisa for the encouragement, advice and recommendations you provided at different stages of this work. I would like to thank Michael for the very useful feedback and sanity checks on draft versions of this thesis and to Anna for helpful suggestions and feedback provided in designing experiments.

Many thanks to my family, especially Wende, for graciously putting up with a lot of early mornings an late nights. I am incredibly grateful to my parents and uncle Ralph, for supporting and encouraging me in all my academic endeavours.

Thank you to all my friends especially Tom, Jen, Simon, Emma, Liam, Ben, Seb and Matt for keeping me in track with reality and empathising with the challenges of the journey that is the PhD. Thanks to Sandy Gould for agreeing on short notice to present the interact paper and to members of the UCLIC group for their generous hospitality during my time there.

Finally, I would like to thank the CHI+MED group especially Paolo, Gerrit, Karen, Abigail, Paul Lee and Carlos, with whom I have had various insightful discussions that affected the path taken in this thesis.

This research was funded as part of CHI+MED*: Multidisciplinary Computer-Human Interaction research for design and safe use of interactive medical devices

*http://www.chi-med.ac.uk

project EPSRC Grant Number EP/G059063/1.

Copyright acknowledgements

Acknowledgement is due to the following, who granted permission to use their images in this thesis:

- *The Computer History Museum (`http://www.computerhistory.org/`)* permitted the use of several images from their online exhibition on Calculators. These images appear in Chapter 2 to illustrate the styles of number entry interfaces used in ancient mechanical calculators.
- *Friedrich Diestelkamp* of `http://www.addiator.de` permitted the use of the image of the Kollektor in Chapter 2.
- Figure 2.5b is originally by R. Sull from Wikipedia. The image is licensed and used under the Creative Commons Attribution-Share Alik3 license.

I created all other images used in this work.

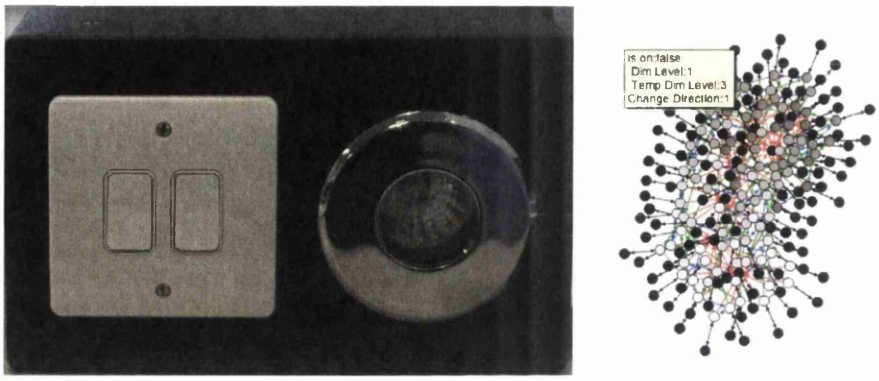
Preface

Light switches are often cited as examples of simple state machines. They are either on or off. A few months before I started this PhD, Harold Thimbleby, my supervisor, showed me his latest toy. It was a light box he'd built using an off-the-shelf dimmable electronic transformer[†] and two generic switches. The switches looked exactly identical but had different physical properties when pressed. The first switch, on the left, was a normal toggle switch and the other was spring loaded, returning to an *off* state once released. Harold challenged me to turn on the light box. After about half a minute of probing the light box, I managed to turn it on. But why should such a simple device have a complicated interface? I later learnt that the light box was a replica of the light control our postgraduate seminar room.

I had just finished my master's degree and was at the time doing some work on automatically exploring interaction graphs of simple interactive systems from programmed simulations. So I thought I'd find out exactly how complicated the box was by writing an ActionScript simulation and discovering the user interface model. Figure 1b shows the result of my endeavours. The only detail worth taking out of Figure 1b is its apparent complexity. I quickly realised that I could reason about the light box as a simple value entry system used to change the intensity level of the bulb.

This research started in April 2010 as part of CHI+MED – a multidisciplinary research project on human computer interaction for safe use and design of interactive medical devices. This project provided a platform of opportunities to

[†]He used the *Pico Wolf X 60* electronic transformer



(a) (b)

Figure 1: The lightbox (a) and its internal interaction user interface model (b). Each circle in the graph represents a state of the lightbox and the color of the circle represents one of seven levels of brightness.

engage and interact with devices in the medical context with various styles of number entry interfaces and I decided to explore the implications of the different styles for the design of safety critical interactive systems.

This thesis is the result of my work of exploring, analysing and discovering number entry interfaces over three years.

Style

I have elected to write up this work in a mixture of two different but coherent styles. I introduce and conclude the work in the active voice in order to clearly frame and state the contributions of this research. I report the scientific research within the body of the thesis in the passive voice.

Ethical issues

I recruited human participants for the experiments in Chapters 6 and 7. Consequently, prior to the recruitment stage of each experiment, I sought and received ethical approval for the experiment from the Computer Science Department's Ethics and Risk Assessment Committee in Swansea University.

Prior to the beginning of each study, each participant completed a consent form and was told about their right to withdraw from the study without any

penalty. Each participant that partook in the experiments received a gift voucher in return for their time.

Contributing publications

Some of the ideas and research reported here have been published in peer-reviewed international conferences and journals. I list these publications below, highlight my contributions and the chapters where they are reported.

1. **Oladimeji, P.**, Thimbleby, H., Cox, A.: *Number entry interfaces and their effects on error detection.* Proceedings of the 13th IFIP TC 13 international conference on Human-computer interaction. 178–185, (2011).

   My Contribution

   The concept behind this research was mine. I designed and ran the study, implemented the user interfaces evaluated, analysed the results and wrote the paper with some insightful feedback from the co-authors. The findings of this paper are reported in Chapter 6.

2. **Oladimeji, P.**, Thimbleby, H., and Cox, A. *A performance review of number entry interfaces.* Proceedings of the 14th IFIP TC13 Conference on Human-Computer Interaction, in press (2013).

   My Contribution

   Carlos Monroy and I designed the prototype device used in this study and the fabrication of the physical device was overseen by Ian Culverhouse of PDR[‡], Cardiff. I designed and ran the study, implemented the different interfaces and analysed the results. I wrote the paper with feedback from the co-authors. The findings of this paper are reported in Chapter 7.

3. **Oladimeji, P.**: *Towards safer number entry in interactive medical systems.* In: Proceedings of the 4th ACM SIGCHI Symposium on Engineering Interactive Computing Systems. 329 – 332, (2012).

[‡]http://pdronline.info/

My Contribution

This doctoral consortium paper contained my ideas of metrics that can be used for performing relative evaluation of number entry interfaces. This paper formed the foundation for Chapter 8, although the ideas have been refined into evaluative criteria over the course of this research, particularly in light of feedback I got during the doctoral consortium.

4. Monroy Aceves, C., **Oladimeji, P.**, Thimbleby, H., and Lee, P. *Are prescribed infusions running as intended? Quantitative analysis of log files from infusion pumps used in a large acute NHS hospital.* British Journal of Nursing, in press (2013).

   My Contribution

   I was responsible for writing the scripts that were used in analysing the logs files for the results reported in the paper. Part of Chapter 4 reuses this script for exploring the nature of numbers used on the Graseby infusion pump. I presented several new ideas not presented in the paper in Chapter 4 including the analysis of logs from two other devices. I also contextualise the predicted performance in Chapter 5 using results from this analyses.

5. Masci, P., Ruksenas, R., **Oladimeji, P.**, et al. *The benefits of formalising design guidelines: a case study on the predictability of drug infusion pumps.* Innovations in Systems and Software Engineering, (2013), 1–21.

   My Contribution

   I facilitated the formalisation of the Asena (Alaris) interface reported in this paper by providing simulations from which Paolo and Rimvydas base the creation of their formal models. I also contributed to the recommendations section in this paper. Specifically I provided recommendations for improving the chevrons interface with respect to improving speed and reducing error based on the results of an experiment [Ola11] which I conducted.

Chapter 1

Introduction

Interactive computer systems have improved over the years and despite various technological advancements, a notion that persists in interactive devices is that they need to be used either passively, for instance, by merely informing a decision based on a reading or feedback, or actively, by engaging the user in a more interactive session of input - output interpretation loop. The interactive nature of computing systems and devices, thus requires a means of communication between the user and the device. The user provides a sequence of commands for instructing the device about how they intend to execute a task. This means of communication can be broadly described as *data entry* and it may be transmitted through different forms of user interfaces including (but not limited to) speech based user interfaces or tangible user interfaces such as buttons or touch screen devices. In general, data entry in interactive systems usually has a simple purpose - to supply some value to an application in order to facilitate the execution of an intended task. This makes data entry a core aspect of using an interactive system.

Data entry itself has two components. One of them is *Data*, which is concerned with the type of information supplied to the system. The other is *entry*, which is concerned with how the information is supplied to the system. One type of data is text. Text entry interfaces primarily offer character level control. The user specifies the letters that make up the intended words. The use of

text is ubiquitous. This is evident, for instance, in the proliferation of mobile phones coupled with a myriad of social media applications that enable users to share text based information and messages. Consequently, there has been a lot of research on text entry for a variety of interactive devices in different contexts. These range from the performances of different layouts of the typewriter keyboard [Kro01] to that of a 12-key text entry interfaces found on millions of mobile phones [But02, Sou03, Sil00, San04].

Another ubiquitous type of data is numeric. Number entry has been a part of human culture since we learned to count. The cultural and economic importance of numbers is evident in their presence in languages around the world. Tally sticks were among the earliest artifacts that were used to represent numeric quantities [Dan68], usually acting as a memory aid that had better persistence than finger counting. After tally sticks came devices like various forms of abacuses and counting boards like the Salamis Tablet [Men92, p. 300]. Much later, starting in the mid 17th century, the advent of a series of mechanical calculators such as Pascal's Calculator and later, the Arithmometer, brought about a series of different design options for interacting with numbers and designing modern number entry user interfaces. Variants of these interfaces are still in use today in various interactive devices - although implemented in ways that account for technological advances both in software and hardware.

Tasks that require entering numbers are vital to the use of many interactive devices and are consequently extremely common. For instance we enter or select numbers at the ATM, we enter, select or modify numeric values in our microwave ovens to specify time, and we often change the volume on our music player. While performing any of these tasks, the user might be oblivious to the number entry aspect of the task, after all, you only wish to withdraw some money from the bank, warm up your food or increase the volume of music. Number entry is usually a sub-task to achieving a more primary goal and is therefore very often perceived as trivial.

The reader could probably think of more than one type of interface for performing these tasks. For instance, an ATM might use a 12-key numeric keypad, a microwave might use a dial and the music player might use a slider. In short,

there are several ways a number entry interface might be designed and implemented. Despite dating back many years, until recently, research has failed to identify a classification or a review of the performance of the different styles of interfaces that might be beneficial to designers of interactive systems.

Research on number entry has focused mainly on one type of interface: the 12-key numeric keypad found on many telephones and calculators. The popularity of this interface is not surprising as user interaction directly maps to the way numbers are written in western languages. This is just like text entry where digits in a number are specified sequentially from left to right. This makes it very easy to learn and adopt.

There are, however, other alternative ways for designing number entry interfaces, each having different context-dependent advantages and disadvantages including consequences for speed, error, user interface footprint or user experience. Many of these factors play an important role in the dependability of a style of interface in the safety critical context and particularly in their appropriateness of use in such contexts.

1.1 Data entry errors in medicine

A context of interest in this thesis is the healthcare setting where poorly designed data entry interfaces could lead to error, and errors could have severe consequences such as death. Medication errors are the leading cause of adverse events in hospitals with as many as 6.5% of inpatients and 27.4% of outpatients experiencing adverse drug events (ADEs). ADEs are responsible for 4.7% of all hospital admissions in American hospitals [Mor04, Bat95, Gan03]. The majority of these adverse events are preventable and have been classified based on their position in the medication process (e.g., see [Mor04, Rot05, Lis05]).

Errors occur in the following stages of the medication process:

- *Prescribing/Ordering stage* - (i.e., when medication is requested by the physician or consultant)

- *Transcription stage* - (i.e., when a physician's order is transcribed by a secretary or a nurse)
- *Dispensing stage* - (i.e., when medication is supplied by a pharmacist)
- *Administration stage* - (i.e., when medication is delivered to the patient by the physician, nurse or the patient)
- *Monitoring stage* - (i.e., when the status of an administered medication is checked by the physician, nurse or patient to ensure safe delivery)

All these stages in the medication process involve the correct perception and specification of numbers used to specify settings such as drug doses, frequency of therapy and duration of therapy. These can occur over a variety of contexts ranging from a General Practitioner's office, to a patient's home, and on a variety of medical devices ranging from computerised order entry systems running on PCs to drug delivery systems running on infusion pumps.

According to a National Patient Safety Agency (NPSA) report in 2007, about 7000 medicine doses are administered each day in each hospital in England and Wales [Age07]. Some drugs have to be administered intravenously due to the treatment requirements of patients who need multiple intravenous drugs to be delivered simultaneously [Keo05]. Devices such as infusion pumps, used for controlled delivery of drugs in hospitals, require *timely* and *accurate* programming in order to avoid patient harm [Age10]. Setting up an infusion pump requires entering numbers that correspond to the rate of infusion, the volume to be infused (VTBI) and duration of the infusion. Many adverse incidents in hospitals have occurred as a result of avoidable number entry errors in programming infusion pumps, (e.g., [Vic03, Wes11, ISM06]). Consequently, designers of medical devices ought to be able to make informed design decisions on number entry interfaces with a clear understanding of the strengths and weaknesses of a style of interface.

Tenfold medication errors are well reported in literature. They refer to numeric data entry errors where the intended number is different from the transcribed number by a factor of ten. Thimbleby and Cairns [Thi10b] refer to this type of error as an out-by-ten error. Tenfold errors have been documented to

occur during all five stages of the medication process although they are most common during prescribing (43%) and administering (35%) [Doh12].

Zhang et al. [Zha04b] report an example incident that they discovered in the Manufacturer and User Facility Device Experience (MAUDE)* database, that represents poor number entry interface design in medical devices. In this incident, a nurse inadvertently programmed a pump to deliver an infusion at 1,301 mL per hour instead of 130.1 mL per hour. The nurse did not realise that the decimal point on the infusion device in question was ignored for numbers above 99.9. This incident would have led to an overdose although the outcome is unspecified in the paper. Another incident reported by Syed et al. [Sye06] highlights a case involving a morphine overdose to a patient, who consequently went into respiratory arrest, but was resuscitated. One of the many errors that occurred in this incident was a number entry error where a nurse programmed a pump at a concentration of 0.5 mg per mL instead of 5 mg per mL. The lower concentration value entered into the device meant that the patient received more drug per volume than was entered into the device.

Although errors leading to ADEs can occur at different stages of the medication process, those that occur as a result of errors in interactive number entry are smaller and hard to estimate. It is difficult to get an accurate proportion of medical errors that occur as a result of number entry. Vicente et al., [Vic03] estimated for a single device that the probability of mortality due to programming errors has a range between 1 in 33,000 to 1 in 338,800. There is a limitation in the quantity of data available for analysis in the medical context. This is in part due to the voluntary nature of error reporting in the field, coupled with the ethical and technical issues involved in potentially automating the error reporting process. Research also suggests that errors are under-reported for reasons including time pressure and a culture where errors are typically associated with punitive consequences [Res03].

Despite error rates being low, the consequences of error can be potentially devastating. As a result, this research proceeds on the premise that errors will occur and sometimes go unnoticed, however rarely.

*http://www.accessdata.fda.gov/scripts/cdrh/cfdocs/cfmaude/search.cfm

1.2 Problem statement

The importance of safety in healthcare means that medical devices need to be designed to have the highest degree of dependability, reliability and safety. This means medical devices need to be designed in the best possible ways to properly handle and manage error in order to reduce harm to patients that arise as a result of poor interaction design.

The focus of this Thesis is the number entry aspect of data entry. Currently, a variety of number entry interface styles are in use in medical devices without any empirical evidence showing which design option is better and why. This research explores the design space of number entry interfaces focusing on interfaces that have been in use in medical devices (specifically those in use in infusion pumps).

The overall goal of this Thesis, then, is to explore different ways of designing number entry systems, evaluate the performance of the different interface styles and understand the trade-offs involved in choosing to implement one number entry interface style instead of another. The motivation is to help improve the design choices made by designers of interactive number entry systems.

I set out to achieve this goal by:

1. Exploring the design space of number entry interfaces. The design space will be validated using instances of number entry interfaces found throughout history, starting from early designs found on ancient calculators to more recent designs.

2. Building a repository of simulations of different styles of number entry interfaces to enable exhaustive analyses for predicting the performance of the different interface styles and identify features in each interface style that are unique to that style as well as identifying potential bottlenecks for speed and accuracy.

3. Running user studies to evaluate the performance of a selection of the number entry interface styles identified earlier as well as evaluating the effects of interface style on error detection. I will run the repository of

simulations on a high fidelity prototype unit built specifically for testing number entry systems.

4. Identifying a list of criteria that can be used to perform a relative comparison of the different styles of number entry interfaces available to improve decision support for designers of devices that require number entry interfaces.

1.3 Methodology

In this Thesis, I employ a dual approach to evaluations based on analytical evaluation and usability studies. Where possible, I initially perform preliminary analyses based on the user interface model discovery method [Thi07, Gim10]. I then conduct laboratory studies to support/validate the results of the preliminary analytical evaluation. The reason for this dual approach is to harness the strength and relative speed of automated and exhaustive user interface model discovery with the results obtained from running user studies including the qualitative feedback obtained when talking to users about usability perception of interfaces.

Model discovery is a technique in interaction programming and was first described by Thimbleby [Thi07]. The method systematically explores the user interface of a class of interactive devices that support discrete user interactions (usually button clicks). It involves exhaustive exploration of an interactive system by successively stimulating all possible user actions permissible on the interface until all possible states in the system reachable by user interaction are explored.

The model discovery process produces a graph of states in an interactive device (nodes) connected by the user actions necessary to transition between these states (edges). A state in this context is modelled by a set of variables in the interactive system. The graph produced can thus be formally analysed using graph theory algorithms such as finding the shortest paths between any two states in the device, the most central state in the device, as well as finding safety properties such as connectedness. In this method, one can apply network analysis on the resulting graph to find out user strategies or quantitative usability metrics such as the average menu depth for an interactive device [Thi09b].

Model discovery is particularly suited to the analyses of number entry interfaces, specifically in the medical device domain because the range of numbers used in medical devices such as infusion pumps is finite as I shall show in Chapter 4. This makes the number of states for a typical interface discovery process for number entry tractable. For the purposes of this Thesis, I explore the shortest paths properties for a variety of number entry interfaces to discover the cost of setting and changing numbers on these interfaces.

1.4 Thesis structure

In the following chapter, in order to have a view of the evolution of number entry, I look at different instances of interfaces built for interacting with numbers throughout history. This review sets the scene for understanding the similarities between number entry interfaces and teases out the fundamental functions necessary for the design of different styles of number entry interfaces. I then conduct a review of literature related to performance metrics used in data entry research. Specifically, I explore metrics used in text entry research and highlight the limitations of those metrics with respect to the evaluation of number entry interfaces. I also review past and current research on the evaluation of number entry interfaces with respect to ergonomics (i.e., keypad layout) and error. I conclude the chapter with a review of current practices for managing numeric errors.

Following this, I explore the design space of number entry interfaces in Chapter 3 in order to create a classification of number entry interface styles. I will use this classification to assess the performance of the different interface styles in Chapter 5 and explore the relative task completion times for number entry using instances from the different classes of interfaces. The results of this analysis sets up predictions for usability evaluations performed in Chapters 6 and 7.

Chapter 6 explores the concept of the resilience of an interface based on the likelihood of a user to detect and correct an error while using the interface. This experiment monitored the eye fixation of users while entering numbers on two types of interfaces. Chapter 7 expands on the experiment run in Chapter 6 in two

ways. Firstly, it increases the number of types of interfaces tested and secondly, it introduces a high fidelity prototype unit on which the experiments are run.

Chapter 8 presents a list of criteria that could be used to perform relative comparison of different number entry interfaces. These criteria would be useful to designers for structuring the options in the design space of number entry interfaces while also being able to understand the trade-offs between one interface and another in a given context.

Chapter 9 presents a reflection on the results of my research with an emphasis on research contributions as well as new research questions that show the viability of the research area and possible future research directions.

1.5 Research contributions

This thesis makes the following contributions to research in number entry interface design:

1. A classification of number entry interface styles.
2. An evaluation of the effects of number entry style on error detection.
3. An evaluation of the effects of number entry style on performance using custom built high fidelity prototype units.
4. A set of properties for evaluating number entry interfaces and performing relative comparisons between a number of design options.

Chapter 2

Background

Number entry is an important aspect of using many interactive devices. Today, we perform many number related tasks, possibly without realising. Tasks such as dividing up the bill in the restaurant with a calculator, or specifying an amount to withdraw at an ATM are obvious number entry tasks. Others, such as reducing the volume of a music player or tuning a radio station manually might not be so obvious.

Since numbers have been an integral part of many cultures, designing interfaces that are used to interact with numbers either to easily manipulate digits or for a higher order goal such as performing arithmetic has been necessary. As a result, the design of number entry interfaces dates back to the invention of very old devices like the Salamis counting tablet and the abacus. Later the invention of mechanical calculators brought about a variety of number entry interface designs.

2.1 Number systems

The design of modern number entry interfaces is greatly influenced by the numeral system used for expressing numbers. According to Menniger [Men92], some of the earliest forms of expressing numbers include the use of finger counting, tally sticks and counting boards such as the Salamis counting board or the

abacus. Early systems such as the unary system of representing numbers rely on the use of repeating symbols that represent predefined numeric quantities. In its simplest form only one symbol representing the unit quantity is required. So, the number 3 could be represented by the string '*aaa*'. A popular modification of the unary system is the Roman numeral system where there are explicit symbols for representing frequently used large quantities such as X, L, C representing 10, 50, 100 respectively. There is no need to represent zero in this system because the value of a written number is derived by successively adding up the quantities represented by each symbol.

The numeral system most common today is the positional system or the place value notation usually expressed in base 10. In this system, a numeral in a number is assigned a value which depends on the position it occupies with relation to other numerals representing the number. Thus, the same numeral can be attributed different meanings in different positions, for example, the 7 in 897, 75 and 730 represent seven, seventy and seven hundred respectively. This notation was only possible after the invention of the symbol for zero by an unknown Indian mathematician [Dan68].

2.2 A survey of number entry interfaces

A good understanding of the design of number entry interfaces is impossible without an understanding of the history of design of calculators. Below is a review of examples of the common types of number entry interfaces found in calculating machines throughout history.

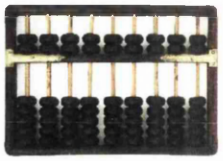


Figure 2.1: An abacus

2.2.1 Pre 17th century

The first forms of number entry interfaces were seen on ancient counting devices and ancient calculators. While counting devices such as tally sticks used the unary system of representing numbers, the abacus (see Figure 2.1) primarily used a place value notation for representing numbers although they used a modified unary approach for representing the digits within the numbers. For instance, abacuses are usually split into two rows with each bead in the top row representing a numeric quantity of 5 while each bead in the bottom region represent the unit number. The value of each digit is obtained by adding the active beads in the top and bottom row.

2.2.2 Between 17th century and early 20th century

The calculating machines [Mar92] and *Origin of modern calculating machines* [Tur21] are both invaluable resources containing historical accounts and comprehensive reviews of calculating devices between 1642 and 1925. By the 17th century, the use of the positional notation was well established around the world and as a result, number entry interface designs were mainly about controlling or specifying the digits that make up a number. Consequently, calculating machines had interfaces made of dials, setting slides or levers. Key driven interfaces were invented much later.

Dials, sliders and levers

This period saw the invention of Pascal's Calculator (also known as the Pascaline) in 1642 (Figure 2.2), and the first Thomas Machine or Arithmometer in 1820 (Figure 2.3). Dial-based number entry interfaces, such as Pascal's Calculator allowed number entry using a series of toothed wheels each marked with digits 0 to 9. A varying number of these wheels (typically 8) were used to specify the digits of the number the user wishes to use in their calculations. It was common for these dials to be operated using a stylus, such that a user would set the intended digit by inserting a stylus at the intended number on the wheel and then rotating the wheel until it stopped, similar to the way dial-based telephones worked. Errors in entry could be easily corrected by simply changing the digit in

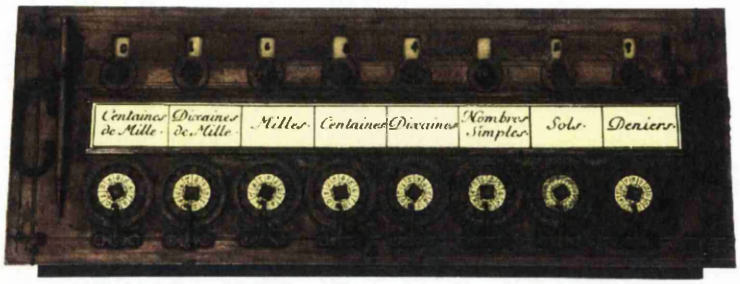


Figure 2.2: Pascal's Calculator (1642)

the same way it was set initially. Some devices provided a dedicated mechanism for resetting the system. Calculators that used levers or sliders to set numbers, such as the Arithmometer or the Millionaire worked similarly except the input widget used was a slider with digits marked from 0 to 9.

Key driven calculators

Towards the end of the 19th century, various engineers around the world invented smaller key-driven adding machines. These devices typically had 10 keys and were only used for adding up a single row of numbers (since they were only able to specify one digit).

In 1885, Dorr E. Felt invented the Comptometer, a key driven mechanical calculator with a *full keyboard*. It had explicit keys for specifying all possible digits up to the hundred thousands place value. Figure 2.4 shows several examples.

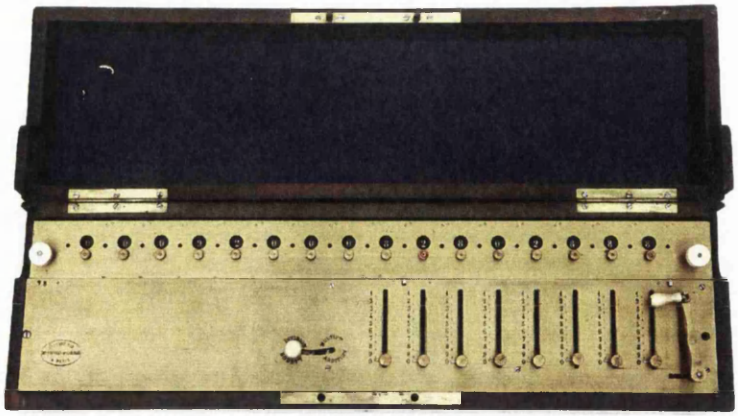


Figure 2.3: The Thomas Machine (1820)

Many devices invented after the Comptometer used a similar interface that provided a $9 \times N$ grid of digit keys, where N is the maximum number of digits the device permits the user to enter during a calculation. Note that N is usually less than the maximum number of digits that could be displayed in the result of a calculation. The numbers within each column of keys begins at the bottom, from 1 and up to 9. There were no zero keys and the display section of the interface started off as zeroes. Users entered numbers by pressing on the required digit in the column representing the intended place value, one at a time. If the user committed an error and detected it, they could simply press the correct digit in the same column to fix the error. In instances where the wrong place value was entered, (i.e., the user unintentionally selects a digit in the wrong column), some models had dedicated keys in each column that reset the digit to zero. Others simply had a mechanism for resetting all the keys. When used for currencies, the tens and units columns were used to denote the tenths and hundredths place values used for pennies or cents for example. To facilitate easy demarcation, many interfaces coloured these two columns differently from the next three.

Two improvements to this style of interface made it faster and less error prone. A feature called *multiplex keys* allowed the user to simultaneously press on multiple keys from different columns. This allowed significantly quicker number entry on the interface. A feature called *controlled keys* prevented entry of the wrong number if the key was not properly depressed*.

The Kollektor (pictured in Figure 2.5a) was invented in 1910 and was relatively portable in comparison to its predecessors. It had only four keys and could be operated using four fingers on the left hand. The keys $\boxed{1,\ 3,\ 4,\ 5}$ were provided on the interface for specifying the corresponding digits. Other digits were produced by double tapping on a combination of the four digits. For example, $\boxed{2 = 1,\ 1}$, $\boxed{6 = 3,\ 3}$, $\boxed{7 = 3,\ 4}$ and so on. When setting multiple digits, users had to remember to move the calculator mechanism to the next digit place to avoid performing incremental additions on the same digit.

*The precise effect of errors that occurred due to partially depressed keys is unclear from [Mar92]. One could argue that the feedback provided when keys are depressed is just as important in ensuring that users are consistently aware that they have not properly activated the key mechanism. This would have been even trickier with *multiplex keying*.

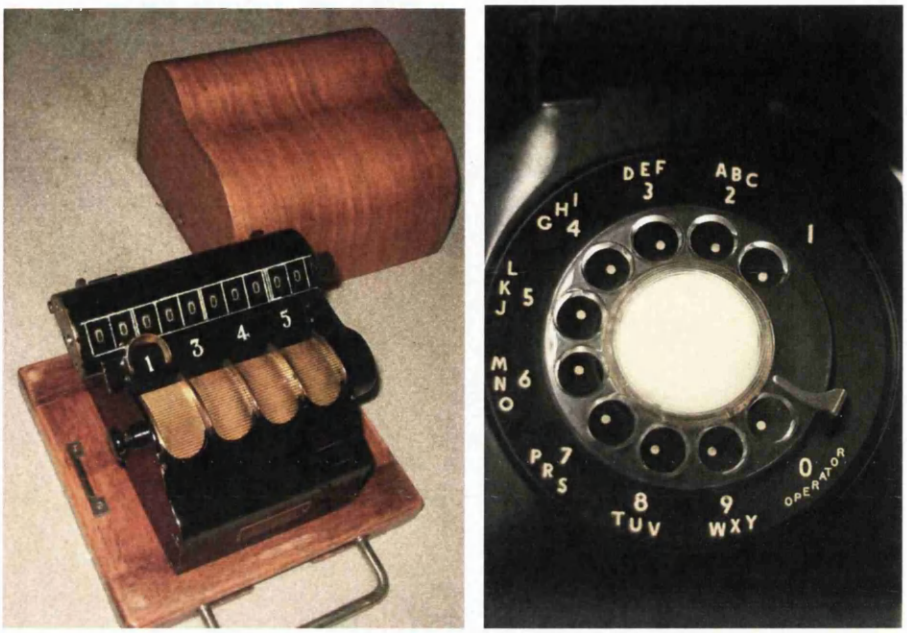
(a) Comptometer (1890)

(b) Burroughs adding and listing machine (1912)

(c) Marchant "ACRM" Calculator (1932)

(d) Monroe LA5-160 Calculator (1940)

Figure 2.4: A variety of full keyboard mechanical calculators in use between 1890 and 1940.



(a) (b)

Figure 2.5: The Kollektor (a) and the Rotary Dial (b)

2.2.3 Post 19th century

Various forms of key-driven calculating machines were produced in the early 20th century although most were of the 'full-keyboard' variety, since they allowed the user to work with numbers in more than single digits. The rotary dial (see Figure 2.5b) was the earliest interface used to enter numbers on telephones. It was a mechanical input device with numbers arranged from 1 to 0 in a circular layout. A number is entered by entering each digit sequentially. A digit is set by using a finger to select the required digit on the wheel and rotate the wheel to the stop position. When the wheel is released, the numbers return to their home position thanks to a spring loaded mechanism. This period also saw the design of telephones which required numeric keypads for specifying telephone numbers [Web12]. Early telephones had a rotary dial interface.

Significant advances in electronics between the 1940s and 1950s led to Casio building the first all electronic calculator in 1957. This period also saw the development of the first push-button telephone sets. The main difference in the

interface design for calculators of the 20th century was the reduction in size of the number of keys present on the devices coupled with an electronic display. This came with the cost that number entry had to be strictly sequential order. It also raised the possibility of alternative key layouts. Figure 2.6 shows various interfaces with differing layouts.

2.3 Text entry research

One of the most common forms of data entry is text entry. There has been a lot of research on text entry ranging from the performance of different layouts of the typewriter keyboard [Kro01] to the layouts of 12-key text entry interfaces found on millions of mobile phones [But02, Sou03, Sil00, San04]. Text entry is a special form of data entry where the words formed from the alphabet are from a relatively small set of valid values as defined by the vocabulary of a given language. This has implications for designing the layout of keyboards in text entry interfaces (in terms of improving speed) as well as better managing errors. A list of valid words make it possible to build error correction models based on

Figure 2.6: A selection of key driven number entry interfaces invented post-19th century.

the similarity of an incorrect word to one in the valid set.

There are various examples of text entry interfaces in use on computer systems today. On desktop systems, these are typically different layouts of the full alpha-numeric keyboards - the QWERTY keyboard being the most common. On mobile devices, a common practice is to overload the numeric keypad with functions for text entry due to the limited surface area available for buttons in these interfaces [San04]. According to Fitts' Law [Fit54], there is a trade-off between speed and the size of keys. This puts a limit on the number of buttons that can be presented on a given surface area. This limitation means mobile devices need to implement text entry with a set of keys much less than the alphabet they are addressing. Consequently, researchers and designers have invented methods such as multi-tap, chording and dictionary based methods for text entry.

The multi-tap technique was one of the earliest mobile text entry methods and it involved tapping on a single key multiple times to cycle through the letters attributed to that key. With the chording technique, users simultaneously press several keys to specify a letter. The linguistic or dictionary based technique, is a form of predictive text entry and is currently very popular in commercial applications. In this method, each key or area can be assigned to multiple letters of the alphabet and a dictionary of words in the entry language is used to disambiguate the user's keystrokes [Mac02b]. This allows automatic text entry when there is no ambiguity. In cases of ambiguity, the application defers correct selection to the user from a list of suggestions based on the words obtained from the language model [Kre89]. The dictionary based model has also been used to optimise the order in which characters are presented in entry lists for three- or five-key text entry keypads [San04, Mac02c].

The advent of better touch sensitive hardware has increased the surface area of mobile devices and many text entry interfaces are now implemented as virtual keypads with gesture control such as shape writing [Zha12, Kri07, Dun08]. The larger surface areas enabled by touch screen interfaces has also allowed devices to have the full alphabet on the keyboard. Current research includes improving language dependent error correction models, or creating optimised key layouts for better speed, accuracy and ease of use [Oul13].

2.3.1 Metrics for text entry interface evaluation

A popular metric for measuring the speed of a text entry interface is the number of words a user can enter on the interface over a given period. This is commonly measured as words per minute (*wpm*). The advent of keyboards with keys using overloaded characters brought about different metrics for measuring other aspects of performance of a text entry interface such as error.

A metric for measuring error rates is the minimum string distance (*MSD*) between the transcribed text and the presented text [Sou01a]. MSD is based on the Levenshtein distance between the presented and transcribed text [Lev66]. This distance is calculated as the minimum transformations (i.e., insertion, deletion or substitution) needed to turn the presented text into the transcribed text. For example, the Levenshtein distance between the word insertion and insert is 3 - representing the three deletion actions needed to convert between the two. A common short-coming of the MSD metric is that it does not take into account the corrected errors in a transcription process. Since it is derived from the transcribed text, corrections do not feature in the evaluation. Key strokes per character (*KSPC*) improves on the MSD metric by incorporating the length of characters in the *inputstream* of transcription. This *inputstream* contains all the keystrokes that were involved in the transcription process. KSPC is a measure of the speed of an interface with respect to how many key strokes are needed to enter a character on that interface. This is 1 for full alphabet keyboards, 1.0072 for dictionary based disambiguation and 2.0342 for multi-tap text entry on a 12-key mobile phone [Mac02a]. KSPC also serves as a metric for evaluating error rates in a text entry interface. Soukoreff [Sou01b] defines KSPC as the ratio between the number of key strokes issued by the user and the number of characters in the user's transcribed text. KSPC however gives an indication of both the efficiency of an interface and a sense of the number of errors a user encountered while using the interface. It does not distinguish between the two. As a result, it is difficult to tell whether a high KSPC value is due to high rate of error or poor interface efficiency.

The nature of number entry tasks are different from the typical text entry task especially in terms of the duration spent on the task as well as the length

of characters or digits involved in the task. Number entry tasks are usually much shorter than text entry tasks and are often found as sub-tasks in text entry. Moreover, not all number entry interfaces operate on the character level of control. Consequently, text entry evaluation metrics are mostly not applicable in the evaluation of number entry interfaces. For instance, the MSD metric gives an indication of the transformations required to change a number into another. The main limitation of this metric is that it is only applicable to number entry interface styles that control numbers at a digit level of operation. Furthermore, no current text entry metric gives an indication of the severity of an error, which is an important safety factor. Secondly, number entry language models cannot be built the same way text entry language models are built based on valid words possible in a given language. As a result automatic error correction of numeric input is limited to obvious syntax errors such as multiple decimal point characters. In general, criteria used for comparative assessment and evaluation of number entry interfaces must be generic enough to apply to all styles of interfaces as well as exploit the numeric properties of numbers.

The rest of this chapter discusses related work in data entry that contextualises the state of research in the design and evaluation of number entry interfaces. Specifically, the review of research will focus on the evaluation of different aspects of number entry interfaces, and how data entry errors have been managed in a variety of research areas. It concludes by highlighting the need for more fundamental research in the classification and evaluation of different classes of number entry user interfaces.

2.4 Number entry interface research

2.4.1 Number entry interaction styles

The Light Handle [New68] was a programming technique used to emulate the effect of rotating a virtual knob on a display by using a light pen to make clockwise or anti-clockwise rotations on the display. These rotations cause incremental changes to a number. The direction of rotation designated the type of change made to the number (i.e., increments or decrements) and the horizontal center

of the rotation determines the rate at which the number changes. Slow changes are caused on the right side of the designated input area while faster changes were caused on the left side. In addition, the rate of change of the value was also dependent on the speed of movement of the pen.

The Number Wheel used the notion of a sliding gesture for specifying numeric values on a tablet [Tho79]. Its function is analogous to that of a wheel whose circumference partly protrudes through an area on the surface of a tablet, similar to a thumb wheel. The range of values in the application is defined by points on the circumference of the thumb wheel such that lateral movements on the circumference makes corresponding changes to the numeric value specified. Given the analogy of a wheel, the question about how to appropriately deal with situations where the user attempts to turn the wheel past its full circumference must be addressed. Thornton identified two possible ways to deal with this. The wheel can *wrap around*, such that the maximum value changes to the minimum value. On the other hand, the action could be blocked such that the wheel stops changing the numeric value until it is moved in the opposite direction. Thornton also identifies various parameters that can be used to specify number wheels, including the range of values addressable by the wheel, the input and output resolution of the wheel and the limit condition to use i.e., *wrap around* or *stop*.

In their seminal work on the human factors of interaction techniques, Foley et al. identified *Quantify* as one of six types of fundamental interaction tasks [Fol80, Fol84]. Other interaction tasks they identified were *Select, Position, Orient, Path,* and *Text.* They describe *quantify* as a task where the user specifies a numeric value to an application. They identify two broad forms of quantifying techniques based on discrete and continuous interactions. These techniques are reviewed below.

- *Continuous quantifying by direct interaction* can be accomplished with a physical device such as a slider, dial or a touch pad. A bounded dial may be used to specify numeric values in the range defined by the limits of the dial such as those found in radio volume controls. A slider may be used in a similar way where physical limits on the widget signify the range

the user can specify to the application. These generally specify absolute quantity. An unbounded dial or a touch pad on the other hand can be used to specify relative quantity where user actions simply map to increments and decrements to the numeric value in the application.

- *Continuous quantifying by scale drag* involves the user pointing to a value indicator on a scale or gauge and then moving along the scale to specify the required value. A highlighted marker may be used to indicate the currently selected value or a numeric value may be updated on the screen as the user moves up and down the scale. In this case, a mouse or (graphic) tablet is used to move to the required value.

- *Continuous quantifying by locator value* involves a situation where the user moves a locator along an axis and the position of a pointer on a scale is changed accordingly. For instance the movement of a device such as a mouse is mapped to the movement of some pointer or cursor on a scale. This is similar to how the Number Wheel described by Thornton works [Tho79].

- *Continuous quantifying by simulated stopwatch* where a numeric value changes at a constant rate when the user pushes a button. The rate of change may be regulated by a dial and the user accepts values by releasing the button.

- *Discrete quantifying by type-in* where the user types in a numeric value with a keyboard.

Mackenzie et al. [Mac94] explored the performance of pen-based numeric entry on computers. They exploited users' familiarity with the number arrangements on a clock face and developed an interface that used the notion of gestures and pie menus to enable number entry using a clock metaphor. In the clock metaphor, a circle is divided into 12 sectors with the digit '0' at the 12 o' clock position. The 10 and 11 o' clock positions were not used. Users of the system could stroke from an arbitrary starting point towards the position of the number on the face of a clock. They investigated two entry methods using this interface. One variation required the user to perform a stroke at the insertion point of the digit, similar to handwriting. They called this the *moving pie menu.* A second

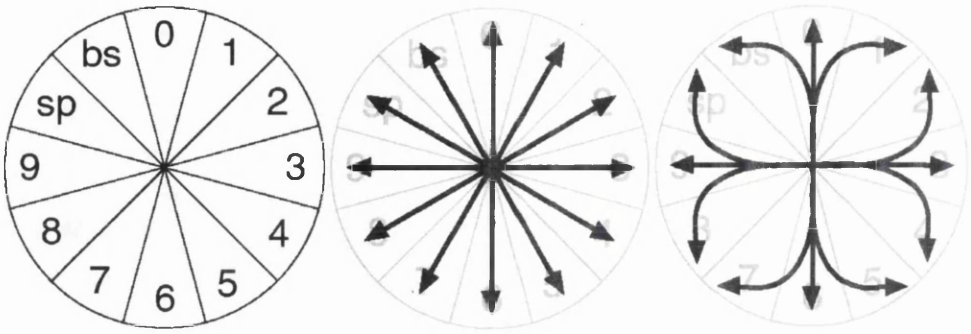


Figure 2.7: Pen-based number entry interfaces using the clock metaphor

variation utilised a stroking pad where strokes were made on top of each other and the resulting digit was automatically appended to the end of the sequence on the interface. They called this the *pie pad*. They compared these two variation with handwriting and the numeric keypad. Their results showed that the numeric keypad was fastest and most accurate with a 1.2% error rate.

In a follow-up to MacKenzie's previous experiment, McQueen et al. [McQ95] compared the *pie pad* and *hand writing* numeric input method specifically to investigate the effects of learning on the performance of both interface styles. They found that participants' entry speed changed significantly after several sessions of the study. Although handwriting was initially faster, the *pie pad* was 24% faster than handwriting by the 20th session. This suggested that participants were able to quickly learn how to use the new interface effectively.

A similar study by Isokoski and Kaki [Iso02] compared two numeric entry methods on handheld touch pads using fingers rather than styluses for input. They compared McQueen's piepad with a hybrid clock face design where digits on the clock face were selected using an "L" shaped gesture as shown in Figure 2.7. The shape is formed by first following the nearest axis and then turning towards the number. Their experiments involved entering a series of 5-digit numbers and results showed a significant improvement in error rate from the pie-pad.

Lin and Wu [Lin13] performed a study to investigate the differences between touch screen devices and physical keypads in the context of numerical typing. They got participants to enter 30 random 9-digit numbers using three interfaces

with the calculator style numeric keypad layout. Two of the interfaces were on a touch screen. One variation showed precise visual feedback of where the user touched on the interface while the other showed inprecise feedback by just inverting the selected button. The third interface was a physical numeric keypad. Their experiment showed that touch screen keypads were as accurate as physical keypads although participants had a slower response time on the touch screen interface. The slower response time on the touch screen interfaces was attributed to a slower pre-motor response time in preparing motor execution for tasks on the touch screen interfaces.

2.4.2 Keypad layout

One of the earliest documented experiments in the design of numeric keypads was performed by Deininger at Bell Labs [Dei60b, Dei60a]. Deininger was interested in how people processed information when entering telephone numbers. He also wanted to find out desirable design features for use in dial-based telephone sets. Amongst the factors he explored was the effect of key arrangement on the performance of users entering telephone numbers. Sixteen different key arrangements shown in Figure 2.8 were evaluated in groups of three in an initial study. As part of the study, participants entered between 10 and 15 telephone numbers containing two letters and five numerals. Based on the results, four layouts (IV-A, II-A, IV-B and I-C in Figure 2.8) found to be superior were further compared with the then standard telephone rotary layout. Although the results of the second evaluation showed that both the circular and rectangular layouts were acceptable, the rectangular arrangements offered better engineering advantages.

Further studies of the rectangular layouts showed that layout IV-A could be made to cover a smaller area by reducing the space between the centre of two buttons to $\frac{5}{8}$ of an inch without causing significant change in performance. This layout has the keys 1 2 3 at the top and is currently in use today in telephones. It is different from that found on calculators which have keys 7 8 9 at the top.

Given the two popular layout in IV-A and I-A, that is, the telephone and the

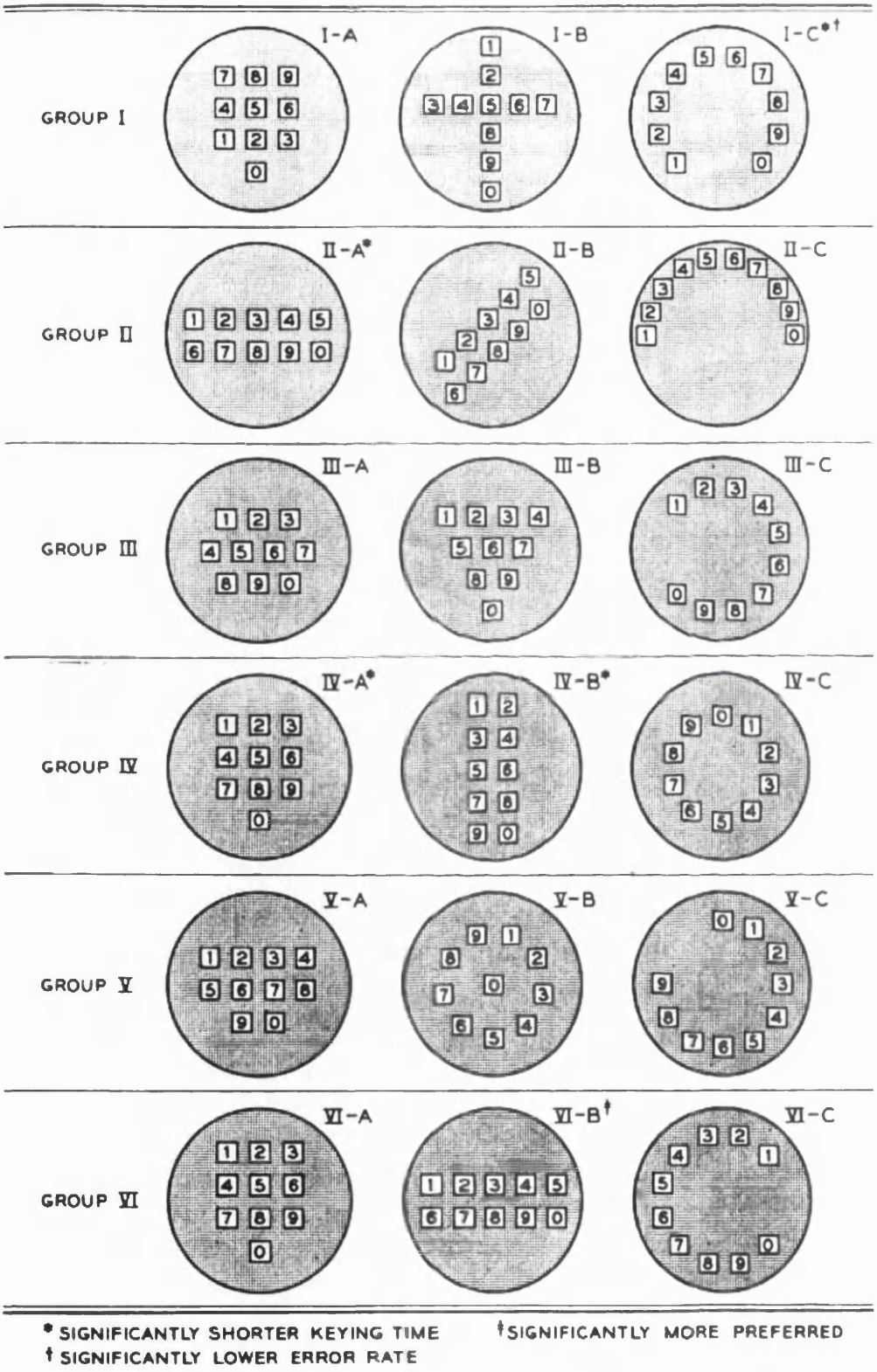


Figure 2.8: Deininger's keypad layouts (Adapted from [Dei60b])

calculator layouts, Conrad and Hull [Con68] compared these interfaces to explore their effect on speed and accuracy for numeric data entry. They assigned subjects into three groups. One group worked exclusively on the calculator layout, the second group worked exclusively on the telephone layout and the third group alternated between the two layouts. The subjects were housewives who had no familiarity with either the telephone or calculator layouts. Over a period of four days, the subjects entered 8 digit codes for 30 minutes everyday. The results showed that the group that alternated between interfaces performed worst both in terms of speed and accuracy. In addition, the group that worked exclusively on the telephone layout was more accurate than the group that worked exclusively on the calculator layout although there was no significant difference in speed. Since the subjects had no experience with either layout, the difference in accuracy was attributed to the fact that the telephone layout conforms more to where subjects expected numerals to be found on the keypad [Lut55]. Later research by Rink [Rin99] supported this reasoning as people tend to recall the layout of the telephone keypad with better accuracy than the calculator layout.

Other researchers have focused on the effect that the task being performed has on the user's choice of key layout. Straub and Granaas [Str93] presented 8 scenarios describing number entry tasks to 100 users and asked them to choose one of the calculator or telephone keypad layout to perform the task. The tasks included entering telephone numbers, personal identification numbers and performing calculations. They found that users preferred to use the telephone layout when they were performing telephone related tasks whereas the preference for the telephone layout was lower when performing tasks related to calculations.

Later research by Marteniuk et al. [Mar96] suggested that the performance difference previously found between the different configurations of the telephone or calculator layout were as a result of the placement of the zero key. Their study investigated the possibility that number entry task performance is affected by both the type of task being performed and the layout configuration of the keypad used. They devised four configurations of the numeric keypad comprising of the calculator and the telephone layout each having two variations where the zero key was placed at the bottom or at the top of the keypad. They used three

types of tasks in their experiment including 4 digit strings, 7 digit strings and 7 digit strings formatted as North American telephone numbers. Using each of the four interfaces, each participant entered 20 instances of the designated type of number. Their results suggested that the performance difference found across the four interfaces were a result of the placement of the zero key. They recommended that the zero key be placed below the other keys in the layout.

These experiments were all based on the numeric keypad and the types of numbers used in the research are long alpha-numeric codes, mainly in the form of telephone numbers.

2.4.3 Errors

Numbers encapsulate a precise quantity and in safety critical environments such as in hospitals, the accuracy and timeliness of medication delivery is essential to reducing any risk of harm to patients. Numbers are used to specify quantities representing drug doses, concentration, volumes, patient age or weight. The design of the number entry interface contribute to the categories of errors that occur in the use of medical devices.

According to Reason [Rea90, p. 9], error is a generic term to encompass all those occasions in which a planned sequence of mental or physical activities fails to achieve its intended outcome, and when these failures cannot be attributed to the intervention of some chance agency. There are three main types depending on the cognitive stage at which they occur in Norman's Action Cycle [Nor02]. *Mistakes* are errors that result from failures in the *planning* of an action sequence, *lapses* are errors resulting from failure in the *storage* of an action sequence and *slips* are errors that result from failures in the execution of an action sequence.

Mitigating the effects of error is an important goal in the design of interactive computer systems. There are currently two approaches to achieving reliability in organisations [Bla06]. One method of managing error focuses on prevention and the other focuses on resilience of a system. The prevention method implies that the possibilities of errors in a system are predetermined and preempted in the design of the system so that opportunities for error can be avoided. The resilience

approach focuses on designing systems that can cope with error for instance by ensuring appropriate feedback that increases the likelihood that users notice and deal with errors. In Chapters 6, 7 and 8, the effects of keying slips are investigated with respect to the severity of errors they cause on different styles of interfaces.

Number entry error

Events involving number entry errors have been documented in literature in settings ranging from finance to healthcare. In 2008, a Norwegian inadvertently sent a large sum of money to an unintended recipient. She keyed the wrong account number entering a twelve digit number instead of an eleven digit number [Ols08]. Unknown to her, the web interface she used discarded the last key she pressed and her entry was still a valid account number - although one completely different from what she intended. This resulted in a serious financial error. In healthcare, a missing or wrong digit in a patient identification number could lead to a dangerous situation where the wrong patient gets and unintended medication. However, numbers in healthcare are used for things other than as patient identifiers.

In 2009, the Food and Drug Administration (FDA) reported a fatal incident involving an infant [FDA09]. The patient was given an overdose 10 times the intended amount due to a missing decimal point error. Further investigation showed that the pressure needed to activate the decimal point key on the device was more than the pressure needed to activate the other keys.

More recently, research has focused less on the ergonomics and layout of buttons on the numeric keypad and more on understanding number entry error, improving design to reduce the risk of error and investigating the resilience of given design options against number entry error. By getting users to enter numbers using the calculator style number entry interface and in conditions specifically designed to elicit number entry errors, Wiseman et al. [Wis11] collected a repository of the different types of errors people make while transcribing numbers. They used this error repository to build a number entry error taxonomy that provides a classification of number entry error based on their underlying causes at a cognitive level. Their taxonomy classifies errors based on their position in

Norman's Action Cycle [Nor02] and contains 21 types of number entry errors.

Traditional usability studies are however very expensive when running experiments with a high number of variables e.g., when testing multiple interfaces each having different ways they can be implemented. The large space complexity coupled with low rate of errors in the laboratories, has led other researchers to take mathematical approaches in exploring number entry errors. These techniques are usually in the form of computer simulated exhaustive interface analyses.

For example, Thimbleby and Cairns [Thi10b] have shown using a variety of quantitative mathematical techniques that the probability of ten-fold or out-by-ten errors, which are a significant risk to patient safety [Les02, Doh12], can be significantly reduced by better programming. These errors are mainly caused by missing decimal points, unintended repeated digits, missing digits or an improperly parsed user input such as the entry of multiple decimal points. Thimbleby and Cairns show that mainstream devices and programs in use in both safety critical settings e.g., infusion pumps and non safety critical settings do not correctly support recovery from error or indeed in some cases they do not correctly interpret a mistyped sequence of numbers. For example, keying in 1 . 2 . 3 is interpreted by one device to mean *1.3* while in some other devices it means *1.23*. They propose a method for parsing the input stream from the numeric keypad so that syntax errors such as multiple decimal points are correctly detected and alerted to the user.

Following this and based on inspection and review of number entry interfaces in interactive systems, Thimbleby and Gimblett [HWT11] provide evidence through various examples that suggests that a lot of data entry systems are implemented in an ad hoc manner. For example, a good number of handheld calculators do not detect overflow in number entry. This means performing calculations with numbers that are larger than the range permissible on the display of the device results in an error. Dividing 308,000,000 by 6,800,000,000 incorrectly results in 0.45. This introduces a level of unpredictability for the user particularly when users are correcting syntax errors in the course of interaction. These apparent ad hoc implementations could be attributed to the lack of a systematic approach in the development of dependable keyed data entry interfaces.

Thimbleby tackles the lack of systematic approach by introducing a design that is correct by construction (CxC) [Jon06]. CxC means that formal methods are used during the production of software rather than afterwards.

The Institute for Safe Medication Practices (ISMP) sets out recommendations for displaying numeric information in medical devices [Ins06]. Thimbleby and Gimblett formalise these recommendations using regular expressions and gave an example of how a dependable interface might be implemented using feedback based on a traffic light system. Here data entry errors, (i.e., those that fail the ISMP regulations) are coloured red, incomplete data specifications are coloured yellow and correct data specifications are coloured green. This technique is however only amenable to number entry interfaces where syntax errors are possible.

Using a similar technique, Cauchi et al. [Cau12a, Cau12b] ran simulations that explored the effects of key slips on 28 variations of the so-called five key number entry interface. A five key or directional pad (d-pad) interface has four navigation style up-down-left-right buttons that are used to navigate a cursor around the place value of numbers as well as increase and decrease selected digits. This interface is described in more detail in Chapter 3. Cauchi et al. introduced a method called stochastic key-slip simulation (SKSS) where one of deletion, transposition, substitution or insertion errors are applied as transformations on a sequence of keystrokes which are executed by a computer simulation. The simulation essentially represents a user performing a task as defined by the keying sequence. The resulting sequence is executed on the interface and a measure of error is obtained by analysing the difference in the intended numeric output and the actual output obtained based on the transformed sequence. With this method, they were able to rank the different variants of a five key interface based on their resilience to keying errors.

2.4.4 Methods for data entry error reduction

In practice, there are currently different methods adopted or suggested to reduce number entry errors or to improve the possibility of detecting number entry

errors based on research in the field of coding theory, accounting and healthcare. A review of these methods follows with a highlight of their shortcomings.

Wang et al. [Wan11] have applied data mining techniques to detect number entry errors using electroencephalography (EEG) data. In their study, participants performed hear and type tasks of entering a series of 9-digit numbers. During the study, participants wore an EEG cap that measured their brain activity. Their initial classification of EEG samples indicated that EEG patterns recorded before making errors may be different from those recorded before correct entry. The potential of this research is in the predictive nature of the method which could provide error warnings to the user as soon as they occur, thus improving error detection rates.

Double data entry

Research in data entry suggests that double-entry is an effective way of reducing data entry error although at an extra time cost [Rey92, Day98]. Double entry is used in electronic forms for validating actions like setting or changing passwords where minimal visual feedback is provided to the user about data entered. Entering passwords twice ensures that the user has entered the same (intended) value correctly.

Double checking

Double checking is a policy advised by many hospital practices to reduce medication error [Ins05]. It involves the user ensuring that the data they have entered for a medication is for the right patient, using the right drug at the right dose. Double checking should ideally be performed by a different person from the one who entered the data. Although research by [Jar02] showed a comparable number of medication incidents when using a single nurse as opposed to two nurses. The main drawbacks of double checking are the extra human and time resources required to perform an independent check [U03].

Independent double checking is also not foolproof. David [U03] cites an incident where a pharmacist correctly calculated a dose requiring a volume of 0.068mL but incorrectly entered 0.68mL into the computer. Despite a second

pharmacist double checking the calculation and arriving at the correct value of 0.068mL, they still misread the incorrect volume due to confirmation bias. The failure of the double checking process is also evident in the root cause analysis of the death of a 43 year old cancer patient [Ins07]. She received, over 4 hours, an infusion medication that she should have received over four days. The wrong calculation that led to the overdose was independently performed by two different nurses.

Dose error reduction systems (DERS)

Dose error reduction systems (DERS), like Guardrails Safety Software [Esk02], are usually installed on infusion systems to manage safe ranges for drugs installed in the system. The infusion system is provided with contextual information about the treatment by allowing the user to select the name of the drug that is being infused. Some pumps have sensors that can detect the drug name from an RFID tag embedded in the syringe. With the name of the drug in place, the drug can then ensure a safe limit of treatment as predefined in the library. Although DERSs detect many infusion programming errors, it has a number of limitations. A wrong rate could still be accepted as long as it lies in a safe range. Secondly, in the case where users select drug from a list, there could be error in the choice of drug name from the library. This means a wrong range limit is enforced and could have severe consequences (especially if there is an error in number entry).

Checksums

Checksums are standard error detection schemes computed from a message and transmitted with the message. The checksum can be recomputed at a different time to verify the integrity of the data. This method can be exploited indirectly in certain types of number entry tasks where there is a mathematical relationship between the different numeric values entered. For instance, while setting up an infusion pump, it is typical to enter two numeric values representing any of *rate* of infusion, *volume* to be infused (VTBI) or *duration* of infusion. Regardless of the two parameters an interface requires the user to enter, the device can calculate the third parameter because there is a mathematical relationship between the

three values, i.e., $VTBI = rate \times duration$. From this relationship, it is also possible to detect certain number entry errors if users are required to enter all three values.

Wiseman et al. [Wis13b] explored two interface designs that used the idea of checksums to help users detect number entry errors. In one interface, users were required to enter two numbers and to verify a checksum value. In the second interface, users were required to enter two numbers as well as a checksum value. They found that participants were significantly faster when they had to check and verify the checksums than when they had to actively enter the checksum value. They also found that participants noticed all errors when they had to enter the checksum value in comparison to noticing only 36% of errors when they had to verify the checksum value. Apart from the additional time cost involved in entering redundant information, the use of checksums described by Wiseman et al. could sometime results in a false positive verification of the confirmed number. This is due to the commutativity of multiplication. For instance if *rate* and *duration* are mixed up by the user, the relationship between the numbers would still hold to verify the VTBI value since $rate \times duration = duration \times rate$.

2.5 Summary

The design of number entry interfaces date back to the invention of the earliest counting devices. Despite the long tradition of humans interacting with numbers, the majority of research has focused on the numeric keypad interface. The popularity of the numeric keypad is evident in its skeuomorphic adaptation in software as well as on touch screen devices, especially in the design of calculator interfaces.

Early research by Foley et al. [Fol80, Fol84] and Thornton [Tho79] described design alternatives for interfaces for use in setting numbers although they present no evaluations for these alternatives. There have also been a variety of studies on different aspects of the numeric keypad as well as comparative studies on variations on sequential digit entry (e.g., handwriting and gesture based systems).

To improve safety in critical environments such as healthcare, it is impor-

tant to study the effect of error, particularly in the context of interacting with numbers. This chapter has reviewed methods that are currently used to abate number entry errors in practice as well as research recommendations in the area of reducing number entry error. Most of these methods, however, apply to only one type of number entry interface. There are other ways a number entry interface might be designed. Depending on the constraints faced by a designer, for example, size constraint, it might be necessary to implement a different style of number entry interface. Research is currently lacking in an exploration of the design space for number entry interfaces. The next chapter explores the design space and proposes a classification for grouping interface styles with similar properties..

Chapter 3

Classifying number entry interfaces

A wide variety of methods exist for designing number entry interfaces. This is partly due to the wide variety of input devices available - for example, those described by Buxton [Bux83], Card et al. [Car90] and Mackinlay et al. [Mac90]. The ease with which the different measures sensed by these input widgets can be mapped to numeric quantities (e.g., a slider specifies position while a dial specifies angle) also add to the diversity of design options.

This chapter analyses the design space of number entry interfaces with respect to the different ways a number entry interface might work irrespective of the hardware or input widget used to achieve control on the interface. The context of the analysis performed is centered on interfaces found on medical devices and the design space is analysed from a high level of abstraction that deals with the question of how numbers are selected or specified in an interactive system.

The set of input widgets that might be used in the design of a number entry interface is a subset of the input widgets found in the design space of all input devices. From the taxonomy of the design space of input devices presented by Card et al. [Car91] and Mackinlay et al. [Mac90], input devices can be described based on a combination of the physical properties sensed (e.g., force or movement), and the dimension within which the properties are sensed (e.g., linear or

rotary in x, y or z dimensions or the degree of freedom sensed). Consequently, the input widgets used in the design of number entry interfaces range from buttons, sliders, rotary dials, force encoders or touch screens.

In order to start structuring the design space of number entry interfaces, this chapter employs a method of design space analysis presented by MacLean et al. [Mac91] where an artifact is explored based on the space of possibilities for the existence of that artifact. The method is based on *Questions, Options and Criteria* (QOC). In this method, *Questions* are used for structuring the space of alternatives, *Options* are possible alternative answers to the questions posed and *Criteria* are used for evaluating or choosing between the options.

3.1 A task level decomposition of number entry

The task of number entry can be viewed as one of the following:

1. Selecting a specific value from a list of options or

2. Explicitly specifying the digits that make up an intended number.

The first option implies that the user somehow selects a number from a set of predefined valid values in the application. This can be achieved by moving a virtual cursor through the number line at a rate chosen by the user i.e., by making incremental changes to the position of the cursor which in turn affects the value of the number that is to be specified in the application or by selecting a value from an explicitly presented finite set. The second option implies that the user explicitly specifies the digits that make up the number they wish to transcribe. This means the interface provides the user with some way of directly controlling the digits that make up the intended number.

The brief historical review of number entry interfaces presented in Chapter 2 shows that there are a variety of ways an interface might be implemented or designed. The list of example interfaces provided there is not exhaustive and only accounts for a subset of the possible styles of interface that might be used

for entering numbers. However, it provides a good sample from the space of possibilities.

The rest of this chapter takes a structured approach towards exploring the design space of number entry interfaces by deconstructing the design space using the QOC method.

3.2 Design space analysis using Questions, Options and Criteria

QOC design space analysis places an artifact in a space of possibilities and seeks to explain the relationships that exist between the different alternative options of an artifact [Mac91]. The role of Questions in this framework is structural and generative rather than evaluative. The Criteria should help in reasoning over the considerations involved in choosing one option in the design space over another. In other words, the Criteria provide a framework within which the different Options can be evaluated. The Criteria make the trade-offs between the different options salient in the analysis.

In the case of identifying number entry interface design options, the QOC method facilitates a structured exploration of the design space which can be expanded to an arbitrary level of detail. This flexibility enables reasoning about the design options at several levels of abstraction that might be useful for people with different roles in a design process. For instance, while designers of the physical user interface widgets would typically be interested in dealing with options at a high level, programmers who implement the logic of the interface would be more interested in the lower level detail.

In light of the task of number entry introduced earlier, features of number entry tasks that are essential to describing how the interface works are now extracted. The design space is structured by asking the question: *What does the interface control?*

There are two possible options for this question.

1. The interface might be used to control the manner in which the individual digits of the intended number are specified or

2. The interface might control a selection mechanism for choosing the intended number from a set of valid values

In the first case where the interface controls how the user specifies individual digits that make up the number, the question *In what order are the digits specified?* can be asked. For this, there are three options. The digits may be specified from left to right, or from right to left (in both cases the digits are specified in a restricted order), or the digits may be specified in an unrestricted order.

For each of these options, it is possible to explore methods for selecting digits in the number in order to further structure the design space and the question *How are the digits specified?* can be asked. This question also has two options.

1. The digits can be specified by providing interface widgets to directly specify each numeral in the base in context

2. The digits can be specified by providing widgets that can be used to navigate through a list of the numerals in the base in context

For the second case where the interface controls how the number is selected from a set of valid values, the question *How are the numbers selected?* can be asked. For this question, two options emerge.

1. The number can be selected directly, by providing user interface widgets corresponding to the intended number e.g., as seen on lift panels or cash machine user interfaces or

2. The numbers might be selected indirectly by providing 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.

3.2.1 Criteria

The criteria aspect of the analysis helps in assessing and choosing between the *Options*. Six criteria are presented. These criteria are informed by standard human performance measures for evaluating user interfaces as found in literature (e.g., [Car91, Thi10b]) and discussions with manufacturers, health care practitioners and the medical device training manager in Singleton Hospital Swansea. These criteria include the following:

- *Speed.* This refers to how quickly the interface can be used to complete the given number entry task.
- *Error Severity.* This is a measure that quantifies the magnitude of typical errors on an interface.
- *Error Rate.* How frequently errors occur while using the interface.
- *Error Detection.* The likelihood that keying slips would be noticed while using the interface.
- *User Interface Footprint.* How many widgets are required to implement the interface and consequently how much space does the interface occupy on the device.
- *Range and precision.* The minimum and maximum value that can be specified using the interface and to what number of digits different values can be specified.

Speed and accuracy are standard quantitative metrics used in assessing the usability of user interfaces [Nie92, Shn04]. Accuracy is of particular importance in the context of evaluating interfaces used in safety critical environments like health care. Consequently, half of the criteria are related to error and different aspects of reasoning about error.

For economic reasons related to the production and maintenance cost of devices, manufacturers are interested in user interface footprint. The number of physical widgets used in a device affects the production cost and this also affects

the chances that at least one of the widgets will break down and require maintenance. From a user's point of view, portability is also important, particularly in ambulatory care where devices are used by patients on the move. Portability of a device restricts the size of the device which in turn defines boundaries for an acceptable user interface footprint on the device.

The rest of the thesis seeks an informed assessment of measures that can be used to weigh different styles of number entry user interfaces against these criteria. This is achieved by performing studies that provide evidence which inform the process of highlighting the different trade-offs that exist between different example interfaces.

The analysis above has presented us with a design space within which number entry interfaces lie. In the context of the interfaces found on medical devices, these categories make it possible to compare groups of interfaces with each other. The next section provides a review of some examples from the categories of interfaces identified.

3.3 Number entry interface examples

Based on the first level options presented in the QOC analysis in Figure 3.1, two broad categories emerge. There are *digit specification* interfaces whose designs are based on the specification of digits in a number. The performance of interfaces in this group depends on the number of digits to be entered. This includes both whole and fractional digits. On the other hand, there are *number selection* interfaces whose designs are based on selecting a number from a list of options. Number entry on this style of interface requires selection (either directly or indirectly) from a set of valid options provided by the host application. This could be done indirectly by scrolling through valid values where a single selection is visible or scrolling through valid values where a range of selections are visible. It could also be done directly, by selection from a fixed set of presented valid values. The performance of the interface thus closely depends on the number of options in the valid set, and the features provided by the interface to facilitate efficient searching and selection. For interfaces based on selection of numbers from a fixed

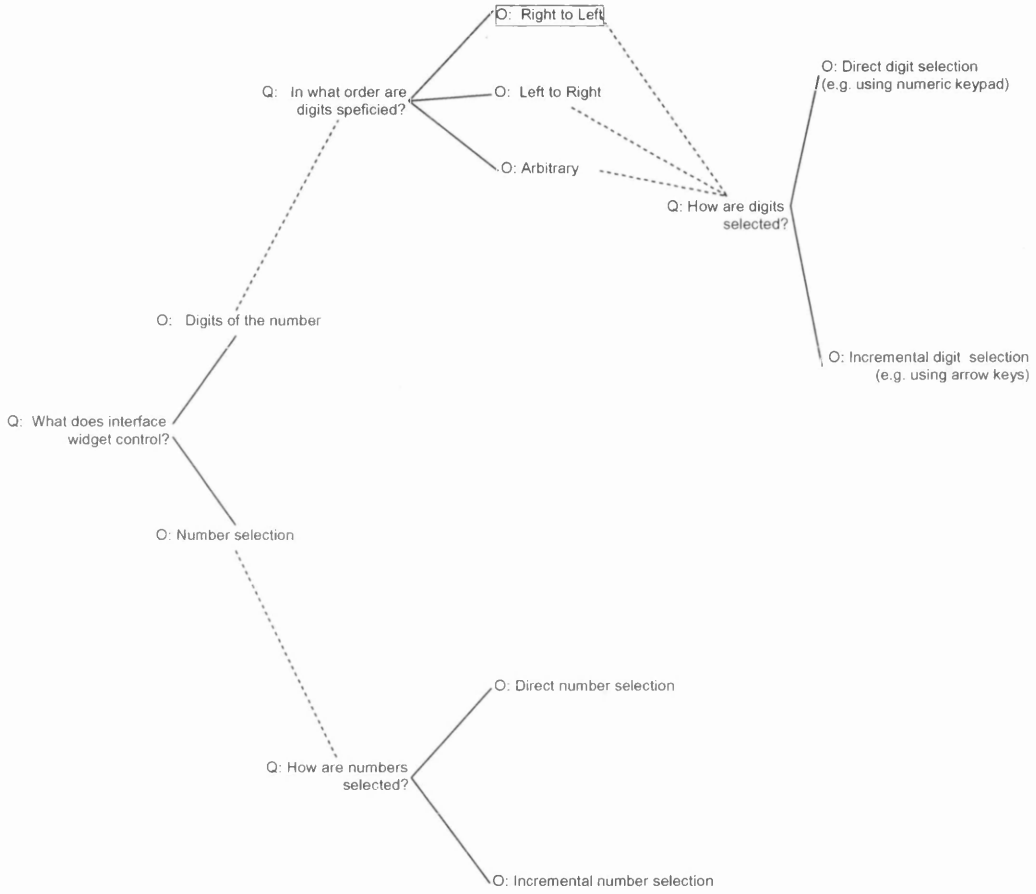


Figure 3.1: A decomposition of the number entry design space based on the QOC method. This analysis segments the design space of number entry systems into two main dimensions. Firstly, interfaces are separated into those that allow users to explicitly specify the digits that make up a number, and interfaces that allow users to select a number from a list of valid values. Digit based interfaces are then further distinguished based on the order in which they allow the specification of digits 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).

set of options that are always displayed, e.g., the control panel representing the different floors in a lift, the entry time could be modelled by a combination of the Hick-Hyman law on choice reaction time [Hic52, HYM53], where the user finds the floor they wish and Fitts' law [Fit54] where the user selects the target button that activates the intended floor. For indirect selections where users scroll through valid values e.g., the interfaces derivable from incremental number selection, it is most efficient that users approach the target number as quickly as the interface permits, but they slow down to fine-tune selection when closer to the number.

This classification also makes it possible to group interfaces in terms of their support for the occurrence of syntax errors. For example, all interfaces based on number selection always have syntactically valid input, since for these interfaces, the application presents the user with valid options from which they can choose. For interfaces based on digit specification, especially when users can explicitly specify decimal points, it is necessary for the designer to think of situations where the user enters syntactically incorrect number specifications such as entering a sequence of digits with multiple decimal points. The designer must think of appropriate ways to block these errors and alert the user.

Based on the second level options presented in the QOC design space analysis, four classes of number entry interfaces can be described. The next section provides concrete examples of interfaces from the different categories in section 3.2.

3.3.1 Serial digit entry

Serial digit entry refers to the class of number entry interfaces which affords the user control of the digits making up the required number in a fixed order from left to right. The user enters the number serially, in sequential order. This interface style is most commonly implemented using a keypad with digits from [0] - [9], a decimal point key and a clear or backspace key. If an error is made in entry, the user must delete all the numerals succeeding the erroneous numeral. Three layouts are currently in use in modern interactive devices.

The calculator layout is the default layout for calculators. It comprises a grid

of buttons arranged in three columns and it has the keys [7], [8], [9] at the top. This is also the default layout on keyboards with dedicated numeric keypads. *The telephone layout* is the default layout on telephones. It is arranged in a grid similar to the calculator layout but it has keys [1], [2], [3] at the top and [0] at the bottom row. *The single row keyboard layout* is the default layout for computer keyboards without a dedicated numeric keypad. It comprises a single row of digits arranged from [1] - [9] with [0] at the end.

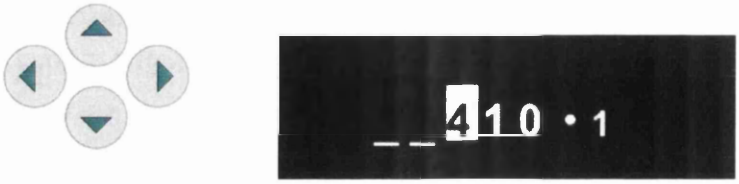
Apart from variations in key layout, this interface style might vary depending on whether or not it contains a decimal point key. When a decimal point key is absent in the layout, the interface renders a decimal point at a fixed location on the display and the user must enter values into the interface at a fixed precision regardless of whether or not the intended number is a fraction. For example, to enter '50.5' on such an interface, the user must execute the keystrokes [5 0 5 0]. This form of interface is often used on cash registers for entering monetary values.

A serial interface might also be implemented using an incremental digit selection interface as opposed to a direct digit selection interface. This might be implemented with 4 keys. Two keys for changing the digit, one key for accepting the digit and another for canceling the last digit entry.

3.3.2 Independent digit entry

Independent digit entry refers to the class of number entry interfaces which afford the user control of the digits making up the required number by specifying each digit in any order. The user is typically able to specify each digit independently and errors in a digit can be simply corrected by changing the digit. This is the oldest number entry interface style as seen in interfaces like abacuses, Pascal's Calculator and the Arithmometer.

There are two popular variations on this interface. One variation provides explicit control for all the digits on the interface and the other provides shared control for the digits and distinguishes between an active digit by enabling the user a selection mechanism that activates digits. Two examples, the *up-down*



(a) Input buttons (b) Display

Figure 3.2: The D-pad *number entry system. (a) shows the four way navigation style keys used as input and (b) shows a typical display used in the interface. The highlighted digit will change when the up or down button is clicked.*

and the *D-pad**, that feature in interactive devices such as medical devices, game pad controllers, television remote controls and combination locks, are described below.

Up-down interface

The *up-down* interface provides a pair of interface widgets for increasing and decreasing each digit addressable by the host application. The number of buttons on this interface depends on the maximum value and the precision addressable by the host application. For an application that allows entry of numbers up to 99.9, this interface will need six buttons, a pair for increasing and decreasing each digit. This style of interface negatively affects the user interface footprint since the number of buttons required to implement the interface grows logarithmically with the maximum number allowed on the interface in the best case condition where only whole numbers are permitted. In other words, the number of buttons required to implement this interface is twice the number of digits permissible on the interface. When fractional number entry is required, an extra pair of buttons is required for each decimal place required in the precision. The number of buttons required for this interface can be halved by implementing a variation where changes to the digits get wrapped around and ensuring changes made per digit are completely independent. Some examples of the *up-down* interface are shown in Figure 3.4

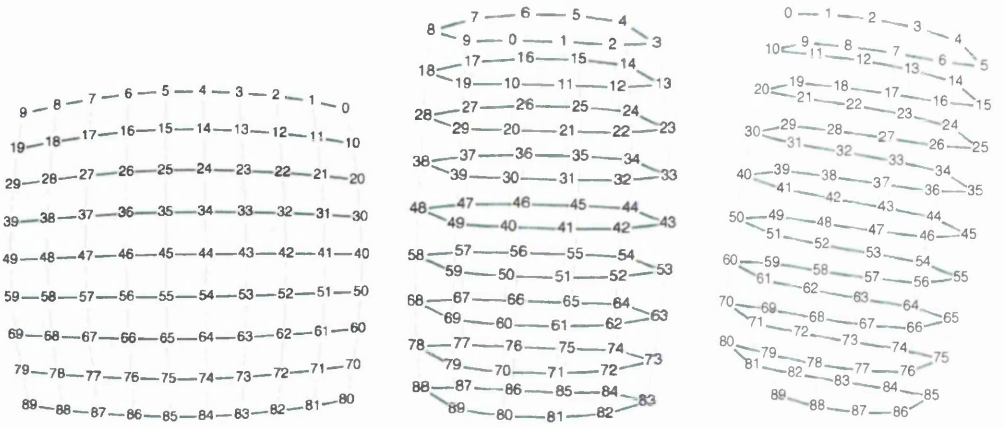
*This interface is referred to as a *five key* interface by Cauchi et al. [Cau12a, Cau12b], with the fifth key usually used for accepting numeric input.

Directional pad (D-pad) interface

The *D-pad*, also referred to as the control pad, originally emerged in the gaming industry as a user interface widget for controlling two-dimensional characters in video games. They were made to function similar to joy-sticks found in arcade games and afforded control in the up - down - left - right direction. When used as video game controllers, it is possible to hold down two adjacent buttons or sections on the controller to move in a diagonal direction. It is commonly found on mobile game controllers as well as TV and DVD remote controls for navigating menu structures on these devices. The *D-pad* provides four keys in a navigation style layout for changing numbers. A pair of left - right keys is used for moving a cursor that selects a digit to edit and a pair of up - down keys is used for making changes to the selected digit. A typical d-pad is shown in Figure 3.2. This interface requires the same number of buttons regardless of the range and precision required by the host application. As a result it is ideal for use in situations where space is limited. In addition to the variations determined by the behaviour of digits when changed around boundaries, i.e., blocking digit wrap, independent digit wrap or arithmetic digit wrap, this interface has variations based on the behaviour of the cursor. The designer can vary the cursor start position, choosing either to place it in the rightmost place value or the leftmost place value of the number.

In addition, the designer must decide what happens when a user attempts to move the cursor beyond the boundaries of the screen. They must decide whether such actions are blocked or whether the cursor movement wraps around to the opposite end of the display. Cauchi et al. [Cau12b] present results of detailed analyses about the effect of these different configurations on error severity.

From these examples, it can be seen that independent digit interfaces are commonly implemented by providing widgets for incrementally specifying each digit, although in full keyboard interfaces like the Comptometer, the interface might have individual digit keys for each place value in the number. When digits are set incrementally, the behaviour of this style of interface varies significantly depending on how certain features are implemented. For instance, the designer must decide precisely how digit changes are implemented in the system, particu-



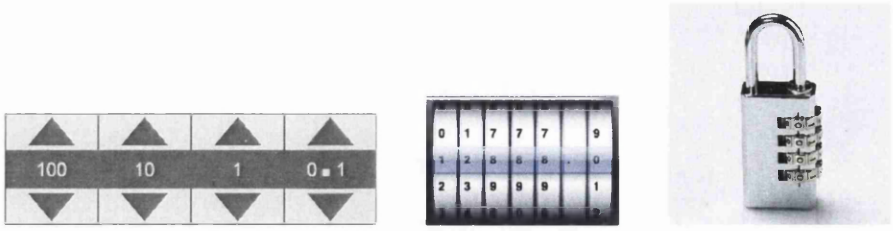
(a) Blocked Digit Wrap (b) Independent Digit Wrap (c) Arithmetic Digit Wrap

Figure 3.3: These graphs visualise the effects of three variation on the digit wrapping feature for independent digit interfaces. From left to right the effect of blocking digit wrap, independent digit wrap and arithmetic digit wrap is shown. The graphs highlight the structural differences in the interaction experiences that are manifest in the different variations. The user can change between numbers that are directly connected by a line using one key press.

larly when the digits change from [0] to [9] or vice versa. Three variation on this feature are described below.

Blocked digit wrap

This variation disallows the change from [0] to [9] or from [9] to [0]. This means the user action is effectively blocked at these boundaries. Using a slider widget to implement the digit controller on the interface would achieve this effect since a slider widget has physical limits to setting a minimum and maximum value. The control of digits in this variation are completely independent of other digits. Figure 3.3a shows the upper and lower bounds of interaction in this variation of the interface. Horizontally, interactions are performed from 0 to 9 on each decade and the same is true vertically.



(a) Up down (b) Number wheels (c) Combination lock

Figure 3.4: Three forms of the up-down *number entry system.*

Independent digit wrap

This variation allows the digits to wrap around, that is, incrementing [9] changes it to a [0] and decrementing [0] changes it to a [9]. The wrapping effect does not carry over to the digit to the left. The control of digits in this variation are also independent of other digits although the implementation of the interface can be best visualised if one imagines the digits are controlled by dial widgets with continuous rotation. Figure 3.3b shows the independent structure of each decade (or place value) in this variation. This structure can be directly compared to physical interfaces that are built with this property. Examples are show in Figures 3.4c and 3.4c where each wheel on the interface can be manipulated independently of the other wheels.

Arithmetic digit wrap

This variation allows the digits to wrap around with the wrapping effect carried over to the digit to the left. This means one numeric value is added or removed from the digit to the left when a digit is increased from [9] to [0] or decreased from [0] to [9] respectively.

3.3.3 Incremental number entry

Incremental number entry interfaces refer to the class of number entry interfaces that allow the user to select a number from a valid set of options by providing an indirect mechanism of selection from the list. Typically, the user navigates through an ordered list of options and can go forward or backwards in the list.

The interface may also allow the user to control the speed of navigating through the list. This interface may be implemented using widgets such as buttons, sliders, dials or any widgets found in Mackinlay's [Mac90] design space of input devices.

An example incremental interface based on the use of buttons might require 4 buttons (see Figure 7.2b). Two buttons are used for increasing the number and the other two are used for decreasing the number. In each case, one of the buttons causes a change an order of magnitude larger than the other. A single button with a pressure sensor might also be used where the pressure level of the button controls the amount of change caused to the number.

In general, an incremental interface can be a zero order or position control where there is a proportional relation between user action and numeric output such as a slider or a mouse. It can also be implemented as a first order control where user action (such as displacement) is proportionally related to the rate of change of the numeric value. In this case the magnitude of the user action determines the magnitude of change or the rate of change of the output number. This control order might be implemented using a spring-loaded dial. It is also possible for an incremental interface to be implemented as a second order control. In this case, user action such as displacement is used to control the magnitude of acceleration (i.e., user action changes the rate of change of changes in numbers). Users must actively decelerate to stop numeric changes in this style of control. Although second order controls are more difficult, people can become skilled at it with practice [Jag03].

Gain and Time-delay

In the control aspect of incremental number entry interfaces, two parameters would influence the stability of the user interaction in a typical number entry task. *Gain* determines how the input signal is transformed to the output signal. In the

Figure 3.5: An example incremental style interface found on an infusion pump.

context of number entry, *gain* affects the speed at which the user approaches the target number (i.e., the reference signal). *Time-delay* refers to the time taken by the system to act on user input and provide feedback to the user. This is also referred to as the *latency* of the system. Together, these two parameters determine the speed of the system and stability of the system (i.e., the likelihood that users would overshoot or undershoot targets in number entry).

When *gain* is low, the system responds slowly, but when it is high, the system is likely oscillates about the target signal. Similarly, a high time-delay increases the likelihood of oscillatory behaviour [Jag03, Doh99].

In the context of the design of number entry interfaces, then, the different combinations of these parameters and the widgets that afford continuous control make for a large set of styles of incremental number entry interfaces. A review of these different combinations is beyond the scope of this thesis.

3.3.4 Direct number selection

This style of interface allows the user to select a number from a valid set of options by providing a direct selection mechanism that allows the user to choose the required number. This interface style is feasible in situations where there are a small set of numeric values to choose from, e.g., as seen in lift floor selection interfaces. Though this interface offers fast entry, its application is limited to use when there is enough space on an interface to render all the different options without the need to navigate through different screens that shows the available options. 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.

3.3.5 Limitations of the classification

The classification presented in this chapter is inspired from examples of number entry interfaces found through out the history of old mechanical calculators and a variety of number entry interfaces found on infusion pumps in use in hospitals. Moreover, the exploration of the design space presented provides a high level de-

composition of possible ways in which numeric digits, and consequently numbers (or a string of digits) can be manipulated.

The current classification does not address issues related to the base of the number entered. For example, it is possible for the user interface to map user actions to manipulate a number in a base that is different from the base that the resulting number is displayed. An interface with three buttons comes to mind, where one button is used to append the binary digit [1], another is used to append the binary digit [0] and the last button is used to undo the last move (if there is any) or reset the number back to zero. If the display of this interface renders the numbers entered in binary, then according to this classification, the interface would be a serial interface. If however the display renders the number in a different base such as decimal, it is not so clear where the interface sits with respect to this classification. It could be thought of as an incremental interface with two types of increments, where pressing one button doubles the current number, and pressing the other button doubles the current number and adds one.

Similarly, this classification does not address issues related to the perceptual representation and interpretation of numbers. It covers the active aspect of user interaction that deals with inputting numbers on interfaces and is applicable in discussing and analysing a myriad of existing number entry interfaces including those found on medical devices.

3.4 Summary

The design space of number entry interfaces has been explored based on whether the interface offers controls over digits or numbers and based on whether the control is offered in a direct or incremental manner. This analysis led to four groups of existing interface styles, that is, serial digit entry, independent digit entry, incremental number entry and direct number selection.

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 earlier is not limited to

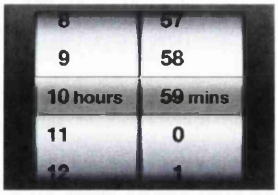


Figure 3.6: iPhone interface showing incremental control for groups of digits.

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. An example is the iPhone interface for entering time where control is not digit based but based on groups of digits which are set incrementally. Figure 3.6 shows an example.

This chapter has presented a classification for number entry interfaces based on two main features. The first feature deals with what aspect of numbers a user controls. The interface may be used to control the entire number as a whole or it may be used to control parts of the number. The second feature deals with the manner with which the interface allows control over the first feature. The number or the number part may be controlled incrementally or directly. Although there are potentially limitless ways of combining the actual input widgets that are used

			Example Devices
Digits	Right to left	direct digit selection	
		incremental digit selection	
	Left to right	direct digit selection	Calculators, Telephones, Microwave ovens
		incremental digit selection	Kollector
	Arbitrary	direct digit selection	Comptometer, Burrough's Add Lister
		incremental digit selection	Millionaire, Arithmometer, Pascal's Calculator, D-pad
Number		direct number selection	Lift panel
		incremental number selection	Alarm clocks, Microwave ovens, Infusion pumps

Table 3.1: The position of various examples of historical and some current number entry interfaces in the design space.

to implement examples from each class of interface, this grouping encompasses all existing number entry interfaces. New interfaces can be created by composing two or more interface styles in a single interface and by segmenting the number into parts which are at least one digit long. These parts can then be controlled using any of the interface styles described.

Table 3.1 shows the position of various interfaces in the design space for number entry systems. Although there are no examples for right to left entry of numbers, that is where the interface restricts the entry of numbers to be from the least significant digit to the most significant digit, there is, in principle, no reason why interfaces can not be designed in this way.

The next chapter analyses the performance of different instances of digit based interfaces.

Chapter 4

Numbers in context

The design of an application or an interactive system that requires numeric input involves the specification of ranges of valid numeric input to the (host) application. This might involve specifying some function that translates user actions into numbers. As seen in Chapter 3 the interface style might implicitly produce syntactically valid numeric input as defined by the application, thus eliminating any possible syntax errors in entry.

This chapter explores the types of numbers used in infusion therapy in hospitals using logs from three different infusion devices. Specifically, it explores the ranges, precision and typical changes of the numbers during therapy. This analysis provides the basis for selection of the numbers used in the tasks performed in the experiment described in Chapters 6 and 7.

4.1 Related Work

The letters of the alphabet in most languages have a varying frequency of occurrence in the vocabulary the language. For instance in the English language, the letter 'e' has the highest frequency of occurrence [Lee99, p 181]. Letter frequency models such as these have various applications ranging from use in security systems for decrypting certain types of cyphers or in user interface design for building error correction models in text entry.

Similarly for numeric data, Benford [Ben38] observed that the frequency of digits in naturally occurring numbers are not evenly distributed. Specifically, he observed that the frequency of the first significant digit of a number closely follows a logarithmic function that is defined on the digit. The digit *1* is more likely to occur than digit *2* which is in turn more likely than digit *3* and so on. This is contrary to the distribution one would expect if all the digits of numbers had an equal probability of occurrence.

Wiseman et al. [Wis12] explored the notion that the frequency of digits used in infusion therapies in hospitals are not evenly distributed by analysing 58 log files taken from Graseby 500 [gra02] infusion pumps in a hospital. Their analysis, showed that the most common digits used when setting numbers in infusion pumps are *0, 1, 2, 5* and *9*. Their analysis combined numbers set as *rate* of infusion i.e. the speed at which to deliver the drug and the *volume to be infused (VTBI)*. Since infusion volumes tend to be delivered in preset bags of 1000, 500, 250, 125 and 50 millilitres, the frequency of the digits *0, 1, 2, and 5* are not surprising. Based on this, they provided recommendations that would allow more efficient number entry for the set of numbers that appeared in the logs.

This chapter presents results from the analysis of logs from three different infusion devices with the aim of understanding the nature of numbers used in infusion therapy. This involves exploring features such as the typical range of numbers entered, the precision of numbers, the range of numbers within a given precision as well as exploring how numbers change within an infusion therapy. For accurate results, only numbers that have been entered by a user are included in the analysis. For instance rate values that occur after infusion has completed and Keep Vein Open (KVO)* mode has been activated have been filtered out.

To achieve this level of accuracy, the event logs were first parsed and separated into sets representing different infusions. The manner in which this was done varied for each pump. For the Graseby and the Asena pumps, the beginning of a new infusion was inferred from the log whenever a start infusion event was found

*KVO is a special mode in infusion devices that is triggered after an infusion has completed. In this mode, the device continues to deliver medication at a predetermined low rate e.g., 1mL per hour, in order to keep the patient's vein open for subsequent delivery of medication.

and the *volume infused* attribute was 0. For the BBraun pump, there was a *new rate set* event which designated that a user had changed the infusion rate of the device.

4.2 Data

The data used in this study were from three infusion devices used in Singleton Hospital in Swansea and the Royal Free Hospital in London. All the logs were anonymous and did not contain any patient identifiable information. The Graseby 500 is a volumetric infusion pump produced by Smiths Medical. It has two 7-segment displays for showing the rate and the VTBI of an infusion and it has an input control panel containing a numeric keypad. The Asena Syringe pump is produced by Alaris Medical Systems. Its control panel contains four chevron keys used for entering numbers and navigating list menu items. The BBraun infusomat is a volumetric infusion pump produced by B.Braun Medical. Its control panel contains a D-pad style interface (4 navigational keys) used for number entry and navigating list menu items.

The log files contain some similar features and concepts. An *event* refers to any significant change in the status of an infusion device such as when the device is turned on or off, when it starts or stops infusion, when infusion settings are changed or when the device alarms for any reason. An *attribute* is a property representing a setting on the device at any given point (for instance when an event occurs). An *attribute* may be the rate, VTBI, alarm volume or time of event. None of the logs contained explicit information about the start or end of infusion therapies. Where necessary, this information has been inferred retrospectively from the log data by parsing the sequence of events for those that signify new infusion parameters as well as the start and end of infusions. Below is a detailed description of the settings recorded by each pump.

4.2.1 Graseby 500

The Graseby logged the most recent 200 events that occurred and maintained a cumulative frequency of all events that have occurred on the device. With each

event record, the attributes representing the device state at the time of the event are logged. The attributes of interest to this study are all the rate values in the logs whenever an event indicated the start of a new infusion. This pump did not log keystroke information.

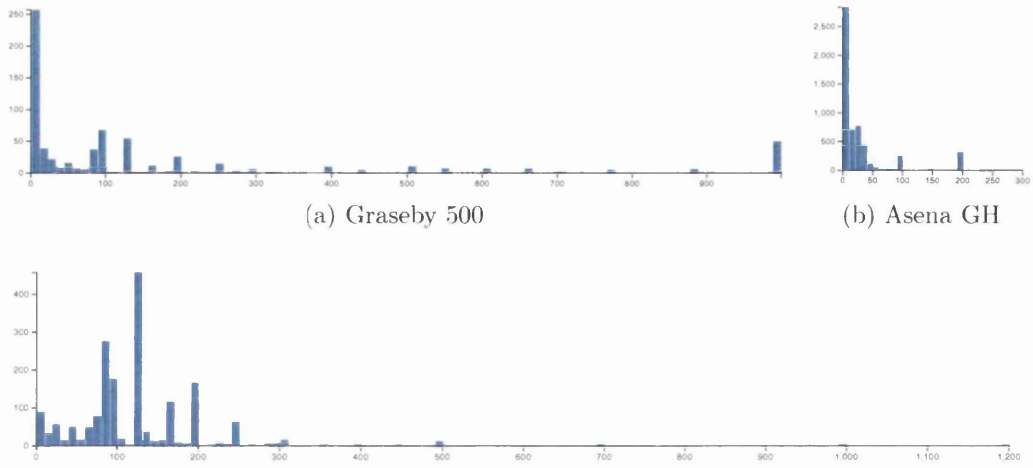
A total of 268 log files taken from 127 infusion pumps were used in the study. The pumps were from a variety of wards and departments in the Singleton Hospital in Swansea. These logs have been previously analysed by Lee et al. [Lee12b] to explore the time and monetary costs of the events and alarms causing interruptions to the workflow of healthcare practioners. They have also been analysed by Monroy et al. [MA13, Lee12a] to explore the discrepancies between intended infusions and actual infusions. The analysis in this chapter builds on the work reported by Monroy et al. specifically by using the blocks of intended infusions derived in their analyses. In total, the logs represented 3,681 infusions which account for about 9,048 hours of infusion therapy.

4.2.2 Asena GH Syringe Pump

The Asena GH syringe pump kept a log of the last 1500 events as well as the last 200 key strokes. Each event had a description which contained values where appropriate. For instance a start infusion event description contained the starting rate and VTBI. Each event also had a date attribute which had a time component that was precise to the nearest second. The keystroke logs contained a date accurate to the nearest second and the name of the key pressed. Sixty files containing event logs from 60 syringe pumps in the Singleton Hospital were analysed.

4.2.3 BBraun Infusomat

The BBraun pump kept a log of the last 1000 events as well as the last 200 key strokes. Each recorded event had a description of the event, the value and unit of attributes that were set during the event (e.g., for rate and volume values) and a time attribute that was precise to the nearest second. The keystroke logs contained the name of the key that was pressed, the display mode of the pump when the key was pressed and the time of the action, precise to the nearest



(a) Graseby 500

(b) Asena GH

(c) Bbraun infusomat

Figure 4.1: These histograms show the frequency distribution of rate values set on each of the infusion pumps.

second. The keystroke logs were not used in this analysis. Ten log files containing event logs from 10 infusion pumps from the Royal Free Hospital in London were analysed.

The next section presents details of the nature of numbers used in these three infusion devices.

4.3 Distribution of numbers

4.3.1 Range

Graseby

For the Graseby pump, the rate values ranged between 1 and 999. A total of 705 rate entries were extracted from the logs. Fifty-six percent of these entries ranged between 0 and 100 and seventy-six percent ranged between 0 and 200. Figure 4.1a shows a histogram with a detailed distribution of the values.

Asena

The Asena pump had the smallest range with all values occurring between 0.1 and 300. A total of 5,578 rate entries were extracted from the logs and 90% of these were less than or equal to 100. Figure 4.1b shows a detailed frequency distribution of the values.

BBraun

The rate values entered into the Bbraun infusomat ranged between 0.1 and 1200. A total of 1825 rate setting entries were extracted from the BBraun logs. Thirty-six percent of these entries were of values that ranged between 0 and 100 while 83% of values ranged between 0 and 200. The histogram in Figure 4.1c shows a detailed distribution of the values.

4.3.2 Precision

The precision of a number refers to the position of the rightmost significant digit in the numbers entered into the devices. For example a value of 34 is precise to the *Units*, 300 is precise to the *Hundreds* and 50.5 is precise to the *Tenths*.

From Figure 4.2, it can be seen that the majority of values entered into these devices are whole numbers. It also shows the percentage distribution of the precision of rate values entered into the infusion pumps. For each pump, over 80% of numbers were precise to the units, tens or hundreds. A further break down of the distribution of numbers for each pump shows variation of precision for different ranges in numbers.

Asena

For the Asena pump, numbers precise to the hundredths place value only ranged between 0 - 10, with majority of values occurring between 0 and 2. There is a similar distribution for the tenth place value with the exception of an outlier value where 100 is entered to one decimal place.

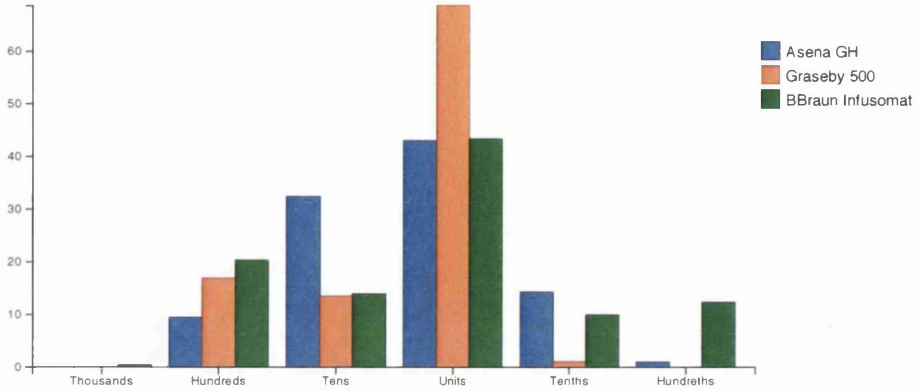
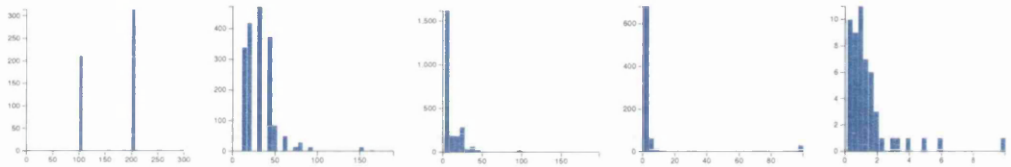


Figure 4.2: The percentage of numbers entered into each device to a precision level shown on the x axis.

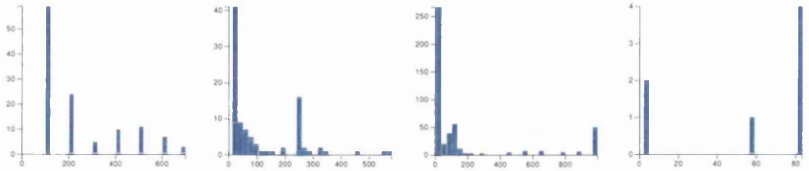


(a) Hundreds (b) Tens (c) Units (d) Tenths (e) Hundredths

Figure 4.3: Frequency distribution of the precision of numbers entered in the Asena GH pump.

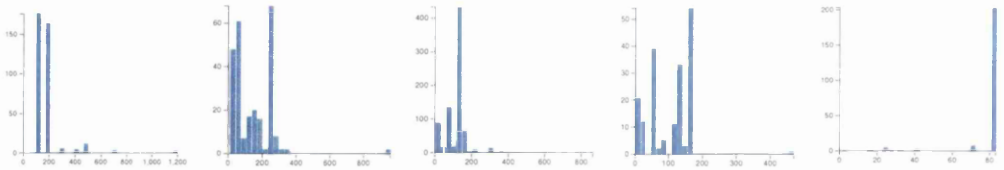
Graseby

Analysis of the distribution of precision of numbers from the Graseby shows that about 1% of numbers were entered to the precision of two decimal places although the range was not as clearly confined as the Asena. Occurrences of numbers precise to the units, tens and hundreds had a wide range although most



(a) Hundreds (b) Tens (c) Units (d) Tenths

Figure 4.4: Frequency distribution of the precision of numbers entered in the Graseby 500 pump.



(a) Hundreds (b) Tens (c) Units (d) Tenths (e) Hundredths

Figure 4.5: Frequency distribution of the precision of numbers entered in the BBraun infusomat pump.

values occurred between 0 and 200.

BBraun

The BBraun pump had the highest proportion of numbers (20%) entered to a precision of tenths or hundredths.

4.3.3 Differences in the distribution of numbers

The distribution of the numbers entered into the three devices analysed in this study show a snapshot of typical rate settings used as programming parameters in these devices. The apparent differences might be due to the following:

Different sources. The log data were obtained from different NHS trusts, therefore, the differences in the numbers might reflect the differences in the practices across these sites.

Different types of devices. The logs analysed were from two different types of infusion devices. One device, the Asena GH, was a syringe pump. In a syringe pump, drug is placed in a syringe and the syringe is driven by an internal motor which controls the precision of the delivery of the drug. The two other devices (Graseby 500 and BBraun infusomat) were volumetric pumps. Volumetric pumps control the flow rate of drugs placed in a bag above the pump using peristaltic or cassette mechanisms to manage to rate at which drug is delivered to a patient [Pox04]. According to the Medicines and Healthcare Produces Regulatory Agency (MHRA) [Med13], volumetric infusion pumps are the preferred choice for high volume drugs at medium

Figure 4.6: Changes in the rate values on the Asena syringe pump sorted by the starting value of each change. The blue lines are initial changes, the red lines are subsequent changes where the number has been reduced and the green lines are subsequent changes where the number has been increased.

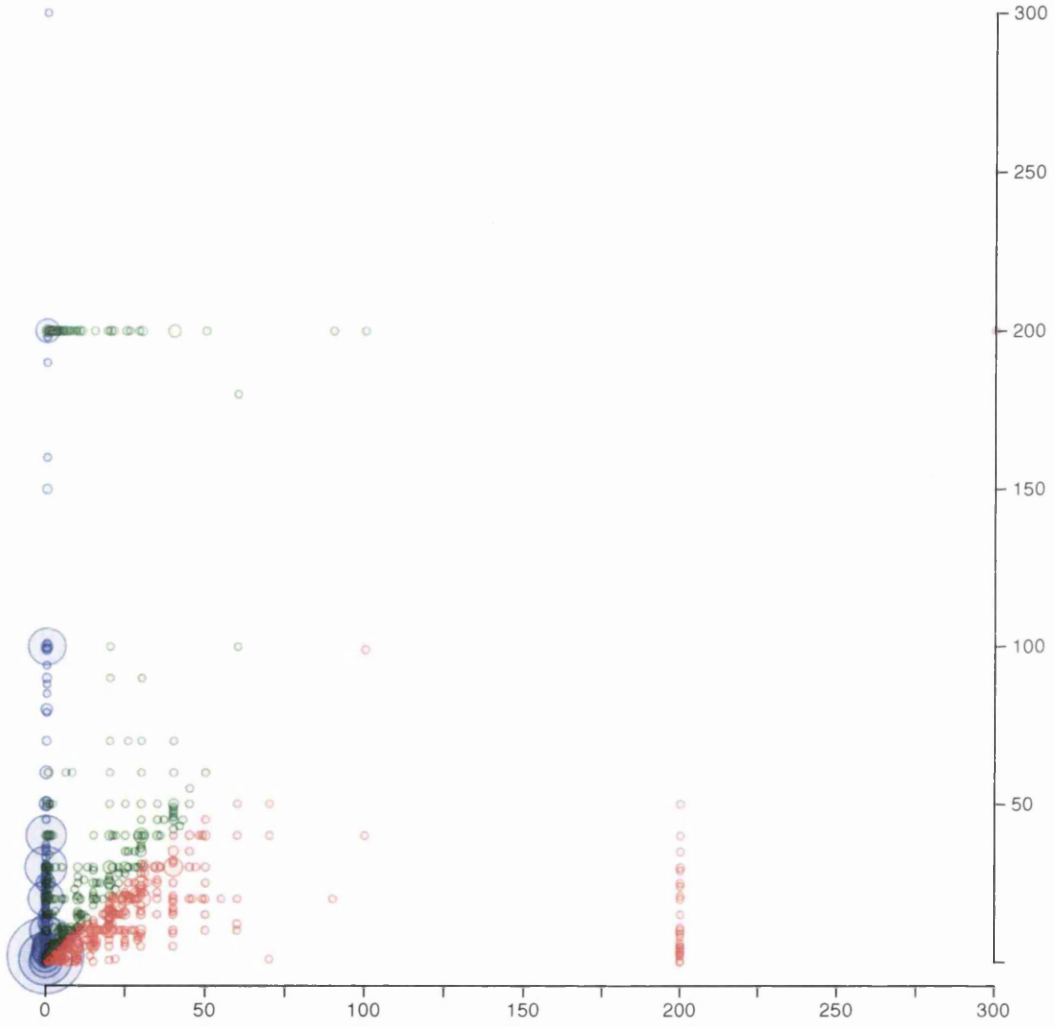
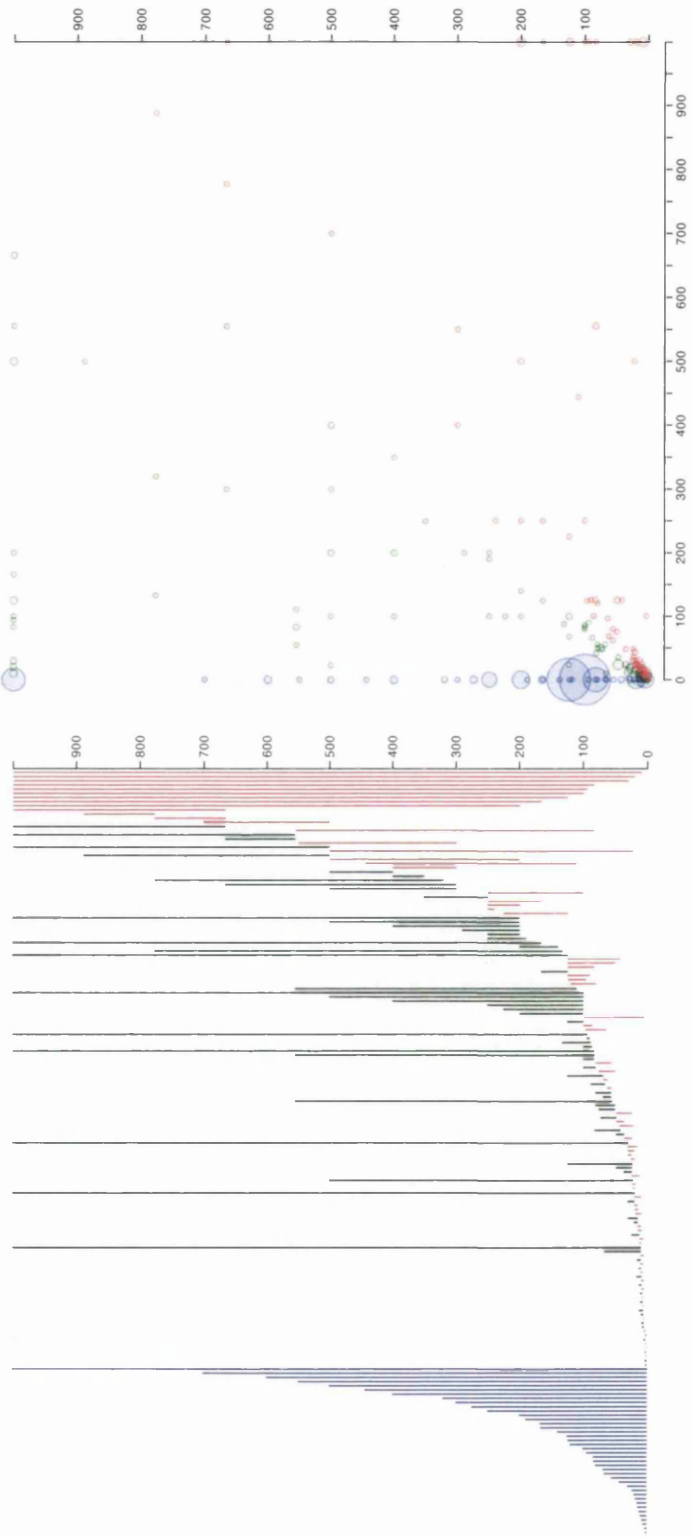


Figure 4.7: The number changes that occurred on the Asena syringe pump and their frequencies. The x-axis are source numbers while the y-axis are target numbers. Initial changes are coloured blue. Subsequent changes that are increments are coloured green while changes that are decrements are coloured red. The relative frequencies of the change is encoded in the size of the circles. Bigger circles represent changes in numbers that occur more frequently.



(a) Graseby rate changes

(b) Graseby rate changes with frequencies

Figure 4.8: All rate changes according to logs from the Graseby infusion pump and the frequency of changes.

(a) BBraun rate changes shows that most number entered into the device were entered from a starting value of zero.

(b) BBraun rate changes with frequencies shows the most frequently set rate value as 125

Figure 4.9: All rate changes according to the logs from the BBraun infusion pump and the frequency of changes.

and large flow rates. Syringe pumps are preferred when delivering low volumes at low flow rates. These preferences are evident from the range of the values seen in the logs as the two volumetric pumps have higher maximum rates (1000 and 1200) than the syringe pump (200).

Different wards and therapies. The logs analysed were obtained from different pumps and even the pumps from the same hospital are potentially used in a variety of wards over a time period. Different wards in each hospital use different therapies depending on the condition of the patients in the ward or departments. Pumps used in the surgical ward might often be set to deliver drugs as quickly as possible to cater for a patient in an emergency situation. This requires using high rates.

Amount of available data. The pumps log information at different level of detail and have different policies for the amount of log history they keep. In addition, the frequency at which the logs are downloaded from the pumps in different trusts could also cause differences in the snapshot captured by the logs and consequently what data is available for analysis.

4.4 Number changes and frequency of changes

Figures 4.6, 4.7, 4.8, 4.9 show an overview of how numbers change based on the analyses from three infusion pumps. Two types of number changes are identified. There are changes where numbers are set from zero to the target number. These will be referred to as *initial* changes and there are changes where numbers are set from non-zero values to the target numbers. These will be referred to as *subsequent* changes.

For the BBraun pump, 94% (N=1721) of all rate entries were *initial* changes. Figures 4.9 shows that most numbers lie along the y-axis where $y = 0$. The most frequently set value was 125. The graseby pump had 45% (N=186) *initial* changes with a most frequently set value of 100. Finally, the Asena pump had 68% *initial* changes with a most frequently set value of 0 to 2.

The low number of *subsequent* changes in the BBraun logs could be due to a feature that allows users to explicitly start a new therapy. When this feature is activated, all pump settings such as rate, volume to be infused and duration are cleared and set to zero when a therapy is completed. The user has to enter new values when starting a new therapy. Another possibility is that users made less error on this device and did not need to make subsequent changes to the values that they entered.

4.4.1 Differences in number changes

The visualisation of the number changes found in the logs of the three devices highlight some features that are worth discussing. For the Asena pump, Figure 4.7 shows that the value 200 is a prominent source and target for changes of numbers. Further discussions with the medical devices training manager in the hospital where this pump is used suggested that this rate is typically used to deliver antibiotics from a syringe over a relatively short period. For instance, a 50ml syringe set at a rate of 200ml per hour would deliver drug for 15 minutes. It also emerged that 200ml per hour is the maximum rate allowed on this device and could also be used to deliver a bolus to the patient.

There is a similar trend visible in the changes found in the Graseby logs as the value 999 seems to be the source and target of many changes in the logs (see Figure 4.8b). This value can be used as a short "flush" to clear any drug that remain in the infusion line when medication is being changed. In some cases, this rate can be used to deliver drug to a patient as quickly as possible in order to replace lost fluids in an emergency.

This trend was not evident in the BBraun logs, potentially due to the lower number of files analysed from the device.

4.5 Number changes and Dose Error Reduction Systems

As discussed in Chapter 2, Dose Error Reduction Systems are installed on many modern infusion devices to put predefined limits to the dose values set for therapies. They work by allowing the user to select the name of the medication used in the therapy from a list in a predefined drug library. These systems then enforce soft and hard limits which have been predefined for each drug in the library to catch high severity errors in infusion therapies. Breaching a soft limit usually triggers a warning that alerts the user about the limits. Interactions that attempt to breach a hard limit are blocked and alerted to the user.

When a drug has been selected and the details about the patient have been entered, some devices automatically calculate a default value, which is dependent on the parameters of of the patient and the type of drug chosen from the drug library. The user then needs to make changes to a default value which is specific to the dose value in the prescription chart.

Setting numbers from a non-zero value has implications for the performance of different interfaces. These performance differences will be investigated in Chapter 5 where the task times for changing and setting different number combinations are explored.

4.6 Summary

From the logs of the infusion pumps, most numbers entered into the BBraun pump are initial changes (i.e., they are set from a value of zero). This is different for the other two pumps. The number of tasks that are based on subsequent changes to a number (rather than setting new numbers) has implications for the performance of the interface. Sometimes only small adjustments are needed when making changes to a number. This is especially true for treatment regimes where the patient has to be constantly monitored to see how they respond to medication. An example is in the delivery of medication such as insulin, where blood glucose levels determine the rate of drug delivery. In such cases, the serial

style interface does not offer the most efficient interaction since users first have to clear some preceding digits or all the numbers before entering or updating the number.

The next chapter analyses at a keystroke level, the time cost of changing between any pair of numbers in the range 0-99 for different interfaces. The numeric tasks visualised in Figures 4.6, 4.8a and 4.9a are also used in the next chapter to estimate the time costs of performing those tasks on a variety of interfaces.

Chapter 5

Modelling task performance in number entry

Based on the interface classes identified in Chapter 3, this chapter presents a quantitative analysis of task performance (with respect to speed) for number entry interfaces that offer digit level control. It uses the Key-stroke Level Model (KLM), developed by Card et al. [Car83, Car80], to estimate the task completion time for an expert user to perform a series of number entry tasks.

Card et al. presented KLM as a simple model for predicting the time it takes a user to perform a task on an interactive computer system. The model estimates time, based on a user's low level key press operations which constitutes time taken to press a button (K), time taken to point to a target (P), time taken to move the hand to the keyboard or other device (H), time taken for mental preparation (M) and the time taken for the system to respond (R). The Keystroke-Level Model only deals with the time aspect of performance and is not suited to analyses of performance pertaining to *errors, learning curve, or fatigue.*

Errors can be modelled and analysed using methods such as *stochastic key slip simulation (SKSS)*, where user actions are simulated and injected with errors with varying probabilities [Cau12a, Cau12b]. SKSS proceeds on the assumptions that errors will eventually happen and perform analyses on the effects of error as

error rate or probability is varied. Since KLM predicts time, KLM can also be used in conjunction with other error modelling techniques, (e.g., for predicting the cost of error recovery).

While errors are a very important aspect of the design of interactive systems, particularly those related to the design of safety critical interactive systems, the principal purpose of this chapter is a predictive evaluation of the speed of number entry interfaces. Other dimensions pertaining to errors and subjective user preference are evaluated empirically and reported in chapters 6 and 7.

5.1 Interfaces analysed

Chapter 3 introduced different styles on number entry interfaces. For an independent digit style interface whose digits are controlled incrementally, there are variations on how the wrap around feature on the digits might behave. Performing usability studies for all the variations would be very expensive.

In order to reduce the space of possible variations of interfaces discussed in Chapter 3, this chapter conducts initial evaluations of the performance of seven interfaces. It analyses, one variation of the numeric keypad interface and three variations (*arithmetic, digit wrap and blocked digit wrap)* of the *D-pad* and the *Up-down* interfaces as described in section 3.3.2).

5.2 Estimating speed

Correct number entry on digit based interfaces such as the numeric keypad, is straightforward. It requires the entry of the digits that make up the number by successively pressing on the corresponding key on the keypad. On interfaces like the up-down or the *d-pad*, digits may be entered in any order although it might be more efficient to perform digit entry in a specific order and it would probably be common that people enter numbers from the most significant digit to the least significant digit - a reflection of the way numbers are spoken in western languages.

For digit based interfaces, the time required to enter a number is a function of the number of digits in the intended number. For whole numbers, the speed of entry is a function of the logarithm of the intended number. For fractional numbers, the speed of entry is a function of the number of decimal places and the logarithm of the whole part of the number.

The following definitions will be used in the rest of the chapter for consistency in referring to various aspects of the definitions of performance in the different interfaces.

$\boldsymbol{\delta(d)}$	represents the cost of entering a single digit d
$\boldsymbol{\alpha}$ or $\boldsymbol{\beta}$	are used to represent numbers to enter $\{\alpha, \beta \in \mathbb{Z}^+\}$
$\boldsymbol{m}$	represents the number of digits in α or β
$\boldsymbol{C(\alpha, \beta)}$	represents the cost of changing α to β
$\boldsymbol{\alpha_i}$	represents the i^{th} digit in α where α_0 is the rightmost digit in α
$\boldsymbol{g}$	represent cost function used in the A^{star} algorithm
$\boldsymbol{h}$	represents the heuristic function used in the A^{star} algorithm

In general, if z is the number of decimal places in α, and m is defined as $\lceil \log_{10} \alpha \rceil + z$, then the cost of number entry on an independent digit interface can be expressed as:

$$C(0, \alpha) = \sum_{i=1}^{m} \delta(\alpha_i) \tag{5.1}$$

δ can be further expanded into two components that account for the time s_k to select a key on the interface and the number of key presses k_n required to set the required digit. In general, s_k is a target selection task and can be modelled by Fitts' law [Fit54]. For the numeric keypad, k_n can be approximated by a constant value per user as it represents button activation time. For instance the button activation approximation value (K = 200ms), from the Keystroke-level model can be used. For the *up-down* and *d-pad*, k_n can be approximated as a function of average number of key presses required to set any digit from $\boxed{0}$. If p_d represents the probability that digit d will occur in a given number, then the mean cost of setting an arbitrary digit on an independent digit interface that

allows *digit wrap around* can be approximated using:

$$\sum_{i=1}^{9} p_i \min(|10 - i|, i) \tag{5.2}$$

This value is about 3.22 if all non-zero digits had an equal probability ($\frac{1}{9}$) of occurrence. More precise values can be obtained by using a probability distribution that closely models the context of analysis. For general purpose numbers, the probability distribution of digits in numbers as presented by Benford [Ben38] can be used. For more specific contexts, such as predicting the performance of interfaces for use in infusion pump therapies, relevant probability distributions should be derived from frequency analyses of log data, for instance as suggested in [Wis13a] or as shown in Chapter 4.

5.2.1 Improving the interface performance

Part of the aim of this analysis is to highlight aspects of the design of these number entry interfaces where performance can be improved. From this deconstruction, one can improve entry speed by improving key selection time s_k. This can be achieved by increasing button size to enable faster target acquisition. Another way to improve the performance by reducing the time and effort required to set a digit k_n. This can be achieved by using widgets such as dials or sliders for digit selection.

5.3 Method

Since the purpose of this analysis was to discover how quickly, with respect to time, a given interface allows a user to enter numbers, the following question summarises the aim of the chapter: *For any two pairs of numbers α and β in a range defined, how long would it take a user to change α to β on an interface?*

Given a graph representation of an interface, such as the one implicitly specified in a programmatic implementation of the interface, the path in the graph joining α to β is a function of the cost of changing α to β on that interface. As an example, in order to change 0 to 35 on the interface described in Section 3.3.2,

which blocks digit wrapping, multiple paths might be taken. Figure 5.1 shows a state transition diagram where nodes represent all the possible numbers that can be set on the interface and the edges represent discrete button clicks that change one number to the other. For this interface, edges along the same row or column can be navigated using the same button. Paths through this interface can be traced by connecting the edges in the graph. The source and target nodes in this trace represent α and β. Each corner in the paths signify a button change on this interface. Button changes should be minimised to optimise the task completion time. Figures 5.1b and 5.1c shows two equally optimal ways to change 0 to 35, since they both only require one button change. Figure 5.1d shows a less optimal path, which requires three button changes.

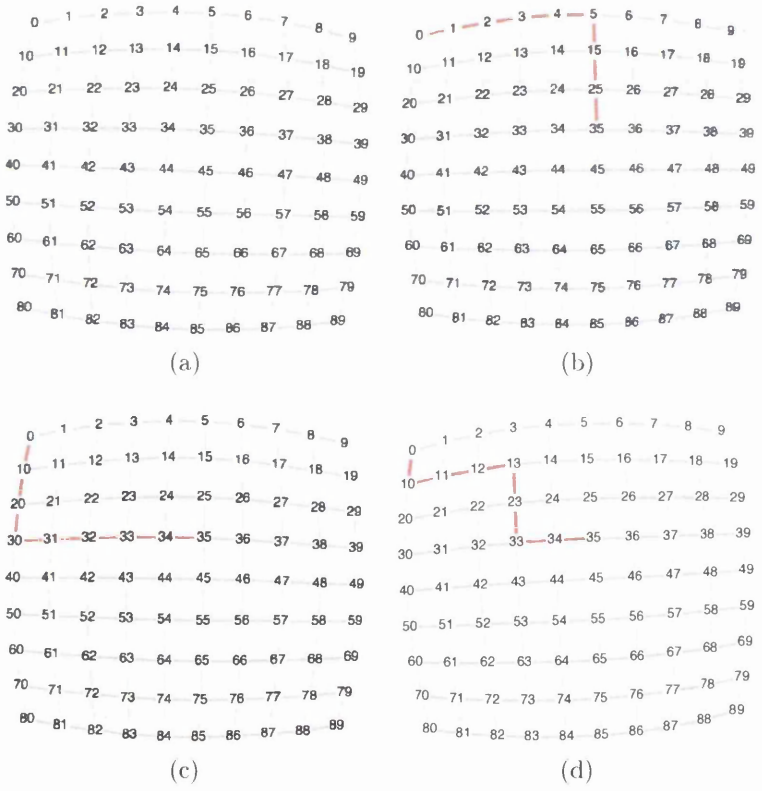


Figure 5.1: (a) shows a state transition diagram for an interface that offers independent digit control with digit wraparound blocked. The interface visualised here allows entry of integers between 0 and 89. (b), (c) and (d) are different paths from 0 to 35 on this interface.

The user interface model discovery process will be used to extract the graph representation of each interface and the A^{star} algorithm will be used to find the path between two nodes, i.e., the source and target numbers, in the graph.

5.3.1 UI Model Discovery

The model discovery process extracts a graph that represents an interactive system by systematically exploring the state space of the system. This is done by stimulating the actions of a user on the system while keeping track of the states encountered in the process and stopping when the state space is exhausted. A generic application programming interface (API) is presented by Gimblett and Thimbleby [Gim10] and various applications of the method are presented in [Thi09b, Thi09a, Gim10].

Some important features are crucial to accurate model discovery. The discovery process must have an accurate view of state in the interactive system and it must have the ability to set the state in the system correctly and must have the ability to perform actions on the user interface.

For the numeric keypad, the discovery process could perform digit click actions for all digits 0 - 9 as well as a backspace key to delete the last digit clicked. The decimal point key could not be clicked. The state on this interface was defined by the value property on the interface.

The *up-down* interface could perform four actions. Unit up and Unit down were used to increase and decrease the digits at the units place value and ten-up and ten-down were used to increase and decrease the digits at the tens place value. Similar to the numeric keypad, state on this interface was defined by the value property on the interface.

The *d-pad* interface could perform four actions. Left and right buttons moved a cursor left or right to select a place value, while up and down buttons increased or decreased the selected digit. The state on this interface was defined by the cursor position and the value property on the interface.

For all variations of interfaces tested, the maximum value allowed was 99 and the highest precision allowed was the units.

5.3.2 Path finding algorithms

Path finding is a method in graph theory used to find a path between any two nodes in a graph (if one exists). It is commonly used in network analysis to find a route between two locations and also used in the design of artificial intelligence for games, for instance in helping a game character navigate around obstacles. One of the most common path finding algorithms is Dijkstra's algorithm [Dij59, Cor09]. For a given graph with non-negative edge weights, this algorithm finds the lowest cost path between any two nodes contained in the graph. Starting from a source node, Dijkstra's algorithm recursively traverses adjacent nodes starting with the node that has the least cost path from the current node. The algorithm keeps track of the path costs between the source node and all subsequent nodes traversed until the target node is found. Dijsktra's algorithm is guaranteed to find a shortest path for any two nodes that exist in a graph. It however runs slowly for large graphs due to its time and space complexity. A more efficient algorithm is the A^{star} path finding algorithm.

The A^{star} Algorithm

The A^{star} algorithm improves on Dijkstra's algorithm by using a heuristic function as well as the path cost so far to estimate how far the current vertex is from the source [Har68]. The A^{star} algorithm has two main components. A *cost function* is used to calculate the cost incurred in traveling from the source node to the current node while traversing the graph and a *heuristic function* is used to estimate the cost of traveling from the current node to the target node. The heuristic function allows the customisation of search through a graph by allowing us to create functions that specify how the algorithm perceives proximity to the goal state during the search process.

For the purposes of the evaluation in this chapter, two different cost functions will be explored. Cost will be defined as a function of the number of button clicks required and as a function of time required. To help define these functions, the next section describes mathematical functions that can be used to obtain the costs of specifying numbers on the different variations of interfaces in the analyses.

5.4 Determining the optimal keystrokes for specifying numbers

As a prerequisite to running the A^{star} algorithm, 4 mathematical functions for use in defining the heuristic function for A^{star} are presented. One function will be used for the numeric keypad and 3 functions for each variation of the independent digit interface.

5.4.1 Numeric keypad

For the numeric keypad, the number of keystrokes required to specify a number is a function of the number of digits in the number. Hence the cost can be derived as specified in equation 5.1.

5.4.2 Blocked digit wrap

For the *blocked digit wrap* variation, changes in any digit in the number is completely independent of other digits and increments to digits are capped at $\boxed{9}$ while decrements are capped at $\boxed{0}$. Consequently, the cost of changing α to β on this variation of the interface style is the sum of the cost of changing each subsequent digit in α to the corresponding digit in β. In other words, it is the sum of the absolute difference in the digits of α and β.

$$C(\alpha, \beta) = \sum_{i=0}^{m} \mid \alpha_i - \beta_i \mid \tag{5.3}$$

5.4.3 Independent digit wrap

For this variation, changes in any digit in the number is also completely independent of other digits and changes to digits wrap around such that a $\boxed{9}$ when increased becomes $\boxed{0}$ and a $\boxed{0}$ when decreased becomes $\boxed{9}$. The wrap around feature means that setting a high number when the digit is $\boxed{0}$ can be achieved by navigating backwards through the digit lists. For instance setting an $\boxed{8}$ can be achieved from a $\boxed{0}$ by pressing down twice on the digit. This reduces $\boxed{0}$ to $\boxed{9}$ and then to $\boxed{8}$. The optimal cost, $\delta(d)$ of setting a digit d on this style of

interface is therefore the smaller of incrementing to that digit or decrementing to that digit.

$$\delta(d) = \min(d, 10 - d)$$

Therefore, the cost $C(\alpha, \beta)$ of changing α to β can be defined as:

$$C(\alpha, \beta) = \sum_{i=0}^{m} \delta(|\alpha_i - \beta_i|) \tag{5.4}$$

5.4.4 Arithmetic digit wrap

For the *arithmetic* variation, the digits wrap around and the wrapping effect causes a change to the digit to the left (i.e., the effect of changes on a place value carries over to the place value to the left whenever a digit is decreased from $\boxed{0}$ or increased from $\boxed{9}$). Interaction on this variation of the interface performs incremental changes to the entire number. Consequently, increasing the *units* place value ten times is equivalent to increasing the *tens* place value once. This arithmetic behaviour allows the definition of the cost of changing α to β to be a function of the difference between α and β. Intuitively, the cost can be thought of as the sum of the number of unit changes required to change α to β, which is essentially the numerical difference between α and β.

If $C(\alpha, \beta)$ is the total number of unit steps between α and β, then

$$\begin{aligned} C(\alpha, \beta) &= \alpha - \beta \\ &= \alpha - \beta - 0 \\ &= (\alpha - \beta) - 0 \\ &= C(\alpha - \beta, 0) \\ \text{Hence } C(\alpha, \beta) &= C(\alpha - \beta, 0) \end{aligned}$$

In general, the problem of the cost of changing α to β can be rewritten to the cost of changing 0 to $\mid \beta - \alpha \mid$ if $\alpha < \beta$ or the cost of changing $\mid \alpha - \beta \mid$ to 0 if $\alpha > \beta$.

The cost of entering a number α from an initial value of zero, can be represented as the sum of entering the individual digits that make up α. Initially, it

can be observed that the cost $\delta(d)$ setting a single digit d on this interface is:

$$\delta(d) = \begin{cases} d & \text{if } d \leq 5 \\ 11 - d & \text{otherwise} \end{cases}$$

The first condition stipulates that for digits d up to 5, one can simply increase the place value d times. For digits greater than 5 however, there is a more efficient entry strategy that becomes feasible as a result of the wrap around feature of the digits on the interface. Hence, similar to the wrap around variation in Section 5.4.3 one can decrease the digit from $\boxed{0}$ to get to the higher numbers (i.e., those greater than 5). However for this variation, it is necessary to account for the action needed for correcting the carry over effect that occurs whenever the digit wraps around, hence $11 - d$ as opposed to $10 - d$. To calculate the cost of entering a number, two utility functions S_c and S_f are defined. Listings 5.1 show JavaScript implementations for S_c and S_f. For a given number α, these functions return another number that is rounded up or down to the next higher place value. For instance:

$S_c(343)$	$\rightarrow 350$	$S_f(343)$	$\rightarrow 340$
$S_c(350)$	$\rightarrow 400$	$S_f(340)$	$\rightarrow 300$
$S_c(400)$	$\rightarrow 400$	$S_f(300)$	$\rightarrow 300$

It has been already been established that the cost of changing β to α is the same as the cost of changing $\beta - \alpha$ to 0 where $\alpha < \beta$. Therefore to calculate the cost of changing $\beta - \alpha$ to 0, a recursive function $C(\beta - \alpha, 0)$ which evaluates $S_f(\beta - \alpha)$ and $S_c(\beta - \alpha)$ until both functions no longer change the value in the parameter is defined. Listing 5.2 shows a JavaScript implementation for the cost function. This process creates a graph from node $\beta - \alpha$ to node 0 with each node in the graph leading to two other nodes. The paths in the graph are weighted and indicate the cost of traveling between the adjoining nodes. The total cost of changing $\beta - \alpha$ to 0 is thus defined as the path within the graph with the least cost.

```
/**
 * @param x integer
 * @returns integer
 */
function nextceil(x) {
    var m = numDigits(x), i, ceil, pow = Math.pow(10, m - 1);
    for (i = 0; i < m; i++) {
        pow = Math.pow(10, i);
        ceil = Math.ceil(x / pow) * pow;
        if (ceil !== x) {
            return ceil;
        }
    }
    return x;
}

/**
 * @param x integer
 * @returns integer
 */
function nextfloor(x) {
    var m = numDigits(x), i, floor, pow = Math.pow(10, m - 1);
    for (i = 0; i < m; i++) {
        pow = Math.pow(10, i);
        floor = Math.floor(x / pow) * pow;
        if (floor !== x) {
            return floor;
        }
    }
    return x;
}
```

Listing 5.1: A JavaScript implementation of function S_c and S_f

```
function cost(x) {
    var nextc = nextceil(x), nextf = nextfloor(x);
    if (x === nextc || x === nextf) {
        var d = digit(x, numDigits(x) - 1);
        return Math.min(11 - d, d);
    }
    return Math.min(cost(Math.abs(x - nextc)) + cost(nextc), cost
   (Math.abs(x - nextf)) + cost(nextf));
}
```

Listing 5.2: JavaScript function for calculating the number of clicks required to change x to 0

Let $x = \beta - \alpha$

$$C(x,0) = \begin{cases} \delta(\frac{x}{10^{m-1}}) & \text{if } x == S_f(x) \text{ or } x == S_c(x) \\ \min(C(x - S_f(x)) + C(S_f(x), 0), & \text{otherwise} \\ \quad C(S_c(x) - x) + C(S_c(x)), 0) & \end{cases}$$

For the interfaces analysed in this chapter, the functions defined above give a quantitative representation of the number of button clicks needed to change between any two numbers. In order to perform Key-stroke Level Model analyses, it is necessary to determine the sequence of buttons clicked on the interface in the course of changing between the numbers. This sequence of button presses were obtained by path finding using different cost functions.

5.4.5 Cost as a function of number of button clicks

One can reason about the performance of an interface in terms of the number of clicks required to complete a given task. Discrete key clicks of user interactions give a very good indication of the number of steps required to perform or carry out a task on a user interface. The functions defined in the previous section are used for calculating the cost function with respect to number of clicks required to set the values.

5.4.6 Cost as a function of estimated time

Another way to reason about interface performance is in terms of the optimal time required to complete a given task. This value cannot simply be obtained from the optimal number of clicks required to complete a task because the time required to click a button varies depending on the button. For instance, button acquisition takes time. This means it costs more time to click on two different buttons than it does to click twice on the same button. This detail is lost when thinking of performance in terms of number of button clicks alone. It is straightforward to estimate the time required to enter a number given a finite number of clicks.

Using KLM to estimate the performance component in the cost function, produces time optimised paths through the graph that are different from those

```
1  /**
2   * calculate KLM cost for performing a sequence of actions
3   * @param actions Array of strings
4   */
5  function klmcost(actions) {
6      var point = 1100, click = 200;
7      if (!actions || actions.length === 0)
8          return 0;
9      var score = click, action = actions[0];
10     if (actions.length > 1 && action !== actions[1]) {
11         score += point;
12     }
13     return (score / 1000) + klmcost(actions.slice(1));
14 }
```

Listing 5.3: JavaScript code for calculating estimated time

generated when the search is optimised for least number of button clicks. Using optimised time as a cost function allows the discovery of number entry interaction strategies that are different from those that appear to be intuitively optimal with respect to the number of clicks required to specify numbers. These strategies are arguably closer to what a non-expert user of the interface might use since figuring out the optimal keying sequence takes mental preparation as well as time required to acquire different buttons.

Using the A^{star} algorithm, the rest of this chapter explores the time cost of entering numbers on a variety of key based number entry interfaces as well as the cost of changing numbers within a given range.

5.4.7 Cost function g

In the A^{star} algorithm, g is used to calculate the cost of traveling from the source to the current position in the graph. As such, g is defined over a sequence of edges traversed so far (in our case actions performed so far). Simply, g returns a KLM estimation of the time taken to execute the sequence of actions between the source node and the current position in the graph.

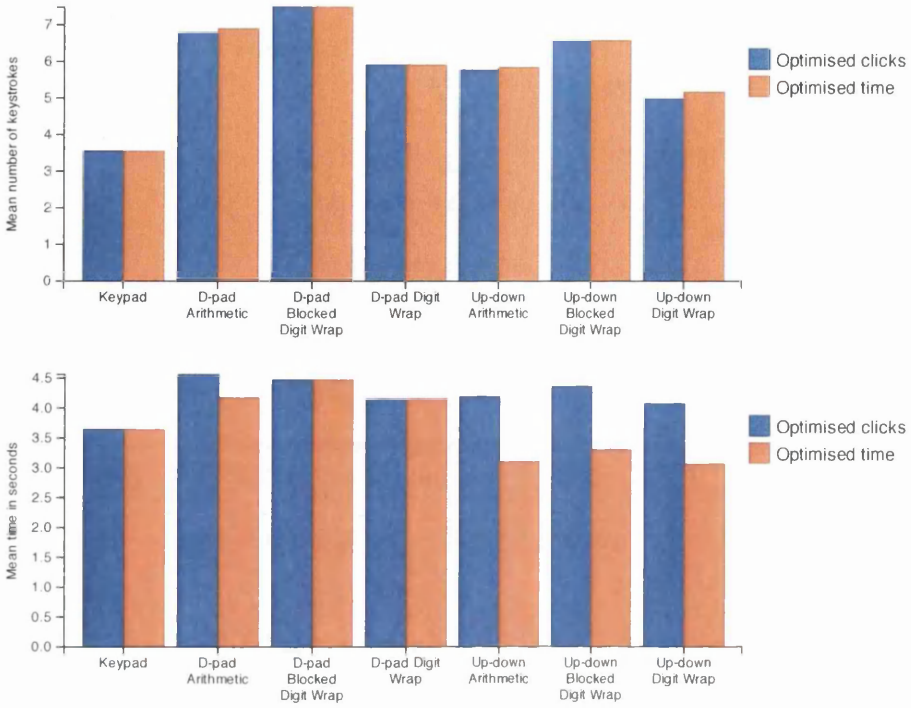


Figure 5.2: (Above) A comparison of the number of keystrokes required to change between any numbers in the range of 0 - 99 for all interfaces, when task strategy was optimised for minimum number of clicks versus minimum elapsed time. (Below) A comparison of mean time required to change between any two numbers when task strategy is optimised for least keystrokes versus least time.

5.4.8 Heuristic function h

The heuristic function provides an estimate of the cost from the current state to the goal state, i.e., from the current number to the intended number. The heuristic function essentially determines the strategy that the analysis appears to use while exploring the performance of the interfaces. The accuracy of the path that the analysis discovers is dependent on the accuracy of the heuristic function in predicting the distance to the goal state. The heuristic functions used in the A^{star} algorithm are based on those defined in Section 5.4.5 for calculating the number of clicks needed to change α to β. However instead of simply using the number of clicks, the function is updated with the KLM functions for estimating time to acquire buttons and time to activate buttons. Listing 5.3 shows a JavaScript function for estimating time given a sequence of actions.

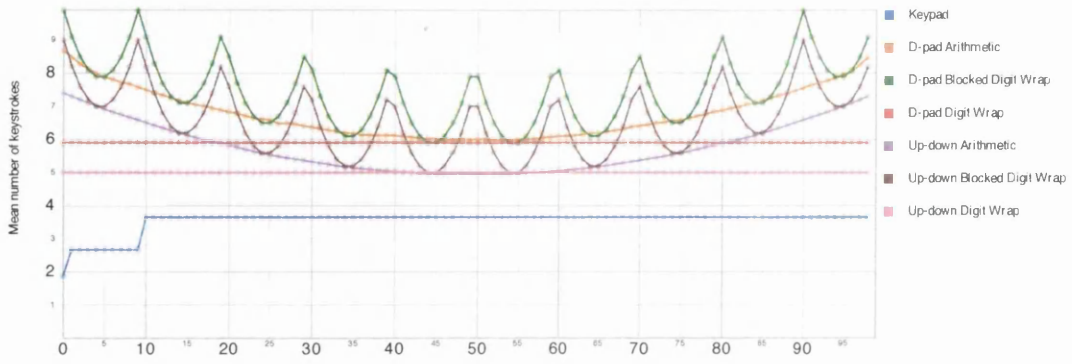
5.5 Results and Discussion

The A^{star} algorithm was run for all 7 interfaces for numbers ranging from 0 - 99. This was used to find a path between all pairs of numbers within this range as stipulated by the cost and the heuristic functions. For each interface, the cost of setting a fresh number, i.e., setting a number from zero, and the mean cost of setting a number from any other number is reported. Figure 5.2 shows the mean costs (both keystrokes and time) of changing between any numbers for the range between 0 - 99. Figures 5.3 and 5.4 shows detailed averages of cost of changing from each number in the range to the other numbers and Figures 5.5 to 5.11 shows an adjacency matrix representing the cost of changing between any two numbers in the range.

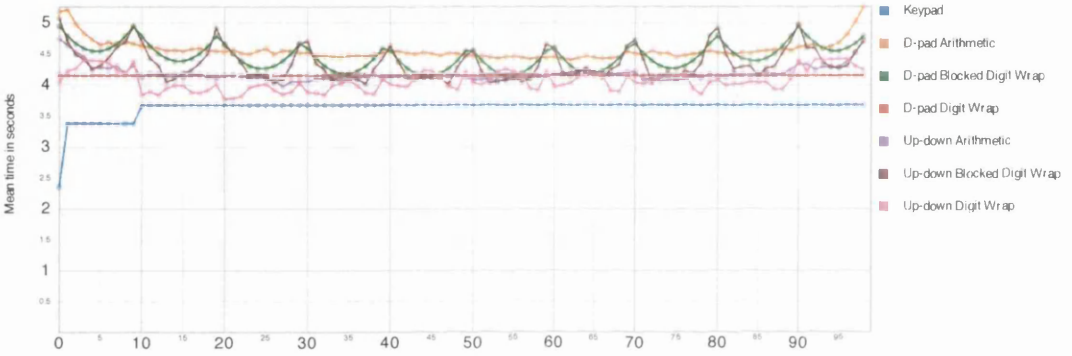
5.5.1 Numeric Keypad

For the numeric keypad, there was no difference between the strategies uncovered when the task performance was explored for optimal number of keystrokes or optimal time. In both cases, the average number of clicks required to change zero to any other number was 1.9 with a mean time of 2.4 seconds and the overall mean number of clicks need to change any numbers from any other number was 3.5 clicks with a mean time of 3.6 seconds. The numeric keypad in this analysis uses a clear digit action rather than a clear number action for error correction. Figure 5.3 shows the relationship between the mean cost of changing a number to any other number and the number of digits of the source number. The longer a number is, the further away it is from any other number. An extra digit in the input sequence increases the task time by 200ms. This can be seen, for instance, in the step increase for mean time costs for numbers above 9.

The all-pairs costs shown in Figure 5.5 shows that performance on this interface is symmetrical about the diagonal. This means that the cost of changing α to β on this interface is the same as the cost of changing β to α. The main diagonal simply shows that there is no cost involved when $\alpha = \beta$. The ten lighter square grids spanning the diagonals show that the cost of changing between values that lie in the same decade (tens group) is the same.



(a) Mean number of clicks to change numbers to any other number



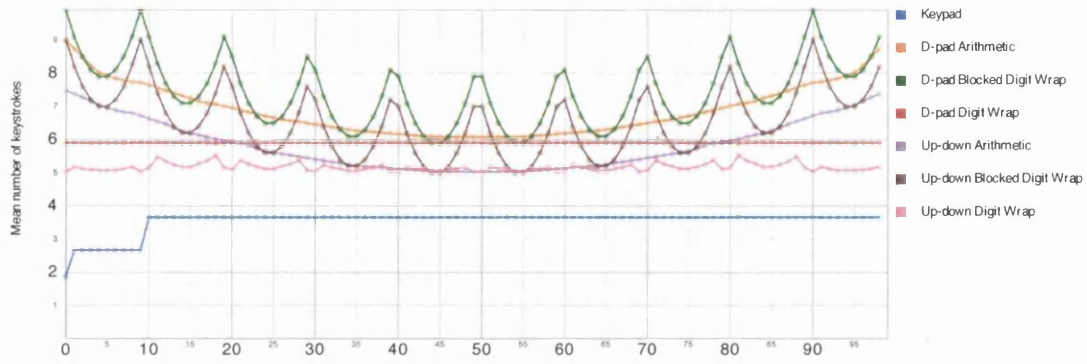
(b) Mean duration to change numbers to any other number

Figure 5.3: This shows averages for cost of number entry on 7 interfaces. The costs are keystroke optimal. 5.3a shows average number of clicks for each number on the x-axis to all other numbers within the range 0 - 99 and 5.3b shows a time estimate of cost based on the optimal keystrokes.

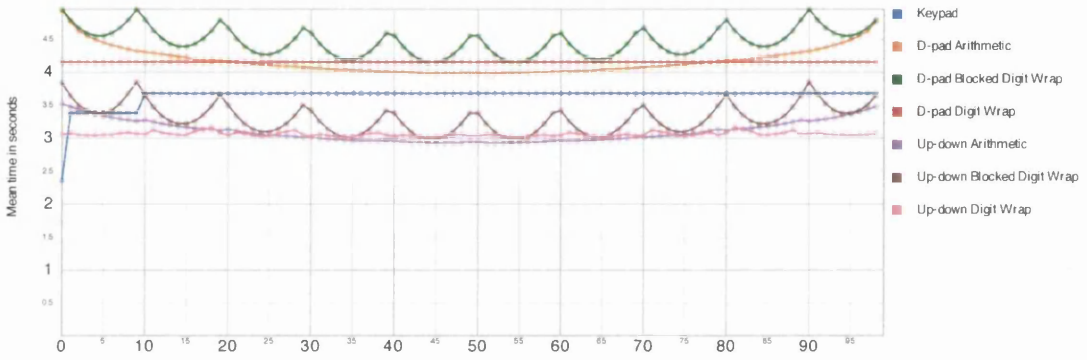
5.5.2 D-pad

The *arithmetic* variation of the D-pad interface required an average of 8.7 clicks to change zero to any other number with a mean time of 5.2 seconds. The overall mean distance between any two numbers was 6.8 clicks over 4.6 seconds. When task strategy was optimised for time, the mean number of keystrokes to change zero to any number increased to 9 although the time required dropped to 4.9 seconds. The overall distance between all pairs was 7 clicks lasting for a duration of 4.2 seconds.

Figure 5.6 shows that the performance of this *arithmetic* variation of the D-pad interface is symmetrical about the diagonal. The light diagonal bands in the



(a) Mean number of clicks to change numbers to any other number

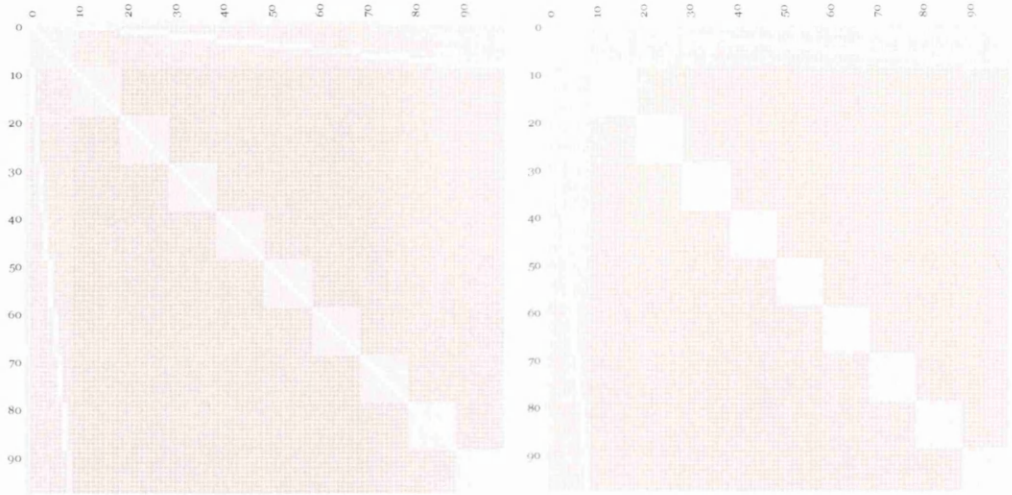


(b) Mean duration to change numbers to any other number

Figure 5.4: The costs here are time optimal. 5.4a shows average number of clicks for each number on the x-axis to all other numbers within the range 0 - 99 and 5.4b shows a time estimate of costs.

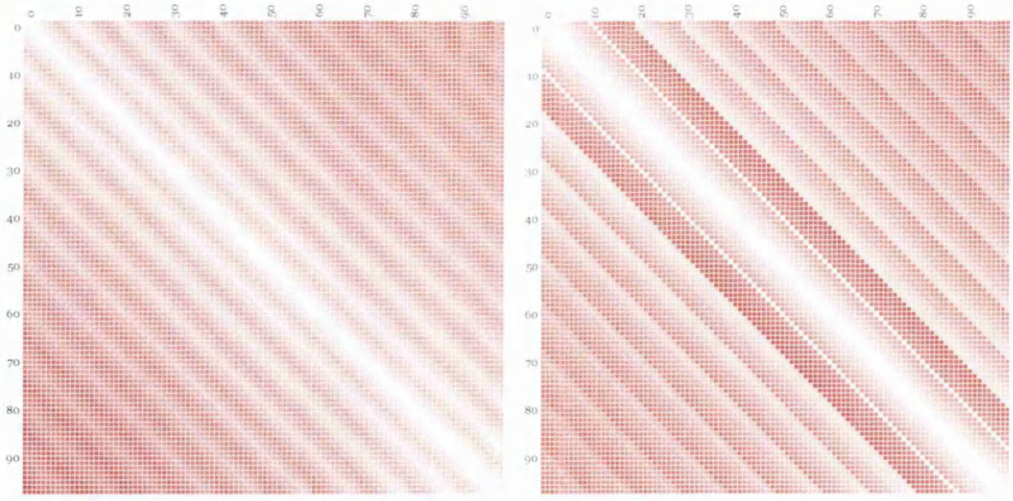
matrix show that the cost of changes made between two numbers are a function of how numerically close the numbers are as well as how many digits they share in common. Figure 5.6b shows a visible difference in cost by highlighting points in the matrix where the arithmetic feature becomes beneficial to use. This point occurs beyond the seventh unit of every decade. Instead of pressing repeatedly on the up key, it becomes more cost effective to move the cursor to the left, increase the digit, move the cursor right (to return to the initial position) and decrease the digit to 9 or 8.

The *blocked digit wrap* variation, like the numeric keypad, produced the same results for both strategies. On average, it required 10 clicks to change zero to any other number with a predicted mean duration of 5 seconds. The mean number



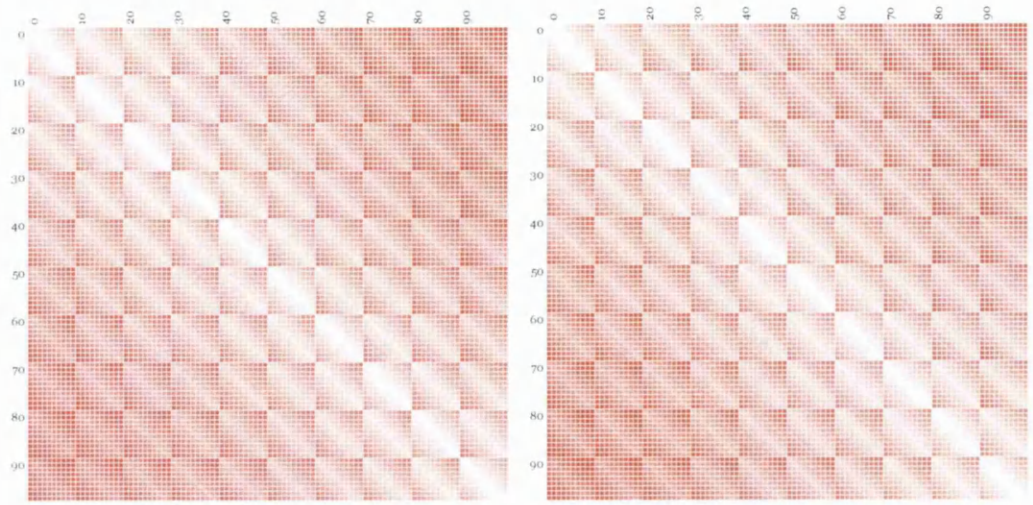
(a) cost as number of keystrokes (b) cost as predicted duration

Figure 5.5: A matrix diagram showing relative cost of changing between all pairs of numbers in the range in the analysis for the numeric keypad.



(a) cost as number of keystrokes (b) cost as predicted duration

Figure 5.6: The relative costs of changing between numbers for the arithmetic variation of the D-pad interface.

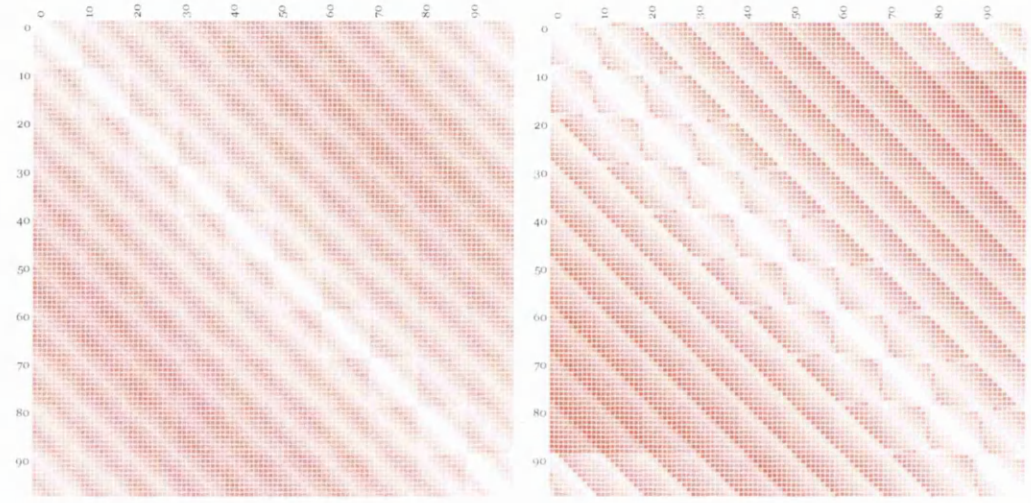


(a) cost as number of keystrokes (b) cost as predicted duration

Figure 5.7: The relative costs of changing between numbers on the D-pad interface when digit wrap is not allowed.

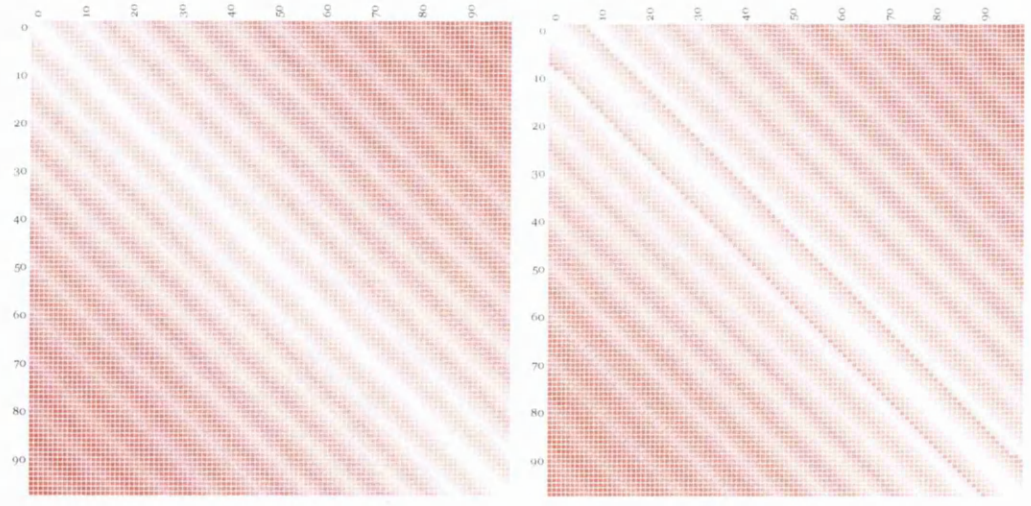
of clicks between any pair of numbers was 7.5 lasting for 4.5 seconds. Note that, on average when setting fresh numbers, this variation requires more number of clicks than the *arithmetic* variation but requires less time. This is because when digit wrap is blocked, the user is forced to set each digit explicitly by continuous increments or decrements. This means the user typically performs monotonic changes to each digit successively. This requires more keystrokes, but means that the user is performing clicks on the same button, rather than using two buttons to change the same digit. In the standard KLM, switching buttons, is equivalent to clicking on the same button 5 times. The way this variation works essentially ensures that users employ a time optimal strategy.

Similarly, the *digit wrap* variation for this interface produced the same results when explored with both strategies. It required an average of 6 keystrokes to set a number, with an average task time of 4.2 seconds. The overall distance between all pairs was also 6 keystrokes lasting 4.1 seconds.



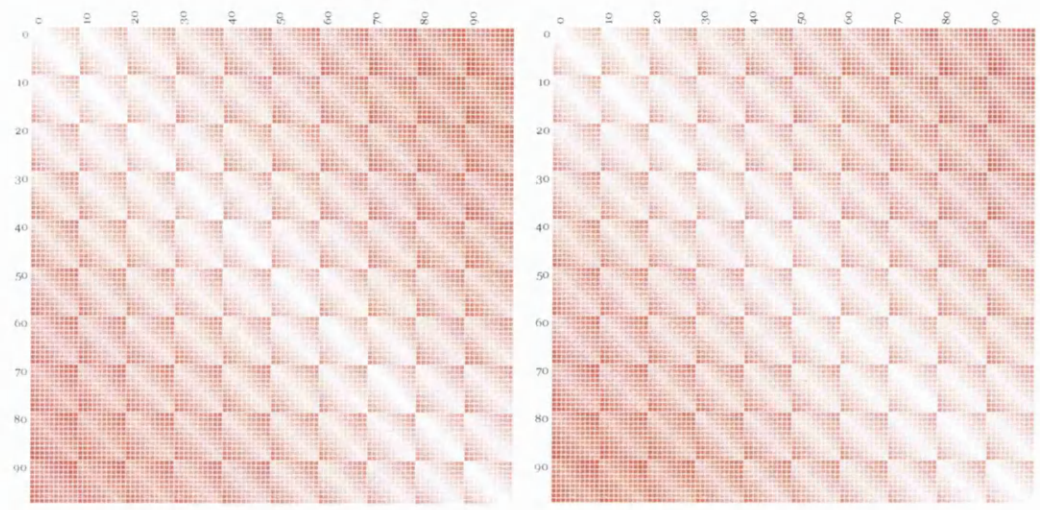
(a) cost as number of keystrokes (b) cost as predicted duration

Figure 5.8: The relative costs of changing between numbers on the D-pad interface when independent digit wrap is allowed.



(a) cost as number of keystrokes (b) cost as predicted duration

Figure 5.9: A visualisation of the all-pairs cost for the arithmetic variation of the up-down interface.



(a) cost as number of keystrokes (b) cost as predicted duration

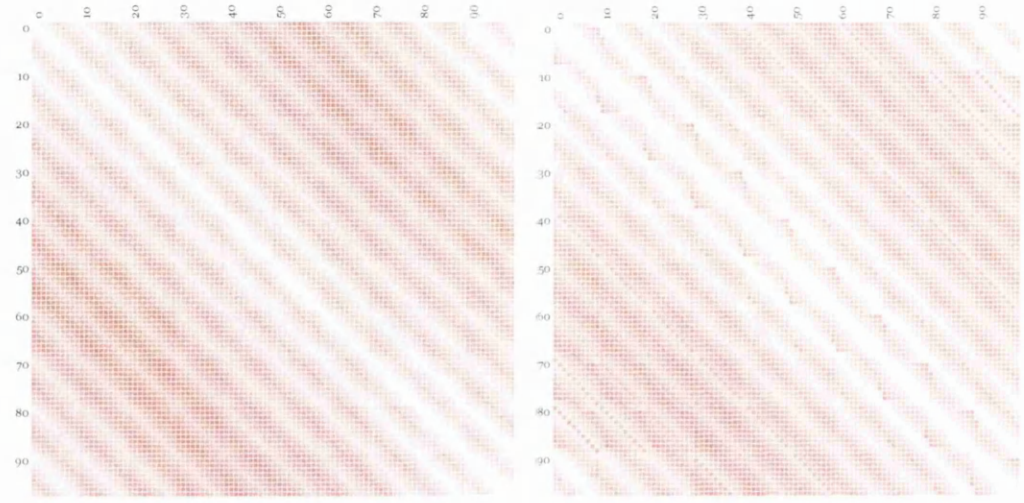
Figure 5.10: A visualisation of the all-pairs cost for the variation of up-down interface where digit wrap is blocked.

5.5.3 Up-down

When a minimum clicks strategy was used, the *arithmetic* variation of this interface required an average of 7.4 clicks to change zero to any number in a mean time of 4.7 seconds. Changes between any pairs on numbers were 5.8 clicks on average and lasting 4.2 seconds. When a minimum time strategy was used, average number of clicks to set a new number increased to 7.5 clicks but mean time required dropped to 3.5 seconds. Although overall number of clicks between numbers remained the same, the mean time required dropped to 3.1 seconds.

The click optimal strategy for the *block digit wrap variation* of this interface required 9 clicks to set any new numbers in a duration of 5.1 seconds while the overall changes between any pair of numbers in the range required 6.1 clicks in 4.4 seconds. Time optimal strategy required 9 keystrokes to set new numbers in 3.8 seconds. The overall distance between pairs was 6.6 clicks lasting 3.3 seconds.

The click optimal strategy for the *digit wrap variation* required 5 clicks to set new numbers as well as to change any number to any number, in 4 seconds. The time optimal strategy required 5 keystrokes to set new numbers lasting an average of 3.1 seconds. Overall, the mean number of keystrokes required to



(a) cost as number of keystrokes (b) cost as predicted duration

Figure 5.11: A visualisation of the all-pairs cost for the variation of the up-down interface with digit wrap around.

change between number pairs within the interface was 5.2 clicks in 3.1 seconds.

5.6 Performance of numbers in context

Chapter 4 presented analysis of logs from three medical devices to understand the nature of numbers used in infusion therapy and the way those numbers change. The analyses showed that depending on the device, numbers are often changed between non-zero values rather than being set from an initial value of zero. The performance of the d-pad, up-down and the numeric keypad interfaces are further evaluated with respect to time and in the context of those numbers that featured in the logs. This analysis would give some insight into the comparative performance of the different interfaces when used to enter the same numbers.

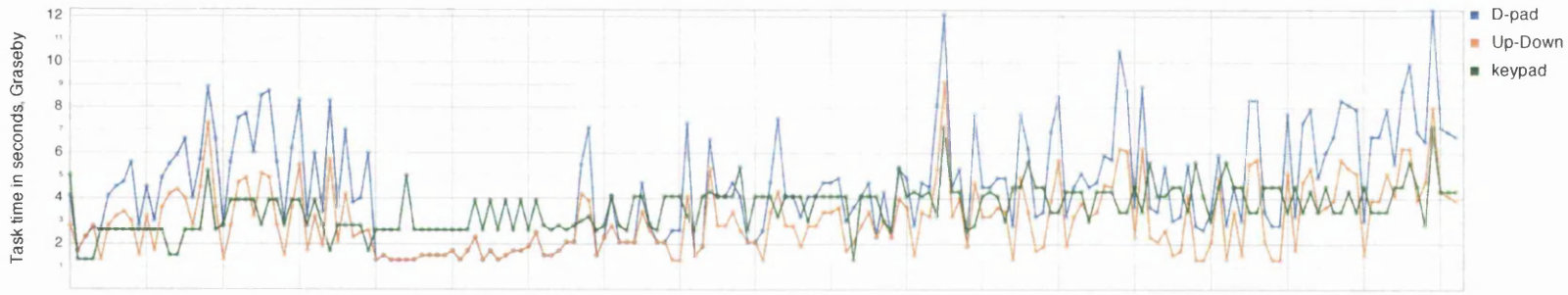
5.6.1 Method

For every different number change found in the logs, a corresponding task time was predicted for each one of the three interfaces (dpad, updown and keypad). The task times were predicted using a time optimal strategy in the A^{star} algorithm

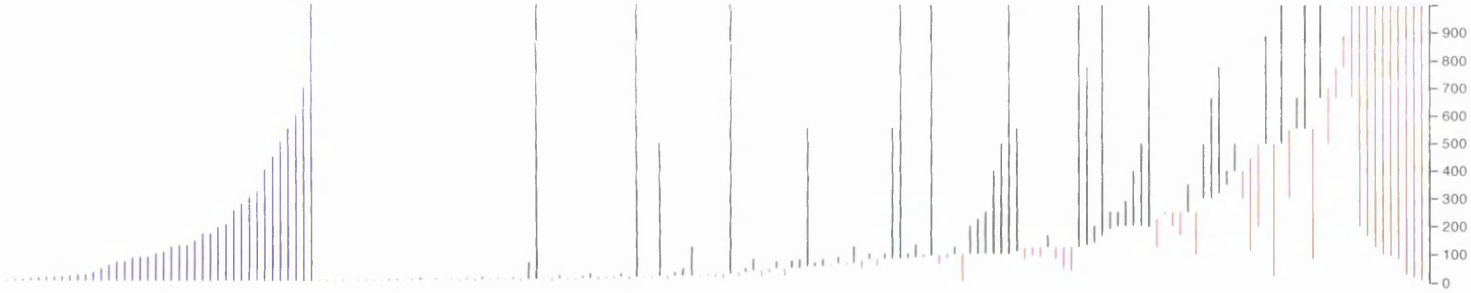
(a) Predicted interface task time

(b) All number changes

Figure 5.12: Predicted task times for the d-pad, up-down and numeric keypad for all the number changes that occurred on the logs obtained from the BBraun infusion pump. The blue lines in (b) show the changes that start from zero while the other lines show changes that start from non-zero values. (a) shows that both the d-pad and the up-down interface are faster than the keypad when changes are made from non-zero values. Particularly, the d-pad and up-down both have the same predicted performance for the majority of numeric changes that occurred on this device. This means most of the changes were incremental changes in the units part of the number.



(a) Predicted interface task time



(b) All number changes

Figure 5.13: Predicted task times for the d-pad, up-down and numeric keypad for all the number changes that occurred on the logs obtained from the Graseby infusion pump. The blue lines in (b) show the changes that start from zero while the other lines show changes that start from non-zero values. (a) shows that both the d-pad and the up-down interface are faster than the keypad when changes are made from non-zero values.

(a) Predicted interface task time

(b) All number changes

Figure 5.14: Predicted task times for the d-pad, up-down and numeric keypad for all the number changes that occurred on the logs obtained from the Asena infusion pump. The blue lines in (b) show the changes that start from zero while the other lines show changes that start from non-zero values. (a) shows that both the d-pad and the up-down interface are faster than the keypad when changes are made from non-zero values.

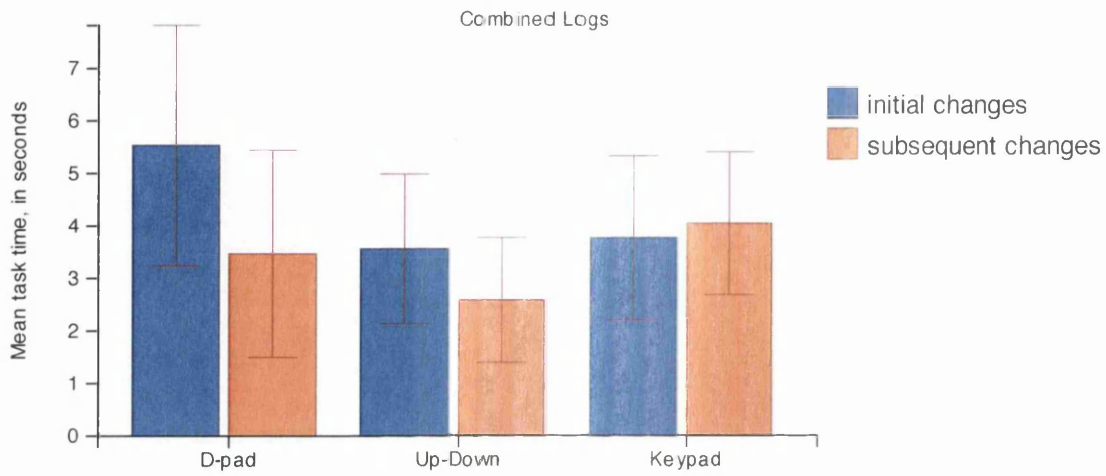


Figure 5.15: Mean predicted task time for the three interfaces grouped by the type of number being set. Error bars show standard deviation.

with functions defined earlier in this chapter. This study was a repeated measures design and the independent variable was the interface style with three levels: the different prediction functions run for each of the three interfaces.

5.6.2 Results

Initial number changes

For all numbers set from zero, a one-way repeated measures ANOVA with Greenhouse-Geisser correction found a statistically significant effect of interface style on predicted time $F(1.5, 394.3) = 295.78,\ p < 0.001$. Post-hoc analysis using multiple t-tests ($p < 0.001$) showed the *d-pad* was slower than the *keypad* $t(264) = 15.84,\ p < 0.001$ and the *d-pad* was slower than the *up-down* $t(264) = 29.25,\ p < 0.001$. There was no significant different between the *up-down* and the *keypad* $t(264) = -2.55,\ p = 0.011$.

Subsequent number changes

A one-way repeated measures ANOVA with Greenhouse-Geisser correction found a statistically significant effect of interface style on predicted time for subsequent number changes $F(1.27, 889.33) = 309.55,\ p < 0.001$. Post-hoc analysis using multiple t-tests showed a significant difference between all pairs ($p < 0.001$). The

d-pad was slower than the *up-down* $t(696) = 24.64$, $p < 0.001$, the *d-pad* was faster than the *keypad* $t(696) = -7.59$, $p < 0.001$ and the *up-down* was faster than the *keypad* $t(696) = -25.11$, $p < 0.001$.

Figure 5.15 shows the mean and standard deviation for all interfaces and for both initial and subsequent number changes. Figures 5.12, 5.13, 5.14 show details predicted task times for each number changes that featured in the BBraun, Graseby and Asena logs.

5.7 Discussion

The results show that for all the different number distributions from the different devices, the up-down interface has the mean predicted fastest entry time when making changes to a number and the task time is comparable to the *keypad* when setting new numbers. These results could be due to the nature of numbers used in infusion therapy as well as the nature of number changes made during therapies. These changes might be further informed by the therapies the devices are used to manage.

Other factors that might be responsible for this result are the generic KLM values used in the prediction of time for the different interfaces. This issue will be revisited later in Chapter 7 where the inter key durations observed during the experiments are contrasted with the generic KLM values used in the prediction.

The numeric keypad is probably the most popular number entry interface. It is the interface that users are most familiar with. Consequently, in practice, it is probably the fastest interface. However, based on this analysis, in the context of medical devices, and the types of numbers entered into infusion pumps, independent digit interfaces are sometimes faster. This finding means that new designs of independent digit interfaces can offer faster number entry for the medical context.

5.8 Summary

Using a mathematical approach, this chapter has provided generic functions that model the cost of changing between any two numbers on the numeric keypad

and six variations of independent digit interfaces. It also introduced the novel concept of discovery strategies used in the exploration of tasks in different variation of number entry interfaces. These strategies were implemented as heuristic functions in the A^{star} search algorithm and have been explored based on optimal number of clicks and optimal time taken to complete task.

In some cases, such as for the numeric keypad and the variation of the independent digits interface that blocks digit wrap, the different strategies produce the same path through the graph. This has implications for consistency and ease of learning. In these cases, regardless of user intent (i.e., less clicks or less time), the optimal route for performing the task is the same. When there is a difference between the results of these two strategies, the user is faced with an interface that has ambiguous ways of achieving the same goal.

This chapter presented a two-part analysis. First, the performance of changing between all pairs of numbers within the range 0-99 was explored to provide a restricted, but generic view of the relative performance of different variations of the interfaces tested. This analysis showed that the numeric keypad is fastest when setting new numbers. This is followed by the *digit wrap, arithmetic* and *blocked digit wrap* variations of the up-down and d-pad interfaces. When making changes between numbers however, the up-down interface appears to perform better than the numeric keypad. This highlights the efficiency in easily making changes to a number on this interface.

A second analysis, explored relative performance of the interfaces based on the numbers entered in three different types of infusion pumps. Unlike the first analysis, this one did not include number pairs which are unlikely to occur in the healthcare setting. The result of this showed that the up-down interface had the quickest mean speed of entry when making changes between numbers and performs the same as the numeric keypad when entering new numbers.

Chapter 6

Interface style and error detection

The purpose of any number entry interface is to accurately select or set a numeric value. Chapter 3 introduced a four part classification for number entry interfaces. Two of these are based on specifying digits that make up a number and the other two are based on choosing a number from a set of options. This chapter evaluates number entry interfaces with respect to their effect on error detection.

Number entry is perceived to be a very simple and mundane task — yet numerical drug dosing errors account for a significant portion of adverse drug events in hospitals, particularly in paediatrics [Jan10, Vic03]. Number entry errors can be as a result of a combination of user errors and poor interaction design. Hardware defects such as key bounces on keypads have been reported as a source of error in medical device programming [ISM06, Vai06]. A key bounce occurs when physically pressing a key once causes a repeat of the same key; this is different from a double keying error where a user accidentally presses the same key twice [Vai06].

In many user interfaces, number entry is implicit; for example, adjusting sound levels by rotating an unmarked dial, or moving a scroll bar adjusts a hidden number but the user copes because of direct feedback (direct manipulation).

Unfortunately, errors are inevitable when using interactive systems and these can be in the form of mistakes, slips or lapses [Rea90]. Sometimes, errors are detected by users and corrected. When errors go undetected, the consequences can be very serious. In a safety critical and dependable system, it is important that users realise when they commit errors and correct the errors. The differences in the design of number entry user interfaces place different demands on users in terms of what part of the user interface they focus most of their attention on and as a result whether they notice that an error has occurred and correct the error.

This chapter reports an experiment that investigates the effect of interface design on number entry user error detection. The findings show that the incremental interface produces more accurate inputs than the serial interface. It also presents a classification of error types that have implications for number entry interface designers, particularly in safety critical domains where accuracy needs to approach 100%.

6.1 Experiment

Because interaction on the incremental interface requires users to monitor how the value on the display changes based on what key the user is interacting with, it was hypothesized that users will more likely detect and correct errors when using an incremental interface as opposed to when using a serial interface. It was also anticipated that users will pay more visual attention to the display than to the input when using the incremental interface and pay more visual attention to the input than to the display when using the serial interface. Finally, it was expected that there will be types of errors that are unique to each class of interface.

6.1.1 Design

The experiment was a within-subject repeated measures design. Each participant used both number entry interfaces. The number entry interface was the independent variable and it had two levels: the incremental and serial interfaces. The order in which the interfaces were tested was counterbalanced for all participants. The dependent variables were the number of uncorrected errors, number

of corrected errors, total eye fixation time on the input and display part of the interface and task completion times.

6.1.2 Participants

Twenty-two participants from the University College London psychology pool for participants were recruited for this experiment. There were 12 female and 10 male participants aged between 18 – 55 years. All the participants were regular users of computers. None of the participants was prone to repetitive strain injury and one participant was dyslexic.

6.1.3 Apparatus

A computer with an integrated Tobii eye-tracker was used to present the instructions and the number entry interfaces. Participants interacted with the computer using a mouse to click on "keys" on the interface. Based on the properties of numbers analysed from Asena pump logs in Chapter 4, 100 numbers were generated randomly for the experiment with the following constraints:

- all numbers were between 0 and 10
- all numbers had a decimal point
- all numbers had at least one significant digit after the decimal point
- all numbers were unique

The two number entry interfaces used in the experiment were implemented in HTML and Javascript. Whenever an interface button was clicked, a scaling transformation was applied to the button to provide additional visual feedback that the button was clicked.

Serial digit entry interface

The serial interface was based on the Graseby 500 infusion pump (see Figure 6.1). It allowed number entry using a full numeric keypad in the telephone style layout.

Figure 6.1: The serial interface used in the experiment.

This interface had a decimal point key and a cancel key for deleting the rightmost character on the display. This interface allowed a maximum of 5 characters in its display buffer which may include only one decimal point. If the user clicks the decimal point more than once in a trial, only the first decimal point is registered in the buffer and subsequently on the display. Any other decimal point entries are ignored until the entire buffer is cleared or the currently registered decimal point is deleted.

Incremental number entry interface

The incremental interface was based on the Asena GH Syringe pump (see Figure 6.2). It had four keys. Two of the keys increased the value displayed and the other two keys decreased the value. For each of the two sets of keys, one key caused a bigger change while the other caused a smaller change (usually a factor of 10 smaller). This interface allowed two modes of interaction. The user could press the keys or press and hold the keys. Pressing the keys changed the displayed value as specified above. Pressing and holding the keys changed the displayed value at a rate dependent on the duration the key was held down for. Typically, users were expected to press and hold for faster changes to the number. This interface always displayed a decimal point and all numbers were rendered to a precision of two decimal places.

A key bounce was triggered for the 84th, 88th, 92nd and 97th trial for both interfaces. Once a user corrects a key bounce error, the key bounce was no longer triggered for that trial. Mouse actions were logged to obtain accuracy and performance data on the number entry tasks. Both interfaces had an *Enter* key

to commit the number entry task.

6.1.4 Procedure

All participants were tested individually. Before starting each session of the experiment, the eye tracker was calibrated for the participant. The participant was then briefed about the stages and purpose of the experiment before starting.

The experiment itself was in two parts: one for each interface. Prior to each part of the experiment, the participant got a training session where they could enter 10 numbers and get familiar with the interface. When the participant was comfortable with how the interface worked, they were allowed to proceed to the experiment. The participants could perform the training session as many times as they wanted and they were encouraged to ask questions to clarify how the interface worked during the training session.

For the experiment, each participant was required to enter 100 numbers using both interfaces in the order defined by the experimenter. The participants were instructed to enter the numbers as quickly and as accurately as possible. An instruction on the right half screen showed what number the participant should enter. The participant had to click a 'Next' key to confirm their entry. Doing this triggered the display of the next instruction. The process of number entry and confirmation of entry was repeated until all 100 numbers had been entered using the first interface. The participant was allowed a break of up to 5 minutes before proceeding to the second half of the experiment. The interface was then switched and the participant went through the training session for that interface and proceeded to enter the same set of 100 numbers. At the end of the experiment, each participant was given a gift voucher in return for their time.

Figure 6.2: The incremental interface used in the experiment

6.1.5 Defining corrected errors

Corrected errors for each participant on the serial interface were calculated as the total number of times they pressed the 'Cancel' button. For the incremental interface, the corrected errors for each participant were calculated as the number of times the participant overshot or undershot the target number. In the incremental interface, overshooting the target number was sometimes intentional especially when entering numbers efficiently using a mixture of continuous and discrete actions. For instance, entering the value '5.9' efficiently means slightly overshooting the target number using the continuous (hold-down) interaction in order to reach '6', for instance, and refining the value with a down click (a discrete action) to obtain the target number. This type of intentional overshooting did not count as a corrected error. To distinguish between intentional and unintentional overshooting on the incremental interface, the number of overshoots in a task were first determined for all numbers used in the experiment. This was then compared to the number of oscillations demonstrated by a user for the given trial as recorded by the experiment. The difference between these two numbers was used as the number of corrections per trial.

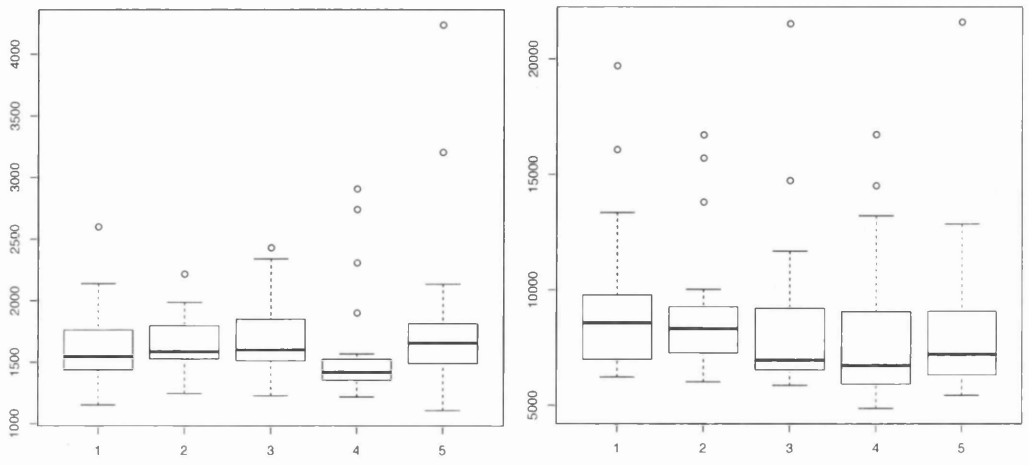
6.1.6 Results

Three participants were excluded from the analyses due to problems encountered while calibrating the eye-tracker.

To check for learning effects on the interfaces over the experiments, the trials

	Serial		Incremental	
	Mean	SD	Mean	SD
Block 1	1.65	0.35	9.35	3.54
Block 2	1.64	0.23	9.10	3.03
Block 3	1.71	0.33	8.72	3.83
Block 4	1.61	0.49	8.09	3.35
Block 5	1.85	0.72	8.53	3.82

Table 6.1: The mean entry time per trial block with corresponding standard deviations for the serial and incremental interfaces.

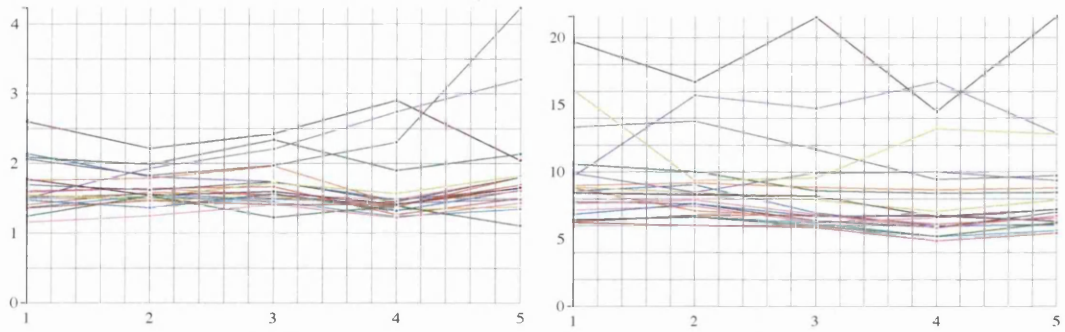


(a) Serial (b) Incremental

Figure 6.3: Box-plot of mean trial time by each participant for each block on the serial and incremental interfaces.

for each participant were split into 5 blocks; each block had twenty trials. A two-way repeated measures ANOVA showed a significant main effect of interface style on speed $F(1, 18) = 99.22,\ p < 0.001$, but no significant main effect of trial block on speed $F(2.74, 49.45) = 2.54,\ p = 0.072$. There was no significant interaction effect of interface style on trial blocks $F(3.02, 54.35) = 2.36,\ p = 0.081$. Table 6.1 shows the mean and standard deviations for the experiment blocks and Figure 6.3 shows the distribution of participant times for each block.

Further investigation was carried out to explore the trend of entry time in



(a) Serial (b) Incremental

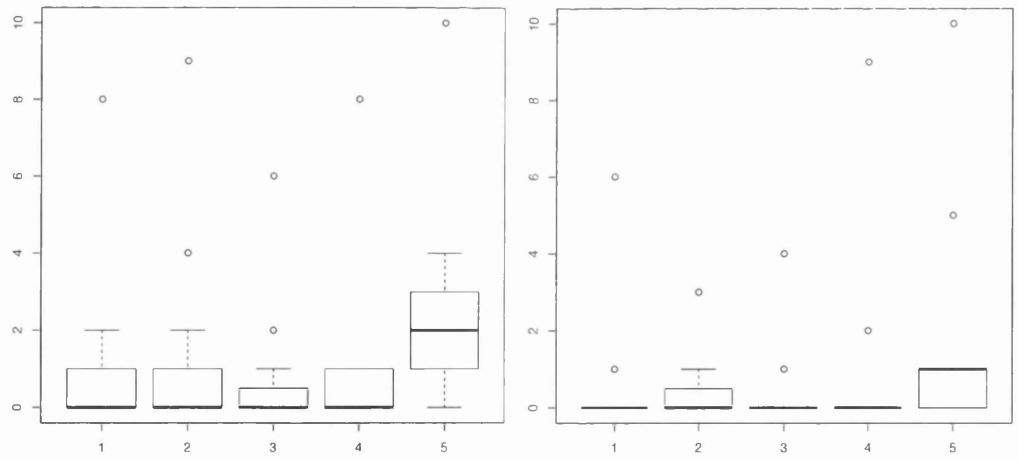
Figure 6.4: Mean trial time per block. Each line represents a participant.

more detail with respect to obtaining more context around the outlier values present in Figure 6.3. Detailed results are shown in Figure 6.4. No further investigation was carried out since no participant was consistently an outlier across all blocks on both interfaces.

Effect of key bounce on error

All key bounce errors were stimulated in the 5th block of the experiment trials. A Friedman test showed a significant main effect of trial block on uncorrected error for the serial interface $\chi^2(4) = 31.06$, $p < 0.001$. Post hoc analysis using Wilcoxon test with Bonferroni corrections showed a significant difference between errors made between blocks 5 and 1, $Z = -3.06$, $p = 0.002$, between blocks 5 and 3, $Z = -3.16$, $p = 0.002$ and between blocks 5 and 4, $Z = -3.35$, $p = 0.001$. Figure 6.5a shows detailed differences between blocks.

The trial blocks also had a significant main effect on uncorrected error for the incremental interface $\chi^2(4) = 23.47$, $p < 0.001$. Post hoc analysis using Wilcoxon test with Bonferroni corrections showed a significant difference between blocks 5 and 1, $Z = -3.13$, $p = 0.002$ and between blocks 5 and 3, $Z = -3.13$, $p = 0.002$. Figure 6.5b shows the distribution of participant errors per block and Table 6.2



(a) Serial (b) Incremental

Figure 6.5: The distribution of total uncorrected errors by each participant for each block on the serial and incremental interfaces.

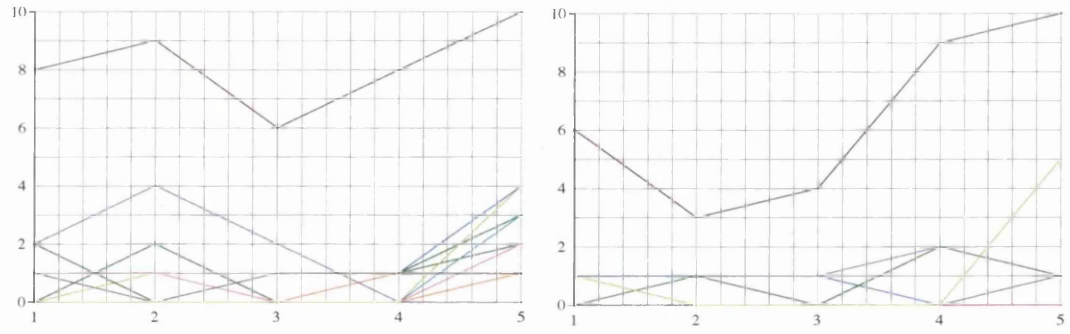
shows the mean and standard deviation for participant error per block.

	Serial		Incremental	
	Mean	SD	Mean	SD
Block 1	0.79	1.87	0.47	1.39
Block 2	1.05	2.20	0.37	0.76
Block 3	0.58	1.43	0.32	0.95
Block 4	0.74	1.82	0.68	2.11
Block 5	2.42	2.27	1.37	2.36

Table 6.2: The mean total errors per trial block with corresponding standard deviations for the serial and incremental interfaces.

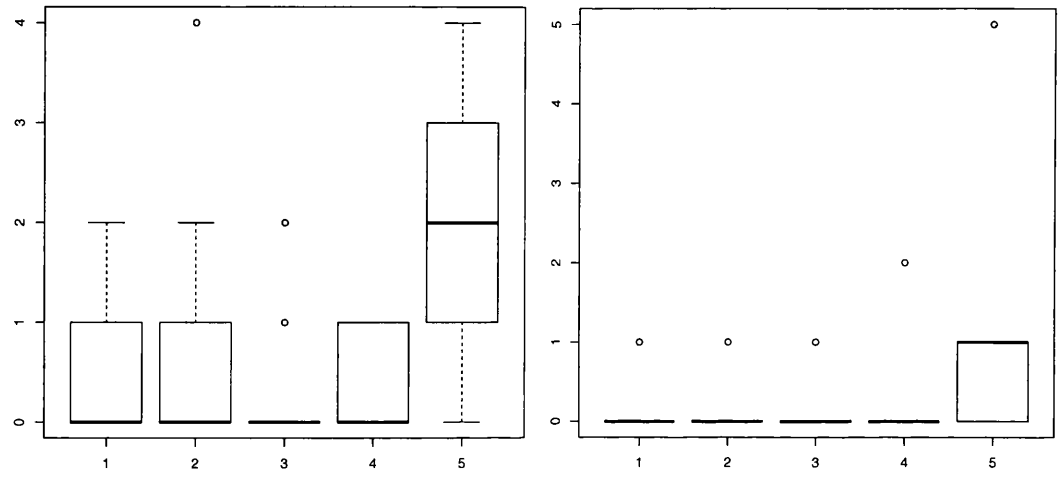
Further investigation was carried out to explore the consistency of the outliers in Figure 6.5. This was done to find out if there were participants that consistently had more errors than other participants for each block. Figure 6.6 shows that one participant (the same person on both interfaces) consistently committed more uncorrected errors than the others. This participant shall be referred to as P_5 from here on. The potential bias introduced by P_5 on the results obtained thus far was investigated by rerunning the analysis without trials from P_5.

A Friedman test showed a significant main effect of trial block on uncorrected error for the serial interface $\chi^2(4) = 28.18$, $p < 0.001$. Post hoc analysis using Wilcoxon test with Bonferroni correction showed a significant difference for errors made between blocks 5 and 1, $Z = -2.95$, $p = 0.003$, between blocks 5 and 3,



(a) Serial (b) Incremental

Figure 6.6: The number of uncorrected errors per participant for each block. A line represents a single participant.



(a) Serial (b) Incremental

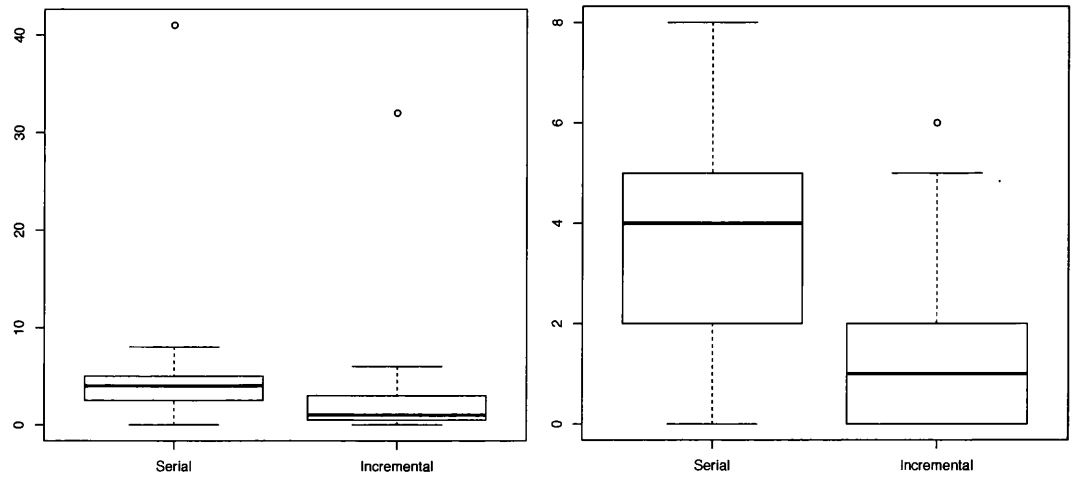
Figure 6.7: The distribution of the number of errors made by participants during each trial block on the serial and incremental interfaces. The data used in this distribution excludes data from P_5.

$Z = -3.03$, $p = 0.002$ and between blocks 5 and 4, $Z = -3.22$, $p = 0.001$, with block 5 having more error in each case.

Similarly, a Friedman test showed that the trial blocks also had a significant main effect on uncorrected error for the incremental interface $\chi^2(4) = 21.44$, $p < 0.001$. Post hoc analysis using Wilcoxon tests with Bonferroni correction showed a significant difference between errors made on blocks 5 and 1, $Z = -3.05$, $p = 0.002$ and on blocks 5 and 3, $Z = -3.05$, $p = 0.002$, with block 5 having more error in each case. Figure 6.7 shows the distribution of errors committed without data from P_5.

Uncorrected Errors

A total of 95 uncorrected errors were made in the experiment. P_5 was responsible for 53 of those errors. A Wilcoxon Signed Ranked test showed a significant main effect of interface style on total uncorrected error, $Z = -3.11, p = 0.002$. The total uncorrected errors on the incremental interface was significantly lower (mean=3.21, sd=7.2) than those on the serial interface (mean=5.58, sd=8.88). To explore the potential bias that P_5 had on the difference between the two



(a) Data including P_5 (b) Data excluding P_5

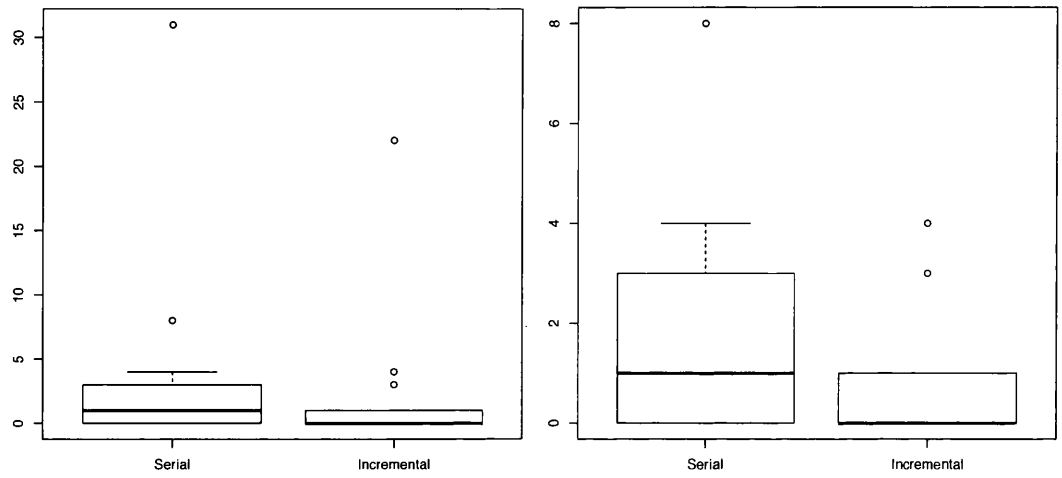
Figure 6.8: A comparison of the distribution of number of errors made on each interface.

interface styles, a Wilcoxon test was carried out with this participant excluded from the dataset. The test showed that the total uncorrected errors on the incremental interface (mean=1.61, sd=1.85) was significantly lower than those on the serial interface (mean=3.61, sd=2.38), $Z = -2.94$, $p = 0.002$. Figure 6.8 shows the median values for the different interfaces as well as the differences in the distributions of the datasets.

A Wilcoxon signed-rank test also showed a significant main effect of interface style on uncorrected error when the dataset included only the first four trial blocks, $Z = -2.06$, $p = 0.039$. When P_5 was excluded from this dataset, interface style did not have a significant main effect on uncorrected error, $Z = -1.8$, $p = 0.072$. Figure 6.9 shows the detailed distributions of total uncorrected errors committed by participants in the first four blocks of the experiment.

Corrected Errors

A Wilcoxon signed-rank test showed a significant main effect of interface style on corrected error. The number of corrected errors per participant on the incremental interface (mean = 74, sd = 17.31) was significantly greater than the number of corrected errors in the serial interface (mean = 7.1, sd = 9.6),



(a) Data including P_5 (b) Data excluding P_5

Figure 6.9: The distribution of total uncorrected errors by each participant for the first four blocks of trials. These represent the uncorrected errors committed out of the context of the stimulated key bounce error.

$Z = -3.73,\ p < 0.001.$

Severity of Errors

The severity of error, measured as the absolute difference between the intended value and transcribed value, was higher on the serial interface (mean=70.91, sd=166.94) than on the incremental interface (mean=0.93, sd=1.41).

Visual Attention

For both interfaces, a paired t-test found a significant main effect of area of interest (on the interface) on fixation duration. The total visual fixation duration on the input of the serial interface (mean = 271.16s, sd = 80.01), was significantly greater than the total fixation duration on the display of the device (mean = 26.28s, sd = 19.31), $t(17) = 13.35, p < 0.001$. Conversely, the total fixation duration on the input of the incremental interface (mean = 185.82s, sd = 87.78) was significantly lower than the fixation duration on the display (mean = 553.47s, sd = 276.25), $t(17) = 7.34, p < 0.001$.

Number of glances at instruction

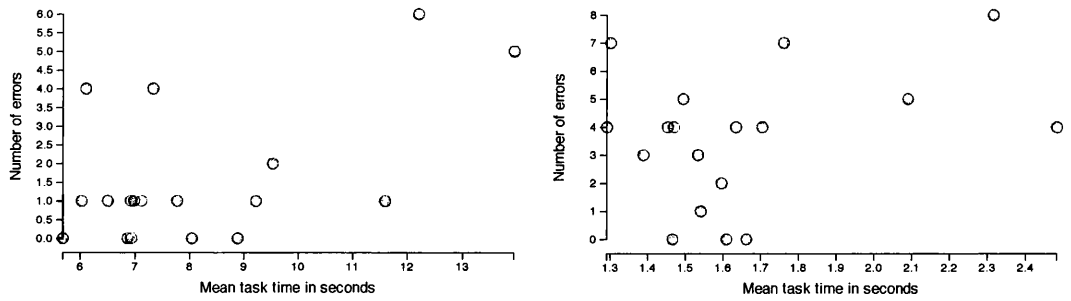
Analysis of fixation data from the eye-tracker using a paired t-test also showed that participants made significantly more glances at the instruction when using the incremental interface (mean=2.70, sd=1.13) than when using the serial interface (mean=1.85, sd=0.82), $t(17) = 3.04$, $p = 0.007$.

Speed of entry

A paired t-test showed a significant main effect of interface style on entry speed. The incremental interface (mean = 8.8s, sd = 3.31) was significantly slower than the serial interface (mean = 1.69s, sd = 0.36), $t(18) = 9.96$, $p < 0.001$.

Speed accuracy trade off

For the incremental interface, it was found that the mean speed of entry and the number of errors per participant were strongly correlated (*Pearson's* $r(16) = 0.55$, $p = 0.018$). For the serial interface, the correlation between speed and error was not statistically significant (*Pearson's* $r(16) = 0.29$, $p = 0.232$). Figure 6.10 shows a scatter plot of speed and errors for both interfaces.



(a) Incremental Interface (b) Serial Interface

Figure 6.10: Scatter plots of the mean speed and number of errors for the incremental and serial interfaces.

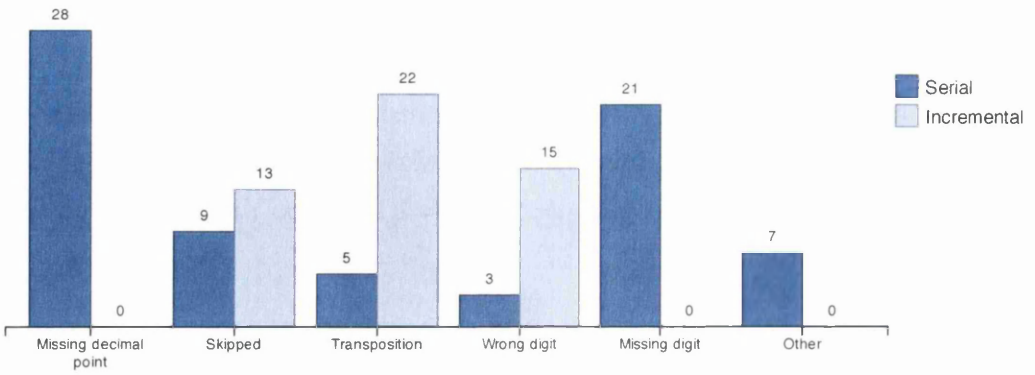


Figure 6.11: The distribution of the error types that occurred on the serial and incremental interface in the course of the experiment.

6.1.7 Error Types

Keystroke logs from all participants were analysed for this section. Below, a selection of the uncorrected error types that occurred in during the experiment is reported. Based on a different experiment, Wiseman et al. have developed a taxonomy of number entry errors [Wis11] and have independently reported and classified these errors. As well as reporting error types, the prevalence of certain error types between the two number entry interface styles is reported. The frequency of each error type is shown in Figure 6.11. For each error type, the severity of the error is quantified by reporting the mean and standard deviation of the difference between the intended number and the transcribed number.

Missing Decimal Point Errors

This error occurs when a decimal point is absent from the transcribed number but is present in the instruction. There were 28 instances of this error on the serial interface and none on the incremental interface. On average, this error changed the intended number by 260.77 (sd=240.85).

Transposition Errors

Transposition errors occur when the user swaps two adjacent digits in a number. For instance, instead of entering 5.84, a user might enter 5.48. The majority of these were committed on the incremental interface with 22 instances and only 5

instances on the serial interface. The dyslexic participant committed no transposition errors. Most of the transposition errors occurred after the decimal point. In our data, a special case of the transposition error occurred when the decimal part of the transcribed number was exactly 10 times more or less than the decimal part of the intended number. For example, instead of entering 7.4, a participant entered $\boxed{7.04}$. Although one participant (P_5) committed 17 of these errors, the potential causes make it a concern for further investigation.

It is possible that the display of the numbers on the incremental interface was responsible for this error: the display always shows two digits after the decimal point. For instance if the numeric value is 7.4, the display shows $\boxed{7.40}$. It may be confusing that the 40 after the decimal point is perceived to be greater than 4. It is important to note that this participant did not commit any transposition errors on the serial interface. It seemed that the incremental interface had an effect on their transcription of numbers specifically for numbers of the form d.0d where d is a numeric digit. In other words, the interface design might have affected their perception of numbers of a certain format. Transposition errors were more serious on the incremental interface. On average this error changed the intended number by 0.54 (sd=0.35) on the incremental interface compared to 0.31 (sd=0.18) on the serial interface.

Wrong Digit Errors

Wrong digit errors occur when one of the digits in the transcribed value is incorrect. This error was more common in the incremental interface. The most serious cases of the wrong digit error happened whenever the whole number part of the number is wrong. For instance a participant entered 4.87 instead of 5.87. Wrong digit errors were more serious on the serial interface but more frequent on the incremental interface with 15 occurrences on the incremental and only three cases on the serial interface. On average, this error changed the intended number by 0.81 (sd=1.27) on the serial interface compared to 0.28 (sd=0.40) on the incremental interface.

Missing Digit Error

This refers to instances of errors where one digit from the intended value is missing from the transcribed value. For instance a participant entered 0.3 instead of 0.43. On average, this error changed the intended number by 3.36 (sd=8.92). The incremental interface was free of this error.

6.2 Discussion

One participant (P_5) was responsible for 56% of all errors committed in the experiment. Excluding this participant from the analysis still resulted in a significant main effect of interface style on uncorrected error. Excluding this participant as well as constraining the analysis to the first four trial blocks, which had no key bounce error stimulation, resulted in no significant main effect of interface style on uncorrected error.

This shows that the key bounce introduced in the last block of the experiment had an effect on uncorrected error on both interfaces. The uncorrected errors were however not limited to the key bounce errors introduced in the last block of the experiment. The uncorrected errors that occurred in the first four blocks for the incremental interface were less than those that occurred on the serial interface. This suggests that the experimental manipulation alone might not be responsible for the significant difference in uncorrected error between the two interfaces.

The relative accuracy of the incremental interface comes with a slower data entry speed, although analysis of the speed of entry and number of errors for this interface showed a significant linear relationship which is contrary to the classic speed accuracy trade-off typical of many target acquisition tasks [Zha04a]. A potential reason for the linear relationship between speed and error rate observed on the incremental interface could be the increase in the likelihood that the longer time spent scrolling through numbers introduces more opportunities for error to occur.

The higher level of visual attention paid to the display of the incremental interface is another possible reason for its higher accuracy since placing visual

attention on the display gives the user a better chance or detecting and correcting any errors.

A third reason could be that participants expect to make errors using this interface. Indeed, the results show a significantly higher number of corrected errors on the incremental interface in comparison to the serial interface. Some participants had a number of tries overshooting and undershooting for the intended number before precisely setting the number. Some deliberately overshoot the intended value and correct the error in a few clicks because that is the optimal way to enter the intended number.

This oscillatory behaviour could also be explained as artifacts of the *gain* and *time-delay* parameters of the incremental interface. The design of this interface encourages the user to adopt the interface widgets that give the system a high *gain* value (i.e., holding down the double chevron buttons to cause larger changes to the number). The user thus appears to use a strategy with a high likelihood of missing their targets, while all they might have been trying to achieve was to get closer to the target number as quickly as possible.

For the incremental interface, the visual attention placed on the input was significantly lower than that on the display. This supports the original supposition, as the interaction on the incremental interface requires the user to monitor how the value on the display changes based on what key the user is pressing. The input part of the incremental interface requires little visual attention and is only used to switch direction and precision of change. However, despite the high attention paid to the display of this interface, the mode of interaction introduced errors that were less likely on the serial interface (e.g., the wrong digit errors and the transposition errors).

The results also show that the visual attention placed on the input in the serial interface was significantly higher than the visual attention on the display. This could be because participants did not feel the need to verify their entry. It is possible that most participants trusted the visual feedback they got from the labels on the keys and felt little need for an extra mode of feedback by checking the display.

By design, the numbers specified on a serial interface require parsing to obtain a numeric value valid in the application space. As a result, serial interfaces are prone to syntax errors. Rather than alert users to errors, this parsing process often produces incorrect and unpredictable results whenever the user commits a syntax error [Thi10b, Thi10a].

Syntax errors are however impossible on an incremental interface since the application guides the user through a valid range of numeric values. It is also plausible that numbers are perceived as a string of characters when using a serial interface whereas using an incremental interface forces users to be aware of the numeric values and the relative order of numbers.

In a safety critical context like healthcare, the incremental interface is safer. It allows better error detection and the severity of errors is much lower than on the serial interface. The missing decimal point and the missing digit errors are the most serious errors and they were both more likely to occur on the serial interface. Overall, the results suggest that the errors on the incremental interface have a much lower deviation from the intended number.

The information access cost (IAC) for participant revisiting the instructions in the experiment was low as the instruction for each trial was presented on the same monitor directly alongside the interface. Analysis of the eye-tracking data showed that participants glanced at the instructions at an mean rate of about two times per trial for the serial interface and about three times for the incremental. In practice, information about parameters for programming an infusion pump could be presented at different locations which could encourage the users to adopt a perceptual motor strategy (attributed to a low IAC) or a memory intensive strategy (attributed to a high IAC) [Bac12]. These different strategies are likely to introduce different types of errors and the effects of these strategies were not explored in this experiment.

6.3 Conclusions

P_5 committed the most transposition errors. Identifying users in the real world who are like P_5 would be useful to better manage errors in practice. Identification

could take the form of routine number entry tests. Alerting these types of users to the existence of these types of errors might be useful during training and might be enough to reduce the occurrences of such error. Similarly identifying user interfaces that afford better error detection would inform better design in a safety critical context. Further research would explore these avenues.

There are significant differences in the error rates for the two experimental conditions of number entry: number entry interface styles do affect error rates and, by implication, medical outcomes. This effect is particularly crucial and significant when there are hardware defects such as bounced keys present on a device. The speed of the serial interface comes at a price: errors are more likely to go undetected due to significantly less visual attention on the interface and undetected errors like the missing decimal or missing digit are more likely to have serious outcomes typically producing numbers out by a large factor (10 or more) from the intended values. In a medical context, such errors can be fatal. The result suggests that it should be a priority to research number entry styles and their relation to error rates, behaviour and performance. There is a wide variety of number entry styles in medical devices (where errors cause adverse events), clearly with no or little empirical justification; we now see useful progress can be made to provide sound guidance for designers of safety critical number entry systems.

Chapter 7

Exploring user performance for number entry interfaces

In the context of evaluating number entry for interactive devices, running experiments on desktop computers have many advantages, one of which is the relative ease of software deployment. Input widgets like mice and keyboards found on most computers are however very different from the specialised widgets found on real world interactive medical devices such as infusion pumps. These, like many other medical devices, are often operated by buttons on a membrane keypad. One of the reasons why membrane keypads feature in the design of medical devices is the ease with which they can be cleaned. Buttons on membrane keypads are not physically separate, therefore, there are no ridges or gaps where dirt could be trapped.

The indirect user experience encountered while using a mouse to interact with an on screen simulation of an interactive device is different from the experience a user has when directly interacting with a physical device with buttons. In order to bridge this user experience gap, we designed and built a bespoke prototype device with the specific aim of testing multiple configurations of number entry interfaces.

Figure 7.1: The prototype device built to run number entry experiments.

7.1 The prototype unit

The prototype device constituted three main parts: the display, the front panel and the housing. The display in the unit was a USB touch enabled monitor. The monitor was a 7-inch iMo* display with a resolution of 800 by 480 pixels. A removable front panel served as the input interface to the prototype and the housing held the display and front panel together.

The front panel was powered by an Arduino board which was wired to the user interface input components on the unit. Figure 7.1b shows the rear of the front panel. There were two variations on the front panel. The first variation had a blank set of 4×4 membrane keypads. The keys of the membrane keypad could easily be marked by sliding in any desired cut-out labels as shown in Figure 7.1d. The second variation of the front panel had a dial. The dial, pictured in Figure 7.1e was a 24 step rotary encoder with tactile feedback on each step. It also had a select switch which could be activated by pushing on the dial. Both front panels had a set of generic buttons laid out in a 3×2 grid. Part of the front panel could be covered up with a plastic card to hide it during an experiment.

As shown in Figure 7.1c, there was a clamp fitted on the rear of the device which enabled it to be securely fitted to a pole. The device had a dimension of 223mm x 282mm x 65mm.

The rest of this chapter presents the results of an empirical evaluation of 5 different number entry user interfaces using this prototype device. The aim was to explore the performance difference across these interfaces with the intent of providing quantitative summary of trade-offs involved in choosing to implement one of the styles of interface over another.

7.2 Related Work

The layout of the numeric keypad has been studied by many researchers. Early research by Deininger [R.L60] in the design of telephone keypads explored the performance differences of 16 layout configurations and the effect of keying be-

*http://www.displaylink.com/

haviour of users on the keying entry speed. This experiment found that the entry speed was dependent on the participant's strategy for reading the numbers. Participants who memorized the numbers before starting the keying sequence, performed significantly better than those who referred back to the number during entry.

Further experiments on the effect of keypad layout by Conrad and Hull [Con68] initially suggested that the telephone keypad layout with 1, 2, 3 at the top was more accurate than the calculator layout with 7, 8, 9 at the top. Marteniuk et. al[Mar96] later found that performance differences between different keypad layouts based on the two popular telephone and calculator layouts were as a result of the placement of the zero key, suggesting that the zero key be placed below the other keys.

Number entry interfaces can often be implemented in a variety of ways. For instance an independent digit entry interface such as the d-pad interface can be implemented in as many as 28 different ways. By running simulated trials of users making keying slips while entering numbers, Cauchi et al., [Cau12b] discovered that the differences in the implementation can have effects on the severity of error i.e., by how much an undetected error deviates from the intended number.

With a few exceptions, research in number entry has so far been based on the numeric keypad, usually testing the performance of different layouts. The serial interface offers very quick number entry and its performance scales well as the size of the number to be entered increases. Numbers used for tasks such as infusion therapy in hospitals are from a well defined range with rules governing the allowed precision of numbers above certain thresholds. For instance, precision of numbers used for rate settings in a critical care unit might be two decimal places for numbers below 10 and only one decimal place for numbers that are between 10 and 100.

Based on the classification presented in Chapter 3, 5 example number entry interfaces : 1 instance of serial digit entry (*numeric keypad*), 2 instances of independent digit entry (*up-down* and *d-pad*) and 2 instances of incremental entry (*chevrons* and *dial*), were implemented and evaluated. With the exception of the *dial* interface, which was based on a microwave oven, all the interfaces

presented below are found in real world medical devices.

7.3 Number entry interfaces

Since previous research has explored the performance effects of different layout configurations of the serial interface, only one instance of the serial interface is evaluated in this study.

7.3.1 Numeric Keypad

This interface allowed number entry using a 12-key numeric keypad in the telephone style layout (see Figure 7.2a). It had a decimal point and a cancel key. The decimal point key appends at most, one decimal point to the number on the display. The cancel key deletes the rightmost character on the display.

7.3.2 Chevrons

This interface utilised four buttons in a single row. The two buttons on the left (i.e., the upward facing chevron buttons) increased the value displayed, while the buttons on the right (i.e., the downward facing chevron buttons) decreased the displayed value. Within each pair of buttons, the double chevron buttons caused a change ten times greater than the single chevron buttons. This interface allowed two modes of interaction. The user could press the buttons or they could press and hold the buttons. Pressing the buttons changes the displayed value as specified above. Pressing and holding the buttons changes the displayed value at a rate dependent on the duration of hold. Users were expected to press and hold for faster changes to the number.

7.3.3 Up-down

This interface had eight buttons arranged in two rows and four columns. The top row buttons were used to increase the number and the bottom row buttons were used to reduce the number. Each column corresponds to a place value in the resulting number. For our set up, the rightmost column matched the hundredth

place value and was used to increase or decrease the value by 0.01. This interface worked using the *arithmetic* configuration described in section 3.3.2. This basically means the effect of decreasing a digit from 0 or increasing a digit from 9 is carried over to the digit to the left.

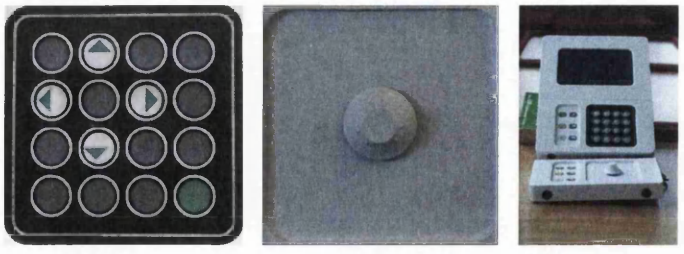
7.3.4 D-pad

This interface had four buttons arranged in a navigation style: up, down, left and right. The left and right buttons moved a cursor on the screen which selected a place value in the number. The up and down buttons increased or decreased the selected digit. Similar to the *up-down*, this interface worked using the *arithmetic* configuration.

7.3.5 Dial

This was a 24 step dial interface with unrestricted continuous rotations in both clockwise and anti-clockwise directions. Users entered numbers on this interface

(a) Numeric Keypad (b) Chevrons (c) Up-Down



(d) D-pad (e) Dial (f) Prototype

Figure 7.2: The prototype unit and the different configurations of interfaces used in our setup.

by turning the dial left or right to decrease or increase the number. Quicker turns on the dials caused bigger changes to the number.

7.4 Pre-study Analysis

Chapter 5 presented a method for evaluating the performance of interfaces that allow digit-level control of numbers. This method was used to explore the performance of the key based interfaces and estimated task completions times using the Keystroke-Level Model (KLM) for user performance [Car80]. Although KLM is a model for predicting error free expert performance, the predictions were used as the best-case performance achievable by users of these interfaces. Moreover, it was expected that the relative ranking produced by this analysis would be maintained in the results of the experiment.

7.4.1 Numbers used

The numbers used for the study were randomly selected from interaction logs analysed in Chapter 4. All 30 numbers used in the experiment had a decimal part and ranged from 0.26 to 83.3. A third of the numbers had a precision of 2 decimal places.

7.4.2 Pre-Study Method

Based on software simulations of the interfaces described in Section 7.3, the user interface model of each interface was exhaustively explored using the model discovery technique presented by Thimbleby and Gimblett [Thi07, Gim10]. The user interface model discovery process produces a graph whose nodes represent the states in an interactive system and edges represent the user actions necessary to transition between the states.

To reduce the number of states produced by the model discovery process, the numbers addressable by the interfaces were restricted to a range covered by those used in the experiment. Using the same method described in Chapter 5, for each button based interface, the optimal keying sequence for entering each

number used in the experiment were determined by searching for a shortest path from 0 to N, where N was the intended number. A JavaScript implementation of the A^{star} path finding algorithm was used, with cost functions that prioritised estimated time of execution over number of button clicks required to enter a number. The task completion times were estimated using the Keystroke-Level Model for expert performance [Car80]. A value of 1100ms was used for the time (**P**) taken to point to a button and a value of 200ms was used for the time (**K**) taken to click a button. The time (**M**) taken for mental preparation was not included in the prediction of task time. The analysis focuses on the execution time of the of each task. Consequently, the prediction presented in the next Section does not include the initiation time, (i.e., the time elapsed before the task is started) or the commit time, (i.e., the time taken to click the enter button to confirm the task).

Due to the nature of the method presented in Chapter 5, the time estimate for the chevrons interface utilised discrete key clicks rather than the continuous press and hold action. The time estimate for the dial interface was performed completely differently. Given that there are 24 steps in the rotary encoder used in the *dial*, to estimate the time T_{dial} required to enter a given number N on the dial interface, the following expression was used:

$$T_{dial} = \begin{cases} \frac{10t \times 100}{24} + \frac{(N-10) \times 10t}{24} & \text{if } N \geq 10 \\ \frac{N \times 100t}{24} & \text{otherwise} \end{cases}$$

Note that t is the time to perform one step rotation on the rotary encoder. The value for t was set as 200ms, the same as the value **P** taken to click a button.

7.4.3 Pre-Study Result

The analysis produced the estimates displayed in Table 7.1. The predictions show that the *up-down interface* is fastest with a slight performance edge over the *numeric keypad interface* and the *chevrons interface* is slowest. The next section describes a user study that was designed and run in order to validate these predictions.

Interfaces	Numeric keypad	Chevrons	Up-Down	D-pad	Dial
Time(ms)	4875	9545	4660	6954	7855

Table 7.1: An approximation of the task completion times for the different interfaces using the Keystroke-Level Model.

Based on error types reported in Chapter 6 and those reported by Wiseman et al. [Wis11], the following types of errors were expected to occur on the interfaces.

1. **Missing decimal** and **missing digit** errors, where users omit the decimal point and digit(s) in a number, were expected on the *numeric keypad*;

2. **Digit added** errors, specifically where an extra '0' is added to the left of the fractional part of a number, (e.g., entering 7.05 for 7.5 as reported in [Ola11]) were expected on the *chevrons* and *dial* interfaces.

Although there are no reported empirical evaluations that report the types of errors that the *up-down* and *d-pad* interfaces exhibit, **wrong digit** errors, where users mistype one or more digits in the presented number, (e.g., entering 1.95 for 1.85), were expected.

7.5 Method

7.5.1 Design

The experiment was a two-way, mixed design. The within subjects independent variable was the type of number entry interface, and it had five levels: the five interfaces tested. The between subject independent variable was the instruction given to the participant: one group was instructed to enter the numbers as quickly as possible (the *speed* group) and the second group was instructed to enter the numbers as accurately as possible (the *accurate* group). The order in which the interfaces were presented to the participants was randomized. The primary dependent variable was the speed of entry of correct numbers. Other dependent variables were the number of incorrect entries, the number of corrected errors.

7.5.2 Participants

There were 33 participants, 17 in the speed condition and 16 in the accurate condition. There were 22 females with 11 in the speed condition. Three participants were left handed. The participants ranged in age from 18–43 with a mean age of 23.5 years (SD=4.86). The participants were undergraduate and postgraduate students in Swansea University. Participants were randomly allocated to conditions.

Prior experience with interfaces

All participants were familiar with the *numeric keypad* and reported using it on interfaces such as calculators, cash machines and telephones. Five participants (15%) were familiar with the *chevrons* interface with experience using it in digital stop watches and alarm clocks, eight (24%) had prior experience with the *up-down* interface on medical devices and games, nine (27%) had prior experience with the *d-pad* interface on remote controls and game controllers and 19 (58%) had prior experience with the *dial* interface on microwave ovens and temperature controls.

7.5.3 Apparatus

The experiment was run on the unit described in section 7.1. With the two front panels of the prototype, it was possible to configure 5 types of number entry interfaces; 4 interfaces were configured using the membrane keypad and one using the dial. The different configurations are shown in Figure 7.2.

A pole was used to mount the prototype unit and the unit itself was connected to a laptop computer (a 15 inch macbook pro running OSX Lion). The laptop was used to display the instruction for the next trial. Instructions were displayed as numbers in the middle of the screen using a white font color on a black background and a font size of 20px. A total of 30 numbers were used in the experiment. Ten numbers were used in a practice session and 20 were used in the experiment. All the numbers used ranged between 0 - 100, all had a decimal part and they were selected from medical device logs of infusion pump settings.

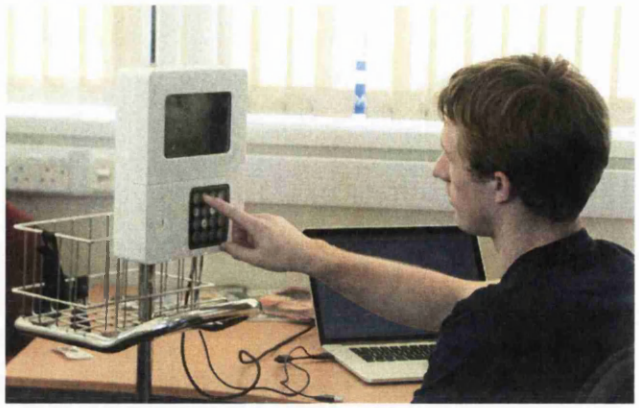


Figure 7.3: The setup for the experiment showing the prototype mounted on a pole and the laptop computer that displayed the instruction.

The software for the experiment was implemented in JavaScript and HTML. For these interfaces, each keystroke or user action causes a change in the numeric value being entered. As a result, for all the trials in the experiment, all keystrokes were logged with a timestamp and the corresponding numeric value after each keystroke. This allowed later analyses of the value change stream for every trial.

7.5.4 Procedure

Each study session lasted about 45 minutes and all participants were tested individually. Each participant was informed that the experiment involves entering numbers using 5 different number entry interfaces. Before the experiment started, a short pre-experiment questionnaire was used to collect demographic information about the age, gender, handedness and whether or not the participant was dyslexic.

The study itself was in 5 parts: one for each interface. Each part was divided into a practice session followed by an experiment session. Participants were randomly assigned to a speed or accuracy group. The speed group were instructed to enter the numbers in the instruction as quickly as possible and the accurate group were instructed to enter the numbers as accurately as possible. The order in which the users encountered the interface was randomised. All study instructions were displayed on the computer. The instruction was a number displayed in the

center of the computer screen. The next instruction was automatically displayed once the participant confirmed entry of the current trial. A message was displayed to signify the end of a session after a participant entered all the numbers required of that session.

Before starting each part of the experiment, the participants watched a video showing them how to use the interface they were about to test. They then had a training session were they could try out using the interface by entering 10 numbers. When they were confident with how the interface worked, they were allowed to proceed to the experiment. The participants were encouraged to repeat the training session if required.

The experiment session involved entering 20 numbers. The same 20 numbers were used for all the interfaces although the order in which the numbers were encountered was randomised. Participants successively entered the numbers displayed in the instruction.

After the experiment, participants were taken through a short post-experiment semi-structured interview to find out prior experience with the interfaces they experienced during the study and their relative preference for the interface styles. Participants were given a gift voucher in return for their time.

7.6 Analysis

The speed of entry recorded for each interface was separated into three constituent parts: the initiation time, the execution time and the commit time. The initiation time referred to the time elapsed between the display of the instruction and the participant's first key press. The execution time is the time elapsed between the participants first key press and the last key press involved in setting the required number. The commit time is the time elapsed between the last key press in setting the required number and the key press for confirming the task.

From the data collected, it was possible to analyse both corrected and uncorrected errors. Uncorrected errors were trials for which the user transcribed and confirmed a wrong number. The experiment elicited a total of 57 uncor-

rected errors, committed by 20 different participants. Only interface 4 was free of uncorrected errors.

7.6.1 Corrected Errors

Corrected errors were keying slips that the user recovered from before confirming the transcribed number. These might be seen as instances of unremarkable errors described by Furniss et al. [Fur11]. In other words, minor errors that are quickly corrected and recovered from. The experiment elicited a total of 833 corrected errors. Instances of corrected errors were spread amongst all participants ranging between 7 to 49 with a mean of 25.24 corrected error per participant. Corrected errors were determined by looking for patterns in the user's input sequence for every trial. Since keystrokes and values were logged for every trial, it was possible to analyse exactly how each participant entered the presented number.

Determining Corrected Errors

For the *numeric keypad*, detecting corrected errors from the key stroke logs was straight forward because there was a dedicated key (the 'C' button) for deleting the last character on the display. This signified an intention to correct an error.

For the *chevrons*, *up-down*, *d-pad* and *dial* interfaces, the value change stream from the experiment logs were analysed. The value change stream is a list of

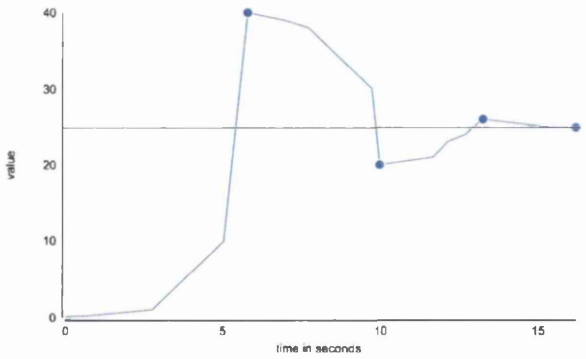


Figure 7.4: Graph showing the value change stream of a trial. The peaks and troughs are identified by the circle spots. In this case the target number was 24.9 - highlighted by the horizontal line in the middle of the graph. The trial visualised here contained three corrected errors.

timestamped values that are caused by user actions on the interface. When the value change stream for each trial is visualised, see for instance Figure 7.4, a graph with peaks and troughs representing the number of times a user overshoots and undershoots their target number during the keying sequence is obtained. By analysing the value changes over time and counting the turning points in the graph produced, i.e., the points when the user action on the interfaces changes from increments to decrements and vice versa, the number of corrected errors a user had while entering a number can be obtained. These turning points can be captured by matching the following regular expression on the input stream: $(u^+d^+)|(d^+u^+)$ where u captures any increment action and d captures any decrement action. In some cases, the optimal key sequence for entering a number contains exactly one turning point. For instance to enter 4.99 it would be quickest to increase the number to 5 and then reduce it to 4.99. These cases were accounted for and were not counted as corrected errors in the analysis.

The *d-pad* had a virtual cursor whose position could be changed by the user to select a digit on the screen. Cursor movements do not directly change the value of the number on the display. They signify an intention of the magnitude of change the user would like to effect on the displayed number. In addition to analysing the value change stream, these cursor movements were analysed for slips by checking the input stream for instances where at least one left key is immediately followed by at least one right key or vice versa. This was captured using the regular expression: $(l^+r^+)|(r^+l^+)$ where l and r are left and right key presses respectively.

7.7 Results

7.7.1 Learning effects

The improvement of speed of entry was investigated over all trials in the experiments. The trials for each participants were split into 4 blocks, each block consisting of 5 trials. A two-way repeated measures ANOVA showed a significant main effect of interface style on entry speed $F(2.06, 66.1) = 397.84, p < 0.001$. There was no significant main effect of trial block on entry speed $F(3, 96) =$

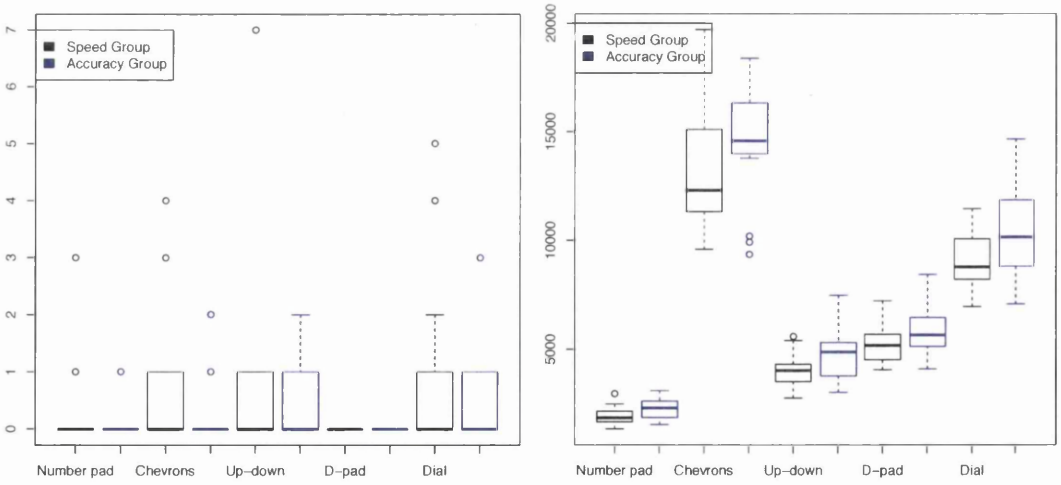
	Entry Accuracy				Entry Speed (in ms)			
	Speed Group		Accurate Group		Speed Group		Accurate Group	
	Mean	SD	Mean	SD	Mean	SD	Mean	SD
Numeric Keypad	0.29	0.77	0.06	0.25	1906	423	2266	466
Chevrons	0.65	1.17	0.19	0.54	13355	3122	14471	2691
Up-down	0.65	1.69	0.38	0.62	3990	745	4783	1210
D-pad	0	0	0	0	5231	908	5911	1213
Dial	0.94	1.92	0.44	0.81	9072	1211	10276	2024

Table 7.2: A summary of the mean, standard deviation for the speed and accuracy of entry between the groups.

$2.64, p = 0.54$. Similarly, there was no interaction effect between interface style and trial blocks $F(4.74, 151.86) = 1.003, p = 0.416$.

7.7.2 Effect of instruction

Table 7.2 shows means and standard deviations for each group across the five interfaces. There was a significant effect of group on entry speed $F(1, 31) = 4.23, p = 0.048$. Although more error was expected in the speed group, a Mann-Whitney test showed no significant effect of group on number of undetected error



(a) Entry Accuracy (b) Entry Speed

Figure 7.5: The distribution of the entry accuracy and entry speed by group.

on any of the interfaces. Figure 7.5 shows the distribution of participant accuracy and speed in the different groups. Since there was no significant difference in errors between the groups, both groups were combined for the rest of the analysis.

We next summarise the results of the speed of entry, the number of uncorrected errors and the number of corrected errors that occurred during the experiment on the different interfaces. For all the results below, except the user interface preference statistic, post-hoc tests were conducted using multiple t-tests with Bonferroni corrections in order to find out which interfaces differed significantly from the others. For the user interface preference, post-hoc test was conducted using multiple Wilcoxon Signed-Rank tests.

7.7.3 Speed of number entry

Initiation time

A one-way repeated measures ANOVA with Greenhouse-Geisser correction found a statistically significant effect of interface style on initiation time $F(3.31, 105.85) = 200.08$, $p < 0.0001$. Post-hoc analysis showed that the *dial* interface had significantly less initiation time than all other interfaces, and the *numeric keypad* had significantly less initiation time than the *chevrons, d-pad* and *up-down* interfaces. The *dial* interface had the shortest initiation time while the *chevron* interface had the longest initiation time. Table 7.3 shows the mean initiation time for all interfaces.

	Initiation Time		Execution Time		Commit Time		Total	
	Mean	SD	Mean	SD	Mean	SD	Mean	SD
Numeric Keypad	1286	339	2080	474	730	223	4096	821
Chevrons	1535	433	13896	2931	1005	291	16436	3231
Up-down	1469	392	4374	1061	970	323	6813	1531
D-pad	1463	354	5561	1105	1017	389	8040	1571
Dial	167	51	9655	1740	910	282	10732	1861

Table 7.3: A summary of the mean and standard deviation for the initiation, execution and commit times for the various interfaces. Time is reported in milliseconds.

Interfaces	Chevrons	Up-down	D-pad	Dial
Numeric Keypad	$p < 0.001$	$p < 0.001$	$p < 0.001$	$p < 0.001$
Chevrons	–	$p < 0.001$	$p < 0.001$	$p < 0.001$
Up-down	–	–	$p < 0.001$	$p < 0.001$
D-pad	–	–	–	$p < 0.001$

Table 7.4: A summary of all pairs of interfaces and the significance scores of the post-hoc analysis for execution time.

Execution time

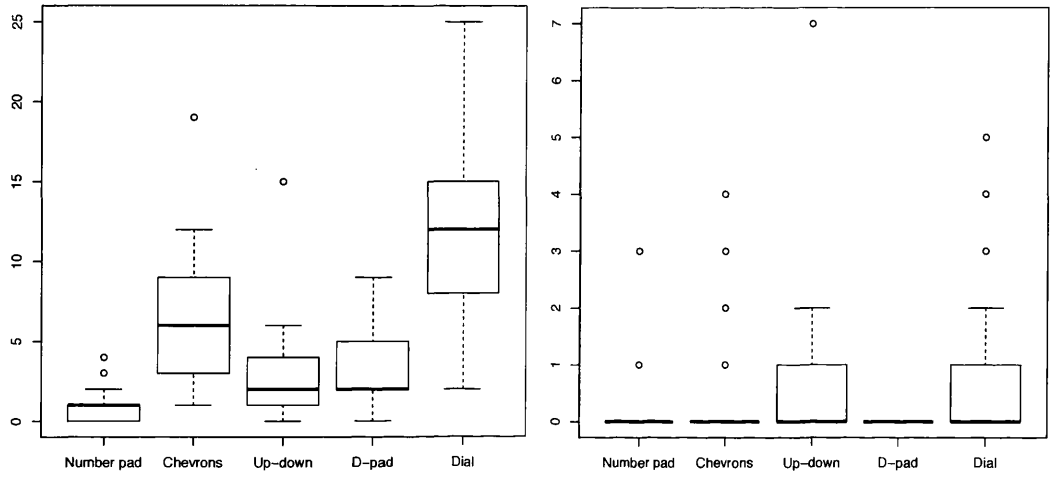
A one-way repeated measures ANOVA with Greenhouse-Geisser correction found a statistically significant effect of interface style on speed of entry $F(1.69, 54.02) = 425.5, p < 0.001$. Post-hoc test showed that the speed of entry of all the interfaces tested were significantly different for all pairs at the 0.001 level. The *numeric keypad* had the shortest execution time while the *chevrons* interface had the longest. Table 7.3 shows the mean execution times for all the interfaces.

Commit time

A one-way repeated measures ANOVA with Greenhouse-Geisser correction found a statistically significant effect of interface style on commit time $F(2.84, 90.83) = 24.35$, $p < 0.0001$. Post-hoc analysis showed that the *numeric keypad* had a significantly shorter commit time than all other interfaces and the *dial* had significantly shorter commit time than the *chevrons* interface. The results also show that the *d-pad* interface had the longest commit time. Table 7.3 shows the mean commit time for all the interfaces.

	Corrected Errors		Uncorrected Errors	
	Mean	SD	Mean	SD
Numeric Keypad	0.88	1.08	0.18	0.58
Chevrons	6.48	3.86	0.42	0.94
Up-down	2.73	2.74	0.52	1.28
D-pad	3.09	2.38	0	0
Dial	12.06	5.53	0.70	1.24

Table 7.5: A summary of the mean and standard deviation for the corrected and uncorrected errors for the various interfaces.



(a) Corrected Errors (b) Uncorrected Errors

Figure 7.6: Distribution of corrected and uncorrected errors on all interfaces.

7.7.4 Errors

We analysed both uncorrected errors and corrected errors. Uncorrected errors were trials for which the user transcribed and confirmed a wrong number whilst corrected errors were determined as described in Section 7.6.1.

Uncorrected errors

A Friedman test showed a significant effect of interface style on uncorrected errors $\chi^2(4) = 20.44, p < 0.001$. Post-hoc tests using Wilcoxon signed-rank test showed that the d-pad had less errors than the up-down Z=-2.97, p = 0.03 and the d-pad had significantly less errors than the dial, Z = -3.13, p = 0.02. The experiment elicited a total of 57 uncorrected errors, committed by 20 different participants. Only the *d-pad* interface was free of uncorrected errors. Table 7.5 shows the mean uncorrected errors on each interface.

Corrected errors

A Friedman test showed a significant effect of interface style on corrected errors $\chi^2(4) = 91.92, p < 0.001$. Post-hoc tests using Wilcoxon signed-rank test showed significant differences for all pairs of interfaces at a 0.01 level, with the exception

Interfaces	Chevrons	Up-down	D-pad	Dial
Numeric Keypad	$p < 0.001$	$p < 0.001$	$p < 0.001$	$p < 0.001$
Chevrons	–	$p < 0.001$	$p < 0.001$	$p < 0.001$
Up-down	–	–	$p = 0.34$	$p < 0.001$
Dial	–	–	–	$p < 0.001$

Table 7.6: A summary of all pairs of interfaces and the significance scores of the post-hoc analysis for corrected errors.

of the *up-down* and *d-pad* interfaces which did not differ significantly. Table 7.5 shows the mean corrected errors on each interface. The experiment elicited a total of 833 corrected errors. The *dial* interface had the highest number of corrected errors while the *numeric keypad* had the least number of corrected errors.

	Numeric Keypad	Chevrons	Up-down	D-pad	Dial
Mean Rank	4.81	1.69	3.50	2.44	2.56

Table 7.7: Mean ranks for interface preference.

7.7.5 User interface preference

At the end of the experiment, each user ranked the interfaces in order of preference. A score of 1 was assigned to the lowest preference while a score of 5 was assigned to the highest preference. There was a statistically significant difference in the preference rating for the user interfaces $\chi^2(4) = 73.8$, $p < 0.0001$. The *numeric keypad* was most preferred with a mean rank of 4.81 and the *chevrons* interface was the least preferred with a mean rank of 1.69. Post-hoc test showed no significant difference in preference ratings between the *up-down* and *dial*, *d-pad* and *dial* and *chevrons* and *five key*. Tables 7.7 and 7.8 show the mean ranks for all interfaces and the detailed significance scores between all pairs.

Interfaces	Chevrons	Up-down	D-pad	Dial
Numeric Keypad	$p < 0.0001$	$p < 0.0001$	$p < 0.0001$	$p < 0.0001$
Chevrons	–	$p < 0.0001$	$p = 0.008$	$p = 0.003$
Up-Down	–	–	$p < 0.0001$	$p = 0.013$
D-pad	–	–	–	$p = 0.82$

Table 7.8: Significance scores of the post-hoc analysis for preference ratings for all interface pairs.

7.8 Discussion

7.8.1 Relative preference of interfaces

Since all participants had prior experience using the *numeric keypad*, it was not surprising that it was rated highest amongst the interfaces tested. This preference rating is also reflected in the speed exhibited by the interface. It was however surprising that the *dial* was not rated significantly worse than *up-down* and *d-pad* interfaces considering the number of corrected errors that occurred on the *dial*. One possible reason for this could be the significantly shorter initiation time for the *dial*. In addition, the simplicity of the interface which is based on increasing and decreasing the displayed number means the user has to do little thinking while executing the task. This was articulated by one participant, who said:

> *"Dial was easier to turn the numbers. No need to move your hands from button to button."*

7.8.2 Types of errors

The types of errors made during the study spanned across seven classes of errors previously reported in separate studies by Wiseman et al. [Wis11] and Oladimeji et al. [Ola11]. A summary of all errors is provided in Table 7.9 and Figure 7.7 shows detailed distribution of error types by interface style.

The most common type of error was the *Digit Added* error. Thirteen different participants made this error on three different interfaces. This error also occurred

Error Type	Frequency	Interfaces	Example
Digit Added	32	chevrons, up-down and dial	4.05 for 4.5
Wrong Digit	7	chevrons, up-down and dial	60.5 for 62.5
Missing Decimal	3	numeric keypad	249 for 24.9
Out by ten	3	numeric keypad and up-down	1.11 for 11.1
Missing Digit	1	numeric keypad	6.5 for 62.5
Skipped	4	numeric keypad, chevrons, up-down and dial	'' for 62.5
No clear reason	9	chevrons, up-down and dial	56.7 for 3

Table 7.9: Frequency of errors made during the experiment.

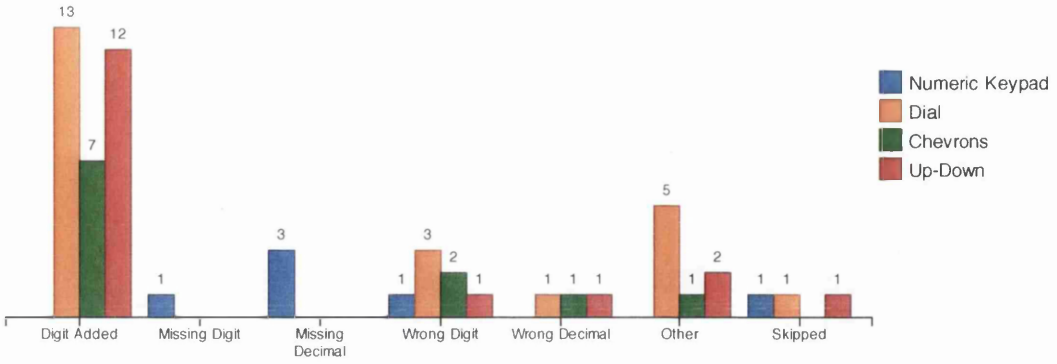


Figure 7.7: The types of undetected errosr and how frequently they occurred on each interface. Note that the D-pad interface is absent from this graph because it had no undetected errors.

in the experiment reported in Chapter 6 which investigated the effect of interface style on error detection. While this error is classified here as a member of the *Digit Added* error type, the nature of the error makes it different from what the error type suggests. Based on the numerals that compose the intended number and the transcribed number, the error type suggests that an extra digit has been added to the number. This extra digit in the case of errors in this experiment, is *always zero*. From a different point of view, however, this error appears to involve the inability to correctly understand the difference between the tenths and hundredths part of a number. It is possible that certain people mix up numbers matching the pattern. Indeed one participant transcribed 4.05 for 4.5 and in another trial transcribed 2.5 for 2.05. Over 50% of all unnoticed errors were of this form.

Despite making this error on the *chevrons*, *up-down* and *dial* interfaces, participants did not commit this error on the *numeric keypad*. This could be because number entry on the *numeric keypad* is a more direct transcription process of keying a sequence of digits that make up the intended number. Analysis of keystroke logs show that an instance of this error occurred on the *d-pad* interface although it was noticed and corrected by the participant.

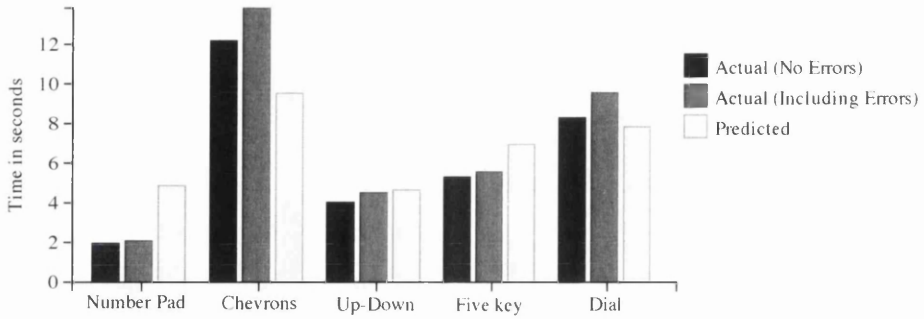


Figure 7.8: A comparison of the actual and the predicted performance for each interface

7.8.3 Difference in speed prediction and study results

It was expected that there would be absolute differences in the prediction of results and the actual study results since the participants that took part in the study were not expert users of all the interfaces. It was also expected that the participants' keying sequence for entering the numbers would not be the same as the time-optimal keying sequence used in the KLM analysis in section 7.4.2. For the numbers used in the experiment, the prediction expected the *up-down* interface to be marginally faster than the *numeric keypad*. Participants' familiarity with the *numeric keypad* however meant that their performance was superior on this interface in comparison to the other interfaces. For the *numeric keypad*, *up-down* and *d-pad*, participants actually outperformed the KLM model prediction. In the case of the *numeric keypad*, they performed the task in less than half the predicted time. This could be due to the mean inter-key duration of 554ms observed in the experiment for the numeric keypad in contrast to the standard estimates used in the prediction model (1300ms for pointing and selecting buttons or 200ms for selecting an already acquired button). The nature of the data logged during the experiment makes it impossible to separate these two parameters, since participants were not explicitly timed for target acquisition and selection.

The relative ranking in performance for the interfaces were preserved in the actual experimental data. In contrast to the *numeric keypad*, *up-down* and *d-pad* interface, the observed times for the *chevrons* and *dial* interfaces were higher than the predicted time. This difference in prediction could be due to the corrected

error rates on the *chevrons* (mean=6.48) and *dial* (mean=12.06) interfaces which were higher than the corrected error rates on the other interfaces. As a result, users spent a good portion of time correcting those errors which increased the task completion time and in some cases the frustration of users as evident in remarks during the post study interview:

> *"With the dial and the chevrons, you don't really know when it switches to higher changes. I tend to stop just before I reach the value I want so that I can increase in one step changes."*

> *"For chevrons and dial, really had to time it right and let it go at the right time otherwise could be annoying."*

The corrected errors alone do not account for the deviation between the prediction and the recorded performance of the *chevrons* and *dial* interface. Another contributing factor is the strategy users develop to reduce the error rates. In the case of the *chevrons* interface, users employed discrete click interactions rather than the faster press and hold interaction, and for the dial, users made slower but more careful turns. Figure 7.8 shows that the cost of correcting an error is larger on both these interfaces in comparison to the others.

7.8.4 Effects of interface style on number perception

The different styles of interfaces had an effect on how numbers were perceived by the users. When users entered numbers on the serial interface (the *numeric keypad*) or the independent digit interface (*up-down* or *d-pad*), they were more likely to think about the numbers as a sequence of digits without thinking much about the the numeric quantity of the number as a whole. Whereas, using an incremental interface such as the *chevrons* or *dial*, participants were more likely to concentrate more on the number as a whole. In the post experiment interview, one participant commented that:

Interface	Total Errors	Error Severity		
		Low Severity	Medium Severity	High Severity
Numeric Keypad	4	0	3	1
Chevrons	11	10	1	0
Up-down	16	13	3	0
D-pad	0	0	0	0
Dial	21	16	5	0

Table 7.10: The severity of undetected errors committed on each interface.

"For the number pad, up-down and the d-pad key, I did not think of the number as a whole, just entered them digit by digit but for the chevron and dial, I had to understand the number."

7.8.5 Severity of errors committed

The types of errors committed were closely related to the interface used to enter the number and consequently the severity of error, (i.e., the deviation of the intended number from the transcribed number or the ratio between the intended and the transcribed number). Theoretically, the *numeric keypad* and the *up-down* interfaces have the potential for producing the largest deviations from the intended number based on keying slips. This is due to the possibility of missing decimal points and missing digits on the *numeric keypad* and the possibility of wrong place value on the *up-down* interface. Three levels of error severity were defined based on the errors committed in our experiment. *Low* error severity referred to those errors where the ratio between the intended number and the transcribed number is at most 2, *medium* error severity refers to when the ratio is at most 10 and *high* error severity refers to when the ratio is greater than 10. Table 7.10 shows a summary of all errors committed and their severity.

7.8.6 Incremental interfaces and varying number precision

As is typical of setting up some infusion devices used in hospital critical care, the set of numbers used for the study required that numbers below 10 were precise to two decimal places while numbers from 10 and above were precise to one decimal

place. This factor meant that the display of incremental interfaces would only render numbers to the appropriate precision. As a result of this, button functions changed modes when the precision of numbers change on the display. For instance on the *chevrons* interface, when users change the value 9.99 to 10.0, the double chevron button changes meaning from 'increase by a tenth' to 'increase by a unit'. Similarly on the *dial* interface, one turn on the dial changes meaning from 'increase by a hundredth' to 'increase by a tenth'. The implementation of the *chevrons* interface was based on a medical device. It was also evident that some participant found the hold-down mode of the *chevrons* very difficult and challenging to predict. In this mode, the longer the buttons were held down, the larger the increments made to the number. This mode change was confusing for some users. A participant remarked during the interview that:

> *"For chevrons, the increments were very confusing. The same button did two jobs and the mode changes are confusing. Sudden changes were very confusing ...for example you could go from 30 - 60 in a very short time span and then going back restarts the counter and climbs up rapidly."*

Another one said:

> *"Chevrons, seem to jump quite a lot, took too long to get to intended number. Same problem with dial. It goes in sequential order*[†] *rather than control individual digits."*

As in Chapter 6, the incremental interfaces in this experiment exhibited oscillatory behaviour in the process of number entry. This is because the *gain* parameter of each button on the interface depends on duration of interaction and the current numeric value being manipulated. If the user glances away, or is distracted from the display of the interface for as little as 500ms, the meaning they have attributed to interaction with a button could change by a factor of

[†]referring to moving through the number line

ten causing a sudden jump on the number line. This is evident from participant comments in the interview.

The feature of varying precision described in this section is a requirement in infusion pumps used in critical care and intensive therapy units where low dose settings are common. It remains a design challenge to create an incremental interface that supports this form of varying precision in a way that is not confusing to the user.

7.8.7 Reuse of numbers and rehearsal effects

The numbers used in the experiment were repeated for all interfaces encountered by each participants. It is possible that the recurrence of the numbers could have increased the participants' familiarity with the numbers used in the trials which would cause quicker number entry time. This has been mitigated in the design of the experiment by randomising the order in which the numbers were presented to the participants as well as randomising the order in which the interfaces are presented. Analysis of learning effects also shows that trial blocks had no significant main effect on entry speed. It is also reasonable to expect that the outcome of rehearsal effects would manifest in the initiation times of the trials. This is because one would expect that reading a number more than once would improve the more times it is done. Analysis of the initiation times showed that interface style had an effect on initiation time. Based on the pairs of interfaces that differed significantly from the others (i.e., the *dial* and the *keypad*), this difference could be as a result of the ease of selecting manipulating the dial (this task was always the same for all numbers) and the users' familiarity with the numeric keypad interface.

7.9 Conclusions

Number entry is a very old art and it is a crucial aspect of the use of many interactive computer systems. The study presented in this chapter explored the performance of a variety of styles of interfaces and the data collected from the experiment conducted supports the following conclusions:

1. The 12-key numeric keypad is fastest , followed by the up-down, d-pad, dial and chevrons.

2. Errors vary per interface as does the type of error. Although the numeric keypad had the least occurrences of undetected errors, it had the most severe errors. This makes it a high-risk interface for safety critical contexts like healthcare. The d-pad interface had no undetected error.

3. Dials are perceived as easy to use by participants because of the easy interaction involved in turning them as opposed to successively pushing membrane keys on a keypad.

An error involving the wrong transcription of numbers with one non-zero digit in its decimal part was seen to occur on all but the numeric keypad interface. Future research would explore the potential causes of such error and investigate ways to prevent them from occurring in the course of use.

Chapter 8

Choosing an interface

The question of what type interface to choose arises in the design and development of many systems. The solution is typically dependent on the context of use and the factors and aspects of the design that are most highly rated or prioritised by the specification of the system. This chapter starts by identifying features for performing relative comparative analyses of number entry interface options. The purpose of these features are to facilitate easy perception of the trade-offs and risks involved in choosing certain interfaces instead of another. It completes the QOC design space analysis that was started in Chapter 3 by using the identified features as criteria which can be used to evaluate different options from the design space.

8.1 Evaluative features

8.1.1 Speed

The speed of entry simply refers to the time cost of entering a number using an interface. This has been evaluated analytically for different interfaces in Chapter 5 and empirically in two different user studies in Chapters 6 and 7. The factors that affect the speed of interfaces from the different classes are identified below. A summary of the results obtained from the evaluation is presented below

while highlighting the properties of the different interface which influence speed.

Serial digit entry interfaces

Interfaces from this group are based on the specification of the digits of a number. They allow digit entry in a predefined and restricted order, from the most significant to the least significant digit. They can be consequently very fast interfaces since the speed is dependent on the number of digits in the number and how quickly each digit can be specified.

Since the order of digit entry is restricted, making changes to digits in a number can be cumbersome depending on the error correction model used. Errors can be corrected using a clear digit key, which deletes one number at a time or they may be corrected using a *clear* key, which deletes all the numbers on the interface. Despite the speed, results from the analysis conducted in Chapter 5 shows that the numeric keypad, a serial digit entry interface, is theoretically not the fastest interface for entering numbers in infusion tasks. Although number entry is efficient on this interface style, error correction is inefficient.

Independent digit entry interfaces

These interfaces are also based on specifying the digits of a number although they allow digit specification in an arbitrary order. As a result, for certain types of numbers (e.g., numbers in infusion tasks), these styles of interfaces can be faster than the serial interface since the user does not have to specify all the digits in the number. This is possible because the digits usually start at an initial value of zero. So, the time cost to the user for entering a value of 1 is the same as that for entering a value of 100 since both cases require only one key press in the corresponding place value. This is different from serial interfaces where cost is proportional to the logarithm of the number.

Digit errors can be corrected easily as these interfaces allow independent control of digits in a number. Chapter 5 shows that this style of interface is theoretically best suited for entering infusion rates although it can be less efficient than a serial interface when entering large numbers with mainly non-zero digits.

Possible improvements

Changing the way digits are selected on this interface by implementing dynamic drop-down lists on touch screen devices could improve the speed of entry of this interface. Using dials for changing digits could also improve the digit selection time.

Incremental number entry interfaces

In comparison to other interface styles, these are usually slower because the interaction is based on making incremental changes to a number usually in the form of using predefined actions that are mapped to set values. This can also be seen as travelling up and down a number line which spans a range and precision defined by the application.

Possible improvements

Based on experiments run in Chapters 6 and 7, high frequency of overshooting and undershooting target numbers contributes to the slow speed of entry on incremental interfaces. Improving the speed of this interface thus, would require the user to have complete and active control on the amount of change caused to the number. This would reduce the error rate and can be achieved by the use of controllers like self-centering dials or spring-loaded dials with which users can control the amount of incremental changes on the number.

Speed can also be improved on this interface by having more widgets for making bigger value changes to the required number. This shows a trade off between speed and user interface foot print.

Direct number selection interfaces

The speed of these interfaces depends on how quickly a user might identify a number from a set of options as well as how quickly they can select that option. This is a search and select task. The search aspect of the entry would be affected by the number of items in the selectable set as well as the provision of any logical structure to the presentation of options, for example, sorting the options in increasing order. In addition, the size of the widgets used to represent the

items in the set affects the selection time. This selection time can be predicted using Fitts' Law [Fit54].

8.1.2 Error Rate

Error rate refers to how frequently errors occur while using the interface. Two types of errors have so far been analysed. On the one hand there are corrected errors which are noticed and rectified by the user. On the other hand there are uncorrected errors which are undetected by the users. These two components of error inform the perceived error rate of a user interface. Table 8.1 summarises the total number of corrected and uncorrected errors committed by participants on the five interfaces tested in Chapter 7.

	Numeric Keypad	Chevrons	Up-down	D-pad	Dial
Corrected Errors	29	214	90	102	398
Uncorrected Errors	6	14	17	0	23

Table 8.1: The total corrected and uncorrected errors recorded on each interface in Chapter 7.

The impact of the error rate of an interface should be assessed in relation to the error detection on the interface. The relationship between the corrected errors and the uncorrected errors on an interface suggests how well errors are detected and consequently rectified on that interface. Results from the experiment are used to estimate the percentage of all error instances that go undetected. In practice, the absolute number of errors would determine the real impact of the error rates of the different styles of interfaces and the overall number of errors would vary dependent on context. In general, reducing the impact of error rate of the different interfaces would be achieved by improving error detection on the interfaces.

Serial digit entry interfaces

The results from Chapter 7 show that the serial interface has the least number of corrected errors. The low number of keystrokes needed to enter numbers on this interface style provides less opportunity for committing errors. Also people's

familiarity with interfaces such as the numeric keypad reduces the chances for occurrences of error on this interface. However, the ratio of corrected errors to uncorrected errors on this interface showed that about 17% of all errors committed were not corrected. These errors were due to missing decimal points, wrong digits and missing digits.

Independent digit interfaces

Independent digit interfaces exhibit more corrected error than serial interfaces but less than incremental interfaces. The higher number of keystrokes required to set numbers containing non-zero digits means there are more opportunities for error when using this interface. Users can make key slips when selecting a place value in the number or they can make a slip when setting the digit itself. Since these interfaces typically require successively clicking the same button to select the required digit, issues of time-delay between button activation and interface feedback are likely to increase the chances of error.

On the *up-down* interface, experiment results showed that 16% of errors committed were not corrected. The majority of these errors were due to instances where an extra zero was added before a digit that follows a decimal point. The *d-pad* interface had more corrected errors than the *up-down*. The *d-pad* was however the only interface clear of uncorrected error.

Incremental interfaces

Incremental interfaces exhibited the most corrected error as many users of this interface oscillated about the target number a few times before selecting the number. Using well calibrated *gain* and *time-delay* parameters when designing this style of interface would improve the stability of this style of interface.

On the *dial* interface 5.5% of all errors were uncorrected and on the *chevrons* 6% of all errors were uncorrected. Most of these errors were instances of the *digit added* and *wrong digit* errors described in Chapter 7.

8.1.3 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. It is formally defined as the ratio of the intended value to the transcribed value in a number entry task [Thi10b, Ola11, Cau12a]. This ratio tells us, for instance, by how much the intended value is bigger or smaller than the transcribed value.

A particular class of errors is *order of magnitude* errors. This class describes errors where the ratio of the transcribed to the intended number is a factor of ten. To simplify the comparison process, these are assessed at a coarse level of *low*, *medium* and *high* severity signifying when an error is out by a factor r where $1 < r <= 2$ for low severity, $2 < r <= 10$ for medium severity and $10 < r$ for high severity. The severity of error presented below are those derivable from a single key-stroke action.

Serial digit entry interfaces

The numeric keypad is the most common serial digit entry interface. Errors on interfaces such as the numeric keypad are usually in the high severity level. Keystrokes on this interface have a direct mapping to the digits of the intended number. One keystroke is used to set each digit in the number and digits have to be entered sequentially. Where x is the current number, each subsequent digit keystroke d changes x by $9x + d$. This is a change of at least 900% over the current number. The actual level of error severity is dependent on the position within the number, at which the keying slip occurs. A keying error thus results in one of missing digit, missing decimal point, wrong digit, added digit or added decimal point.

A missing decimal on this style of interface results in the specification of a number which is an order of magnitude larger than the intended number. The specific order of magnitude is dependent on the precision (the number of decimal places) of the intended number. For instance, if the user wishes to enter a number 1.55 and they do so without the decimal point, then the transcribed number is one hundred-times more than the intended number. Similarly a missing or

added digit, depending on where it occurs in the transcribed number can cause big changes to a number.

Possible improvements

Reducing the severity of error on serial interfaces can be obtained by reducing the severity of errors that could occur as a result of an undetected single keystroke error or by increasing the likelihood that users of the interface would notice any key slip errors.

In the first case, the digit selection mechanism can be changed from direct digit selection to incremental digit selection. This change trades off speed of entry for accuracy. In addition to reducing the severity of error caused by a single key stroke, such a design separates digit selection from digit confirmation. This explicit confirmation could improve the user's ability to detect error. This design, however, deviates from the norm of implicit digit confirmation on selection present in numeric keypad serial interfaces.

Independent digit entry interfaces

These interfaces make changes to digits in a number in any order. This means one-step keying errors are limited to those possible by discrete one step increments in a chosen place value. For interfaces like the *up-down* or those that allow direct access to digits on a number, the user could erroneously set the wrong place value by setting the thousands place value instead of the hundreds place value. However the possibility of this one-step place value error is reduced in instances of this interface style such as the *d-pad*, where an explicit step is required to manipulate a cursor that selects the digit (or place value) the user wishes to edit.

Wrong digit errors could also occur as a result of a one step keying slip on this style of interface although the severity of this type of error is usually less than other digit based errors.

Incremental and direct entry interfaces

Incremental and direct number selection interfaces are not prone to digit based error since they allow number entry based on selection of a number from a set

of options. This interaction style means that the severity of key-slip induced number entry errors on these interfaces are usually much lower than for digit based interfaces.

8.1.4 Error Detection

This refers to the ability of users of an interface to notice key slip errors whenever they occur in the course of interaction. The likelihood of error detection is affected by the user's ability to accurately interpret the current state of a system based on the different modes of feedback the system is delivering to the user. The primary mode of feedback used in the design of number entry interfaces is visual. The detection of errors thus requires that the user pays attention to the part of the interface where feedback is provided.

Serial digit entry interfaces

Chapter 6 shows that errors are more likely to go undetected when using a serial style interface like the numeric keypad than when using incremental style interfaces. This is because the numeric keypad does not encourage the user to check the display after digit selection, probably because of the implicit feedback in searching for and selecting the required digit. Errors such as added digits or missing digits are consequently likely to go undetected.

Possible improvements

To improve error detection rates for the numeric keypad (e.g., on a touch screen interface), one can implement a reactive keypad interface which allows serial digit entry by dragging and dropping digits from the input part of the interface to the display. Digits are appended to the end of the display when they are dragged from the keypad over to the display area. Similarly, digits already on the display of the interface can be removed by dragging them out of the display. This design also has the advantage that syntax errors caused by keying multiple decimal points can be blocked by updating the keypad interface to remove the decimal point once one has been entered.

Independent digit interfaces

These interfaces encourage the user to pay attention to the display because the user interaction is based on selecting a digit and then making incremental changes to the digits. Once the user acquires a button, there is no visual feedback involved in looking at the buttons used to change the digits. The user needs to check the screen to ensure and confirm that the digit they are setting changes correctly. Despite this visual attention, the experiments reported in Chapters 6 and 7 show that interfaces of this style are not error free. Errors such as wrongly transcribing the decimal part of number might still go unnoticed.

Possible improvements

Error detection can be improved on this interface style by providing feedback that highlights the digit that is affected by user interaction. This impact can be further emphasized by highlighting all the digits succeeding the one that is currently being edited. The visual cue might give the user some perception of the magnitude of changes they are making.

Incremental interfaces

The interaction for these interfaces are based on number selection. As a result, the controls used to navigate the number line require very little visual attention. What is most important here is that the user visually monitors the changes to the number and is able to predict and make corresponding adjustments to the rate at which the number changes. For input widgets such as dials, the user does not even need any additional visual attention on the input widget after it has been acquired. Changes are performed with clockwise and anti-clockwise turning actions. Like the independent digit interfaces, wrong transcriptions of the decimal part of a number might also go unnoticed.

8.1.5 User interface footprint

This refers to the number of input widgets required to implement an interface. This feature is responsible for the amount of space that the interface covers on a device. This feature will be assessed based on the minimum number of widgets

needed to implement a style of interface as well as how the minimum required widget is affected by an increase in the range and precision of the interface. An interface with a *fixed* footprint would not be affected by a change in the range or precision of numbers in the host application. An interface that is affected by such changes would be referred to as having a *variable* footprint. This feature would be of interest to interface designers who are trying to minimise the cost of producing or maintaining a device or application.

Serial digit entry interfaces

The numeric keypad has a *fixed* user interface footprint. It requires a minimum of ten keys, 0 - 9, for operation. This variation would permit entry of whole numbers or entry of fractional numbers to a fixed precision (see section 3.3.1). A cancel key and a decimal point key might be added to the interface to allow error correction and entry of numbers to an arbitrary precision.

Possible improvements

A simple modification can be made to this interface to improve the interface footprint by implementing an incremental digit selection mechanism. This would require three or four keys. Two keys for setting the required digit using up and down arrows and one key for moving a cursor to the right ready to set another digit. An optional decimal point key may be provided to allow the specification of numbers to an arbitrary precision.

The user interface footprint can be further reduced by the use of four-way joystick, like those found on laptops, to change digits and to move a virtual cursor. This variation trades off the speed of entry for a smaller interface footprint.

These changes mean the digit selection on the interface would work just like the *d-pad*, although the order of digit specification would be strictly sequential (i.e., as the digits appear in the number).

Independent digit entry interfaces

When implemented without a cursor, this style of interface generally has a *variable* footprint. It requires a digit controller for each place value up to the max-

imum value accessible by the interface. These controllers range from a pair of up-down keys as found in the *up-down* interface described in section 3.3.2 to the full digit row of keys found in early calculating machines such as the Comptometer.

Adding a cursor to the display reduces the number of keys needed to implement the interface to 4 as featured in the *D-pad* interface which has a *fixed* footprint.

Possible improvements

The number of keys used in the *d-pad* can be reduced to two keys if the digits and the cursor movements on the interface wrap around. This improvement trades off speed of entry for a smaller user interface footprint.

The footprint can be further reduced by implementing the interface on a touch screen and controlling the digits with touch gestures such as swiping up or down to control digits and swiping left or right to control the cursor or simply activating the digits to be edited by touching them.

Incremental number entry interfaces

These interfaces have a *fixed* footprint because they can be implemented with a minimum of two buttons, one each for increasing and decreasing the number, or a dial. Control of the magnitude or speed of changes to the number can be provided on the interface with additional buttons although these need not change the footprint for instance, by using force sensitive buttons to allow the user control over the amount of change that occurs.

Direct number selection interfaces

These have a *variable* footprint that is dependent on the size of the set of numbers to choose from. Consequently, these are only feasible when entering numbers from a small sized set.

8.1.6 Range and Precision

The range refers to the minimum and maximum numeric value that can be specified using an interface. Range can be limited by external factors in the environment where the application will be used. For example, the *rate* value on an infusion pump would be limited to the minimum and maximum speed at which the hardware of the pump permits reliable and accurate delivery of medication. From a different point of view, range can be limited by the size of the number that can be written out on the display of the device or by the maximum number that can be held in the memory of the application. This limitation can be found on some calculators where, for instance, the display only permits entry of 8 digits.

The precision refers to the position of the rightmost significant digit in the numbers entered into the devices. Similarly to the *range* feature, *precision* can be limited by factors in the environment that affect the way the interface works. For example limits to the speed of infusion posed by the mechanical aspects of the device flow mechanism would restrict the interface to entry of numbers with a certain amount of precision. For example as seen in Chapter 4, the Asena pump did not permit entry of numbers bigger than 10 to a precision of 2 decimal places.

Range together with precision have an impact on the set of values that can be entered in an application. This consequently has an effect on the speed of entry of an interface and the footprint of the interface. These are reviewed for different classes of interfaces below.

Serial interface

The nature of serial number entry interfaces requires that the digits in the number are specified sequentially. As seen in Chapter 5, the speed of entry on serial interfaces is a logarithmic function on the length of the number to be entered. As a result, changes to the range and precision of the numbers have minimal effect on the interface. When an application only requires low precision values, there is no need for a decimal point key.

Independent digit interfaces

In the variation of this style that provides separate controls for each keys (e.g., the up-down), the number of keys on the interface is affected by the range and precision of the interface. This is because extra set of controls are needed for each place value in the maximum range of the numbers. Here, the interface footprint is negatively affected by an increase in the range or precision of the numbers that can be entered into the application.

For variations such as the *d-pad*, which provide a shared digit controller, but give the user the ability to select the active digit, changes in range and precision has no effect on interface footprint. The same mechanism for digit selection can be used to select amongst an arbitrary number of digits.

Possible improvements

The negative effect of arbitrary range and precision can be mitigated by creating a touch sensitive implementation of this interface as previously described in Section 8.1.5.

Incremental interfaces

Interfaces of this style typically provide ways for the user to navigate through the range of valid values in the application at a variable precision that can be selected by the user. An increase in range or precision does not negatively affect user interface footprint, although, as discussed in Section 8.1.1, speed of entry for large numbers on an incremental interface can be slow. If the designer intends to preserve the mean speed of entry, then they need to increase the footprint of the interface by providing more ways for the user to control the magnitude and speed at which changes to the number occur.

Direct number selection

Changes in range and precision have a negative impact on the footprint for interfaces of this style. Since these interfaces present options for users to select, an increase in the range increases the number of options which consequently increases the footprint of the interface.

8.2 Summary

The evaluations of criteria for the interfaces show several trade-offs that are highlighted below. Firstly, there is a trade-off between entry speed and severity of error on the interfaces. The faster an interface is, the higher the chances of severe errors occurring. This has been observed in the experiments reported in Chapters 6 and 7 with the faster interfaces producing the most severe errors although not necessarily the highest number of errors. Conversely, the slower an interface is, the more likely errors are to be detected and the less severe errors are likely to be. This can be seen, for instance, in the d-pad and up-down - two variations of the independent digit interface. The extra keystroke needed on the d-pad to move an on screen cursor reduces the speed of entry. This reduction in speed might be responsible for the lower severity of errors and better error detection.

There also exists a trade-off between user interface footprint and speed. Typically, a reduction in user interface footprint results in a reduction in speed of entry. Similarly an increase in speed results in an increase in user interface footprint. In certain interfaces (e.g., those offering direct number selection, or the up-down interface), an increase in the range and precision of numbers to be addressed by the application requires an increase in space to implement the interface.

This chapter has presented a set of criteria for use in comparing classes of number entry interfaces. Some of these are not independent of each other such that a change in one has an impact on the other. In the process of choosing an interface, a primary criterion should be selected at the beginning of the process based on factors that are considered to have the highest priorities in the context of the application.

For example, in the design of general calculators, users would be required to have the ability to perform calculations with numbers of arbitrary range and precision, and so might be primary criteria. This consequently limits the options in the design space to interfaces that are not negatively impacted by an arbitrary range and precision (e.g., the numeric keypad or the d-pad interface). To choose

between these two options, the designer might then specify that users must be able to perform calculations as quickly as possible - a criterion that positively favours the numeric keypad.

A second scenario might involve the design of a portable medical device which requires input of numbers having a small range and a low precision. A typical primary requirement for this application would be the user interface footprint. This limits the interface options to any incremental number selection interface or the d-pad. The designer can then further reduce the design options by prioritising one of speed or accuracy.

In any of the two scenarios, the designer may also apply QOC to further explore new variations on the interface styles and then perform usability evaluations to validate an option.

Chapter 9

Conclusion

Number entry is a very fundamental form of interaction task that requires the specification of numeric values to an application. It occurs obviously in simple everyday devices like calculators, timers, televisions and mobile phones and more critically in contexts like healthcare, finance and a different modes of transportation.

This research was motivated by data entry error in the use of interactive medical devices and the seemingly ad hoc manner with which number entry interfaces are currently designed and implemented. In an attempt to mitigate these problems and improve decision support for the design of devices like interactive medical devices, this Thesis has taken a structured approach to exploring different ways a number entry interface can be designed and explored the performance of different styles of interfaces both in terms of speed and accuracy, using an analytical model based technique as well as traditional laboratory studies. Below is a summary of research contributions.

9.1 Research contributions

Based on a historical review I conduct in Chapter 2 and a series of number entry interfaces currently encountered in interactive devices, I presented a classification of number entry interfaces in Chapter 3. This classification explores two

main dimensions and helps in structuring the examples of interfaces evaluated in Chapters 5, 6 and 7. Prior to this research, number entry interfaces have been generally referred to under a single group. This is probably as a consequence or effect of the popularity of the number keypad. This classification allows structural reasoning about the design options for number entry interfaces and it would encourage and inspire research for designing new interfaces.

In Chapter 5, I introduced a novel method of exploring task strategies of an interactive system using different heuristic functions in the A^{star} path finding algorithm and I presented four functions for use in deriving the cost for changing numbers on a variety of interfaces. I used these functions to calculate the number of clicks required to change between any two integers and I also used them as the basis for heuristic functions used to implement two strategies for performing number entry tasks. A first strategy was based on minimising the number of clicks required to complete the task while the second strategy was based on minimising the time required to complete a task. Task time was estimated using the keystroke level model of performance. The analysis showed that the sequence of action derived by these strategies differ for variations of some interfaces. This variation in task sequence has implications for consistency in interaction design.

In Chapter 6, I conducted an experiment to investigate the effects of interface style on error detection. This experiment tracked the eye fixations of participants while they entered numbers using the ubiquitous numeric keypad and the chevrons interface found on some medical infusion pumps. The results show that the type of interface used has an effect on the participant's detection of errors that occur on the interfaces. The effect on error detection is due to the change in user behaviour that occurs when using the interfaces. The amount of visual attention the users place on the display of the interface when using the chevrons interface is significantly more than the amount of attention they place on the input. Conversely, users rarely look at the display when using the numeric keypad. In addition, this experiment elicited error types that showed the effects of interface style on error type.

In Chapter 7, I built on the experiment in Chapter 6 and ran an experiment to evaluate the performance of five different styles of interfaces on a custom built

prototype device. This chapter showed that the speed of entering numbers are affected by the style of interfaces and the error types discovered in Chapter 6 were reproduced.

9.2 Generalising from this research

The motivation of this research was the seemingly avoidable occurrence of data entry errors in interactive medical devices. Consequently, the interfaces evaluated are taken from examples of medical devices that are currently in use in hospitals. Also, unlike other number entry experiments conducted to date, the type of numbers used the experiments reported are numbers within the range and precision found in typical infusion therapies. Limiting the range this way makes the tasks encountered in the experiments relevant to the medical context.

The relative performances of the user interfaces tested, with respect to speed of entry and the types of errors that occur on each type of interface, are transferable to other domains. Error rates may however not apply to other domains and indeed, there is no evidence that these error rates can be found in the medical context. The error rates observed in the laboratory studies were produced by generic users who had no specialist knowledge of the healthcare context. Error rates in practice would differ due to a different user skill level and the context in which the number entry task is performed.

9.3 Future work

This research has provided foundation for more work to be carried out in the design and evaluation of number entry interfaces. I highlight the following avenues where future work could follow.

- The design and evaluation of new number entry interfaces specifically those that exploit the use of touch screen hardware and multi-touch technology could be explored. Since users rarely look at the display when using the numeric keypad, a useful adaptation of this layout on the touchscreen in-

terface might require the user to drag the digits onto the display, thus encouraging a higher visual attention on the display of the interface.

- Exploring the effect of skill and context on performance of different interfaces. This would investigate whether skilled users such as nurses, who get equipment training significantly differ, with respect to speed and accuracy, from non-skilled users similar to those used for the experiments in this research.

- An aspect of number entry that has not been explored here is the different ways of representing numbers. These could be static or animated cues that add information about the type of numbers that a user is setting. This would be particularly useful when a task involves setting up more than one number and might reduce the chances of mixing up numbers.

Prior to the beginning of this work in April 2010, a review of literature showed a very restricted view of research in number entry which was then limited to the numeric keypad. This was perhaps due to the perception of number entry tasks as trivial or secondary. This research has shown that the design space of number entry interfaces is not limited to the numeric keypad and that the various options within the design space have different trade-offs and should be explored and empirically evaluated especially when designing interfaces for the safety critical context.

Bibliography

[Age07] Agency, N. P. S. *Safety in doses: medication safety incidents in the NHS.* In: *The fourth report from the Patient Safety observatory.* URL **http://www.nrls.npsa.nhs.uk/EasySiteWeb/getresource.axd?AssetID=61392**. (Last Accessed July, 2013). [cited at p.4]

[Age10] Agency, N. P. S. *Reducing harm from ommitted and delayed medicines in hospital.* URL **http://www.nrls.npsa.nhs.uk/alerts/?entryid45=66720**. (Last Accessed July, 2013). [cited at p.4]

[Bac12] Back, J., Cox, A., and Brumby, D. *Choosing to interleave: Human error and information access cost.* In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12. ACM, New York, NY, USA. ISBN 978-1-4503-1015-4, page 16511654. doi:10.1145/2207676.2208289. URL **http://doi.acm.org/10.1145/2207676.2208289**. [cited at p.116]

[Bat95] Bates, D. W., Cullen, D. J., Laird, N., Petersen, L. A., Small, S. D., Servi, D., Laffel, G., Sweitzer, B. J., Shea, B. F., Hallisey, R., Vander Vliet, M., Nemeskal, R., Leape, L. L., Group, A. P. S., Leape, L. L., Servi, D., Laird, N., Bates, D., Hojnowski-Diaz, P., Petersen, L. A., Petrycki, S., Vander Vliet, M., Cotugno, M., Patterson, H., Shea, B. F., Hickey, M., Kleefield, S., Cooper, J., Cullen, D. J., Kinneally, E., Nemeskal, R., Sweitzer, B. J., Small, S. D., Demonaco, H. J., Clapp, M. D., Hallisey, R., Gallivan, T., Ives, J., Porter, K., Thompson, B. T.,

Laffel, G., Hackman, J. R., and Edmondson, A. *Incidence of adverse drug events and potential adverse drug events.* JAMA: The Journal of the American Medical Association, **volume 274, no. 1**:(1995) pages 29 –34. [cited at p.3]

[Ben38] Benford, F. *The law of anomalous numbers.* Proceedings of the American Philosophical Society, **volume 78, no. 4**:(1938) pages 551–572. ISSN 0003-049X. doi:10.2307/984802. URL `http://www.jstor.org/stable/984802`. ArticleType: research-article / Full publication date: Mar. 31, 1938 / Copyright 1938 American Philosophical Society. [cited at p.56, 74]

[Bla06] Blatt, R., Christianson, M. K., Sutcliffe, K. M., and Rosenthal, M. M. *A sensemaking lens on reliability.* Journal of Organizational Behavior, **volume 27, no. 7**:(2006) pages 897–917. ISSN 0894-3796. doi:10.1002/job.392. URL `http://doi.wiley.com/10.1002/job.392`. [cited at p.28]

[But02] Butts, L. and Cockburn, A. *An evaluation of mobile phone text input methods.* Aust. Comput. Sci. Commun., **volume 24, no. 4**:(2002) pages 55–59. doi:http://doi.acm.org/10.1145/563997.563993. [cited at p.2, 18]

[Bux83] Buxton, W. *Lexical and pragmatic considerations of input structures.* SIGGRAPH Comput. Graph., **volume 17, no. 1**:(1983) pages 31–37. ISSN 0097-8930. doi:10.1145/988584.988586. URL `http://doi.acm.org/10.1145/988584.988586`. [cited at p.37]

[Car80] Card, S. K., Moran, T. P., and Newell, A. *The keystroke-level model for user performance time with interactive systems.* Commun. ACM, **volume 23, no. 7**:(1980) pages 396–410. doi:10.1145/358886.358895. URL `http://portal.acm.org/citation.cfm?id=358895`. [cited at p.71, 125, 126]

[Car83] Card, S. K., Newell, A., and Moran, T. P. *The Psychology of Human-Computer Interaction.* L. Erlbaum Associates Inc., Hillsdale, NJ, USA (**1983**). ISBN 0898592437. [cited at p.71]

[Car90] Card, S. K., Mackinlay, J. D., and Robertson, G. G. *The design space of input devices.* In: *Proceedings of the SIGCHI Conference on Human*

Factors in Computing Systems, CHI '90. ACM, New York, NY, USA. ISBN 0-201-50932-6, pages 117–124. [cited at p.37]

[Car91] Card, S. K., Mackinlay, J. D., and Robertson, G. G. *A morphological analysis of the design space of input devices.* ACM Trans. Inf. Syst., **volume 9, no. 2**:(1991) pages 99–122. ISSN 1046-8188. doi:10.1145/123078.128726. URL `http://doi.acm.org/10.1145/123078.128726`. [cited at p.37, 41]

[Cau12a] Cauchi, A. *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. ACM, New York, NY, USA. ISBN 978-1-4503-1168-7, pages 317–320. [cited at p.31, 46, 71, 152]

[Cau12b] Cauchi, A., Gimblett, A., Thimbleby, H., Curzon, P., and Masci, P. *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. Birmingham, UK, pages 29–38. [cited at p.31, 46, 47, 71, 122]

[Con68] Conrad, R. and Hull, A. J. *The preferred layout for numeral data-entry keysets.* Ergonomics, **volume 11, no. 2**:(1968) pages 165–173. ISSN 0014-0139. doi:10.1080/00140136808930953. URL `http://www.tandfonline.com/doi/abs/10.1080/00140136808930953`. [cited at p.27, 122]

[Cor09] Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. *Introduction To Algorithms.* MIT Press, third edition edition (**September 2009**). ISBN 9780262032933. [cited at p.77]

[Dan68] Dantzig, T. *Number: The Language of Science.* Plume Book (**January 1968**). ISBN 9780452288119. [cited at p.2, 12]

[Day98] Day, S., Fayers, P., and Harvey, D. *Double data entry: What value, what price?* Controlled Clinical Trials, **volume 19, no. 1**:(1998) pages 15–24. ISSN 0197-2456. [cited at p.32]

[Dei60a] Deininger, R. L. *Desirable push-button characteristics.* IRE Transactions on Human Factors in Electronics, **volume HFE-1, no. 1**:(1960) pages 24–30. ISSN 0099-4561. doi:10.1109/THFE2.1960.4503262. [cited at p.25]

[Dei60b] Deininger, R. L. *Human factors engineering studies of the design and use of pushbutton telephone sets.* Bell System Technical Journal, **volume 39**:(1960) pages 995–1012. [cited at p.25, 26]

[Dij59] Dijkstra, E. W. *A note on two problems in connexion with graphs.* Numerische Mathematik, **volume 1, no. 1**:(1959) pages 269–271. ISSN 0029-599X, 0945-3245. doi:10.1007/BF01386390. URL `http://www.mendeley.com/research/a-note-on-two-problems-in-connexion-with-graphs/`. [cited at p.77]

[Doh99] Doherty, G. and Massink, M. *Continuous interaction and human control.* In: *Proceedings of 18th European Conference on Human Decision Making and Manual Control.* Loughborough, pages 80–96. [cited at p.51]

[Doh12] Doherty, C. and Donnell, C. M. *Tenfold medication errors: 5 years experience at a university-affiliated pediatric hospital.* Pediatrics, **volume 129, no. 5**. ISSN 0031-4005, 1098-4275. doi:10.1542/peds.2011-2526. [cited at p.5, 30]

[Dun08] Dunlop, M. D. and Montgomery Masters, M. *Investigating five key predictive text entry with combined distance and keystroke modelling.* Personal Ubiquitous Comput., **volume 12, no. 8**:(2008) pages 589–598. ISSN 1617-4909. doi:10.1007/s00779-007-0179-7. URL `http://dx.doi.org/10.1007/s00779-007-0179-7`. [cited at p.19]

[Esk02] Eskew, J. A., Jacobi, J., Buss, W. F., Warhurst, H. M., and DeBord, C. L. *Using innovative technologies to set new safety standards for the infusion of intravenous medications.* Hospital Pharmacy, **volume 37, no. 11**:(2002) pages 1179–1189. [cited at p.33]

[FDA09] FDA. *Summary of medsun reports describing adverse events: Large volume infusion pumps and issues resulting in over-infusion.* In: *Med-*

Sun: Newsletter 33. URL http://www.accessdata.fda.gov/scripts/cdrh/cfdocs/medsun/news/newsletter.cfm?news=33. (Last accessed July, 2013). [cited at p.29]

[Fit54] Fitts, P. M. *The information capacity of the human motor system in controlling the amplitude of movement.* Journal of Experimental Psychology, **volume 47, no. 6**:(1954) pages 381–391. ISSN 0022-1015(Print). doi:10.1037/h0055392. [cited at p.19, 44, 73, 150]

[Fol80] Foley, J. D., Chan, P., and Wallace, V. L. *The Human Factors of Graphic Interaction: Tasks and Techniques.* Technical report (**December 1980**). URL http://www.dtic.mil/cgi-bin/GetTRDoc?Location=U2&doc=GetTRDoc.pdf&AD=ADA136605. [cited at p.22, 34]

[Fol84] Foley, J., Wallace, V., and Chan, P. *The human factors of computer graphics interaction techniques.* IEEE Computer Graphics and Applications, **volume 4, no. 11**:(1984) pages 13–48. ISSN 0272-1716. doi:10.1109/MCG.1984.6429355. [cited at p.22, 34]

[Fur11] Furniss, D., Blandford, A., and Mayer, A. *Unremarkable errors: low-level disturbances in infusion pump use.* In: *Proceedings of the 25th BCS Conference on Human-Computer Interaction*, BCS-HCI '11. British Computer Society, Swinton, UK, UK, page 197204. URL http://dl.acm.org/citation.cfm?id=2305316.2305353. [cited at p.131]

[Gan03] Gandhi, T. K., Weingart, S. N., Borus, J., Seger, A. C., Peterson, J., Burdick, E., Seger, D. L., Shu, K., Federico, F., Leape, L. L., and Bates, D. W. *Adverse drug events in ambulatory care.* The New England journal of medicine, **volume 348, no. 16**:(2003) pages 1556–1564. ISSN 1533-4406. doi:10.1056/NEJMsa020703. PMID: 12700376. [cited at p.3]

[Gim10] Gimblett, A. and Thimbleby, H. *User interface model discovery.* In: *Proceedings of the 2nd ACM SIGCHI symposium on Engineering interactive computing systems - EICS'10.* Berlin, Germany, pages 145 – 154. doi:10.1145/1822018.1822041. URL http://portal.acm.org/citation.cfm?id=1822018.1822041. [cited at p.7, 76, 125]

[gra02] *Model 500 and micro 505 volumetric infusion pump : Instruction manual* (**2002**). [cited at p.56]

[Har68] Hart, P. E., Nilsson, N. J., and Raphael, B. *A formal basis for the heuristic determination of minimum cost paths.* IEEE Transactions on Systems Science and Cybernetics, **volume 4, no. 2**:(1968) pages 100–107. ISSN 0536-1567. doi:10.1109/TSSC.1968.300136. [cited at p.77]

[Hic52] Hick, W. E. *On the rate of gain of information.* Quarterly Journal of Experimental Psychology, **volume 4, no. 1**:(1952) pages 11–26. ISSN 0033-555X. [cited at p.44]

[HWT11] Harold W. Thimbleby, A. G. *Dependable keyed data entry for interactive systems.* ECEASST, **volume 45**. [cited at p.30]

[HYM53] HYMAN, R. *Stimulus information as a determinant of reaction time.* Journal of experimental psychology, **volume 45, no. 3**:(1953) pages 188–196. ISSN 0022-1015. PMID: 13052851. [cited at p.44]

[Ins05] Institute For Safe Medication Practices. *Lowering the risk of medication errors: Independent double checks.* ISMP Canada Safety Bulletin, **volume 5, no. 1**. [cited at p.32]

[Ins06] Institute For Safe Medication Practices. *ISMP's list of error-prone abbreviations, symbols, and dose designations.* URL `http://www.ismp.org/tools/abbreviations/`. (Last accessed July, 2013). [cited at p.31]

[Ins07] Institute For Safe Medication Practices. *Fluorouracil incident root cause analysis.* URL `http://www.ismp-canada.org/download/reports/FluorouracilIncidentMay2007.pdf`. (Last accessed July, 2013). [cited at p.33]

[ISM06] ISMP. *Double key bounce and double keying errors.* In: *Medication Safety Alert! Acute Care Edition.* URL `http://www.ismp.org/newsletters/acutecare/articles/20060112.asp`. (Last accessed July, 2013). [cited at p.4, 99]

[Iso02] Isokoski, P. and Kaki, M. *Comparison of two touchpad-based methods for numeric entry.* In: *CHI '02: Proceedings of the SIGCHI conference on Human factors in computing systems.* ACM, New York, NY, USA. ISBN 1-58113-453-3, pages 25–32. doi:http://doi.acm.org/10.1145/503376.503382. [cited at p.24]

[Jag03] Jagacinski, R. J. and Flach, J. *Control theory for humans: quantitative approaches to modeling performance.* Routledge (**2003**). ISBN 9780805822922. [cited at p.50, 51]

[Jan10] Jani, Y. H., Barber, N., and Wong, I. C. K. *Paediatric dosing errors before and after electronic prescribing.* Quality and Safety in Health Care, **volume 19, no. 4**:(2010) pages 337 –340. [cited at p.99]

[Jar02] Jarman, H., Jacobs, E., and Zielinski, V. *Medication study supports registered nurses' competence for single checking.* International journal of nursing practice, **volume 8, no. 6**:(2002) pages 330–335. ISSN 1322-7114. PMID: 12390586. [cited at p.32]

[Jon06] Jones, C., O'Hearn, P., and Woodcock, J. *Verified software: a grand challenge.* Computer, **volume 39, no. 4**:(2006) pages 93–95. ISSN 0018-9162. doi:10.1109/MC.2006.145. [cited at p.31]

[Keo05] Keohane, C. A., Hayes, J., Saniuk, C., Rothschild, J. M., and Bates, D. W. *Intravenous medication safety and smart infusion systems: lessons learned and future opportunities.* Journal of infusion nursing: the official publication of the Infusion Nurses Society, **volume 28, no. 5**:(2005) pages 321–328. ISSN 1533-1458. URL `http://www.ncbi.nlm.nih.gov/pubmed/16205498`. PMID: 16205498. [cited at p.4]

[Kre89] Kreifeldt, J. G., Levine, S. L., and Iyengar, C. *Reduced keyboard designs using disambiguation.* Proceedings of the Human Factors and Ergonomics Society Annual Meeting, **volume 33, no. 6**:(1989) pages 441–444. ISSN 1541-9312,. doi:10.1518/107118189786759642. URL `http://pro.sagepub.com/content/33/6/441`. [cited at p.19]

[Kri07] Kristensson, P. O. and Zhai, S. *Learning shape writing by game playing.* In: *CHI '07 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '07. ACM, New York, NY, USA. ISBN 978-1-59593-642-4, pages 1971–1976. doi:10.1145/1240866.1240934. URL `http://doi.acm.org/10.1145/1240866.1240934`. [cited at p.19]

[Kro01] Kroemer, K. H. *Keyboards and keying an annotated bibliography of the literature from 1878 to 1999.* Universal Access in the Information Society, **volume 1, no. 2**:(2001) pages 99–160. ISSN 1615-5289. [cited at p.2, 18]

[Lee99] Lee, E. S. *Essays about computer security.* Centre for Communications Systems Research Cambridge, © Cambridge. [cited at p.55]

[Lee12a] Lee, P., Monroy Aceves, C., Oladimeji, P., and Thimbleby, H. *Are prescribed infusions running as intended?* In: *Third National Infusion and Vascular Access Society Conference.* London. [cited at p.58]

[Lee12b] Lee, P. T., Thompson, F., and Thimbleby, H. *Analysis of infusion pump error logs and their significance for healthcare.* British Journal of Nursing, **volume 21, no. 8**:(2012) pages S12–S22. [cited at p.58]

[Les02] Lesar, T. S. *Tenfold medication dose prescribing errors.* The Annals of Pharmacotherapy, **volume 36, no. 12**:(2002) pages 1833–1839. ISSN 1060-0280. URL `http://www.ncbi.nlm.nih.gov/pubmed/12452740`. PMID: 12452740. [cited at p.30]

[Lev66] Levenshtein, V. *Binary codes capable of correcting deletions, insertions and reversals.* Soviet Physics Doklady., **volume 10, no. 8**:(1966) pages 707–710. [cited at p.20]

[Lin13] Lin, C.-J. and Wu, C. *Reactions, accuracy and response complexity of numerical typing on touch screens.* Ergonomics, **volume 56, no. 5**:(2013) pages 818–831. ISSN 0014-0139. PMID: 23597044. [cited at p.24]

[Lis05] Lisby, M., Nielsen, L. P., and Mainz, J. *Errors in the medication process: frequency, type, and potential clinical consequences.* International Journal for Quality in Health Care, **volume 17, no. 1**:(2005) pages 15–22. ISSN 1353-4505, 1464-3677. [cited at p.3]

[Lut55] Lutz, M. C. and Chapanis, A. *Expected locations of digits and letters on ten-button keysets.* Journal of Applied Psychology, **volume 39, no. 5**:(1955) pages 314–317. [cited at p.27]

[MA13] Monroy Aceves, C., Oladimeji, P., Thimbleby, H., and Lee, P. *Are prescribed infusions running as intended? quantitative analysis of log files from infusion pumps used in a large acute NHS hospital.* British Journal of Nursing. In press. [cited at p.58]

[Mac90] Mackinlay, J., Card, S. K., and Robertson, G. G. *A semantic analysis of the design space of input devices.* Hum.-Comput. Interact., **volume 5, no. 2**:(1990) pages 145–190. ISSN 0737-0024. doi: 10.1207/s15327051hci0502&3_2. URL `http://dx.doi.org/10.1207/s15327051hci0502&3_2`. [cited at p.37, 50]

[Mac91] MacLean, A., Young, R. M., Bellotti, V. M., and Moran, T. P. *Questions, options, and criteria: Elements of design space analysis.* Human-Computer Interaction, **volume 6, no. 3-4**:(1991) pages 201–250. ISSN 0737-0024. doi:10.1080/07370024.1991.9667168. URL `http://www.tandfonline.com/doi/abs/10.1080/07370024.1991.9667168`. [cited at p.38, 39]

[Mac94] MacKenzie, I. S., Nonnecke, B., Riddersma, S., McQueen, C., and Meltz, M. *Alphanumeric entry on pen-based computers.* Int. J. Hum.-Comput. Stud., **volume 41, no. 5**:(1994) pages 775–792. ISSN 1071-5819. [cited at p.23]

[Mac02a] MacKenzie, I. S. *KSPC (keystrokes per character) as a characteristic of text entry techniques.* In: *Mobile HCI '02: Proceedings of the 4th International Symposium on Mobile Human-Computer Interaction.* Springer-Verlag, London, UK. ISBN 3-540-44189-1, pages 195–210. [cited at p.20]

[Mac02b] MacKenzie, I. S. and Soukoreff, R. W. *Text entry for mobile computing: Models and Methods, Theory and practice.* Human-Computer Interaction, **volume 17, no. 2-3**:(2002) pages 147–198. ISSN 0737-0024.

doi:10.1080/07370024.2002.9667313. URL http://www.tandfonline.com/doi/abs/10.1080/07370024.2002.9667313. [cited at p.19]

[Mac02c] MacKenzie, S. *Mobile text entry using three keys.* In: *Proceedings of the second Nordic conference on Human-computer interaction*, NordiCHI '02. ACM, New York, NY, USA. ISBN 1-58113-616-1, pages 27–34. doi: 10.1145/572020.572025. URL http://doi.acm.org/10.1145/572020.572025. [cited at p.19]

[Mar92] Martin, E. *The Calculating Machines: Their History and Development.* MIT Press (**May 1992**). ISBN 0262132788. [cited at p.13, 15]

[Mar96] Marteniuk, R. G., Ivens, C. J., and Brown, B. E. *Are there task specific performance effects for differently configured numeric keypads?* Applied Ergonomics, **volume 27, no. 5**:(1996) pages 321–325. ISSN 0003-6870. doi:10.1016/0003-6870(96)00024-5. URL http://www.sciencedirect.com/science/article/B6V1W-3VTWBKX-4/2/c9607cb19034cf78a312f79552c6fbc4. [cited at p.27, 122]

[McQ95] McQueen, C., MacKenzie, S. I., and Zhang, S. X. *An extended study of numeric entry on pen-based computers.* In: *Proceedings of Graphics Interface '95.* Toronto, pages 215–222. URL http://www.yorku.ca/mack/GI95.html. [cited at p.24]

[Med13] Medicines and Healthcare Products Regulatory Agency. *Infusion Systems.* Technical report (**December 2013**). URL http://www.mhra.gov.uk/home/groups/dts-iac/documents/publication/con007322.pdf. [cited at p.62]

[Men92] Menninger, K. *Number Words and Number Symbols: A Cultural History of Numbers.* Dover Publications (**May 1992**). ISBN 9780486270968. [cited at p.2, 11]

[Mor04] Morimoto, T., Gandhi, T. K., Seger, A. C., Hsieh, T. C., and Bates, D. W. *Adverse drug events and medication errors: detection and classification methods.* Quality and Safety in Health Care, **volume 13, no. 4**:(2004) pages 306–314. ISSN , 2044-5423. [cited at p.3]

[New68] Newman, W. M. *A graphical technique for numerical input.* The Computer Journal, **volume 11, no. 1**:(1968) pages 63–64. ISSN 0010-4620, 1460-2067. doi:10.1093/comjnl/11.1.63. URL `http://comjnl.oxfordjournals.org/content/11/1/63`. [cited at p.21]

[Nie92] Nielsen, J. *Finding usability problems through heuristic evaluation.* In: *CHI '92: Proceedings of the SIGCHI conference on Human factors in computing systems.* ACM, New York, NY, USA. ISBN 0-89791-513-5, pages 373–380. [cited at p.41]

[Nor02] Norman, D. A. *The design of everyday things.* Basic Books (**August 2002**). [cited at p.28, 30]

[Ola11] Oladimeji, P., Thimbleby, H., and Cox, A. *Number entry interfaces and their effects on error detection.* In: *Proceedings of the 13th IFIP TC 13 international conference on Human-computer interaction - Volume Part IV*, INTERACT'11. Springer-Verlag, Berlin, Heidelberg. ISBN 978-3-642-23767-6, page 178185. Conference. [cited at p.xviii, 127, 138, 152]

[Ols08] Olsen, K. A. *The $100,000 keying error.* Computer, **volume 41, no. 4**:(2008) pages 108 –107. ISSN 0018-9162. doi:10.1109/MC.2008.135. [cited at p.29]

[Oul13] Oulasvirta, A., Reichel, A., Li, W., Zhang, Y., Bachynskyi, M., Vertanen, K., and Kristensson, P. O. *Improving two-thumb text entry on touchscreen devices.* In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13. ACM, New York, NY, USA. ISBN 978-1-4503-1899-0, pages 2765–2774. doi:10.1145/2470654.2481383. URL `http://doi.acm.org/10.1145/2470654.2481383`. [cited at p.19]

[Pox04] Poxon, I. *Infusion pumps - how to pick the best pump for the delivery of fluids.* (**July 2004**). URL `http://www.nursingtimes.net/infusion-pumps-how-to-pick-the-best-pump-for-the-delivery-of-fluids/200125.article`. [cited at p.62]

[Rea90] Reason, J. *Human Error.* Cambridge University Press (**1990**). [cited at p.28, 100]

[Res03] Resar, R. K., Rozich, J. D., and Classen, D. *Methodology and rationale for the measurement of harm with trigger tools.* Quality and Safety in Health Care, **volume 12, no. suppl 2**:(2003) pages ii39–ii45. ISSN , 2044-5423. [cited at p.5]

[Rey92] Reynolds-Haertle, R. A. and McBride, R. *Single vs. double data entry in CAST.* Controlled Clinical Trials, **volume 13, no. 6**:(1992) pages 487–494. ISSN 0197-2456. [cited at p.32]

[Rin99] Rinck, M. *Memory for everyday objects: where are the digits on numerical keypads?* Applied Cognitive Psychology, **volume 13, no. 4**:(1999) pages 329–350. ISSN 1099-0720. [cited at p.27]

[R.L60] R.L. Deininger. *Human factors engineering studies of the design and use of pushbutton telephone sets.* Bell System Technical Journal, **volume 39, no. 4**:(1960) pages 995–1012. [cited at p.121]

[Rot05] Rothschild, J. M., Keohane, C. A., Cook, E. F., Orav, E. J., Burdick, E., Thompson, S., Hayes, J., and Bates, D. W. *A controlled trial of smart infusion pumps to improve medication safety in critically ill patients*.* Critical Care Medicine, **volume 33, no. 3**:(2005) pages 533–540. ISSN 0090-3493. [cited at p.3]

[San04] Sandnes, F. E., Thorkildssen, H. W., Arvei, A., and Buverud, J. O. *Techniques for fast and easy mobile text-entry with three-keys.* In: *HICSS '04: Proceedings of the Proceedings of the 37th Annual Hawaii International Conference on System Sciences (HICSS'04) - Track 9.* IEEE Computer Society, Washington, DC, USA. ISBN 0-7695-2056-1, page 90286.2. [cited at p.2, 18, 19]

[Shn04] Shneiderman, B. and Plaisant, C. *Designing the User Interface: Strategies for Effective Human-Computer Interaction (4th Edition).* Pearson Addison Wesley (**2004**). ISBN 0321197860. [cited at p.41]

[Sil00] Silfverberg, M., MacKenzie, I. S., and Korhonen, P. *Predicting text entry speed on mobile phones.* In: *CHI'00: Proceedings of the SIGCHI conference on Human factors in computing systems.* ACM, New York, NY, USA. ISBN 1-58113-216-6, pages 9–16. doi:http://doi.acm.org/10.1145/332040.332044. [cited at p.2, 18]

[Sou01a] Soukoreff, R. W. and MacKenzie, I. S. *Measuring errors in text entry tasks: an application of the levenshtein string distance statistic.* In: *CHI '01: CHI '01 extended abstracts on Human factors in computing systems.* ACM, New York, NY, USA. ISBN 1-58113-340-5, pages 319–320. [cited at p.20]

[Sou01b] Soukoreff, R. W. and MacKenzie, I. S. *Measuring errors in text entry tasks: an application of the levenshtein string distance statistic.* In: *CHI'01: CHI'01 extended abstracts on Human factors in computing systems.* ACM, New York, NY, USA. ISBN 1-58113-340-5, pages 319–320. doi:http://doi.acm.org/10.1145/634067.634256. [cited at p.20]

[Sou03] Soukoreff, R. W. and MacKenzie, I. S. *Metrics for text entry research: an evaluation of MSD and KSPC, and a new unified error metric.* In: *CHI'03: Proceedings of the SIGCHI conference on Human factors in computing systems.* ACM, New York, NY, USA. ISBN 1-58113-630-7, pages 113–120. doi:http://doi.acm.org/10.1145/642611.642632. [cited at p.2, 18]

[Str93] Straub, H. R. and Granaas, M. M. *Task-specific preference for numeric keypads.* Applied Ergonomics, **volume 24, no. 4**:(1993) pages 289–290. ISSN 0003-6870. doi:10.1016/0003-6870(93)90465-L. URL `http://www.sciencedirect.com/science/article/pii/000368709390465L`. [cited at p.27]

[Sye06] Syed, S., Paul, J. E., Hueftlein, M., Kampf, M., and McLean, R. F. *Morphine overdose from error propagation on an acute pain service.* Canadian Journal of Anesthesia/Journal canadien d'anesthsie, **volume 53, no. 6**:(2006) pages 586–590. ISSN 0832-610X. [cited at p.5]

[Thi07] Thimbleby, H. *Press On: Priciples of Interaction Programming.* The MIT Press, first edition (**2007**). [cited at p.7, 125]

[Thi09a] Thimbleby, H. *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. ACM, New York, NY, USA, pages 221–230. [cited at p.76]

[Thi09b] Thimbleby, H. and **Patrick Oladimeji**. *Social network analysis and interactive device design analysis.* In: *Proceedings of the 1st ACM SIGCHI symposium on Engineering interactive computing systems - EICS'09.* Pittsburgh, PA, USA, pages 91–100. doi:10.1145/1570433.1570453. URL `http://portal.acm.org/citation.cfm?id=1570453`. [cited at p.7, 76]

[Thi10a] Thimbleby, H. *Interactive systems need safety locks.* In: *Information Technology Interfaces (ITI), 2010 32nd International Conference on.* ISBN 1330-1012, pages 29–36. [cited at p.116]

[Thi10b] Thimbleby, H. and Cairns, P. *Reducing number entry errors: solving a widespread, serious problem.* Journal of the Royal Society Interface, **volume 7, no. 51**:(2010) pages 1429–1439. ISSN 1742-5689. doi:10.1098/rsif.2010.0112. PMID: 20375037 PMCID: 2935596. [cited at p.4, 30, 41, 116, 152]

[Tho79] Thornton, R. W. *The number wheel: A tablet based valuator for interactive three-dimensional positioning.* In: *Proceedings of the 6th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '79. ACM, New York, NY, USA. ISBN 0-89791-004-4, pages 102–107. [cited at p.22, 23, 34]

[Tur21] Turck, J. A. V. *Origin of Modern Calculating Machines: A Chronicle of the Evolution of the Principles that Form the Generic Make Up of the Modern Calculating Machine.* The Western society of engineers (**1921**). [cited at p.13]

[U03] U, D. *Double-Checking: does it work?* ISMP Medication Safety Alerts, **volume 56, no. 3**:(2003) pages 167–169. [cited at p.32]

[Vai06] Vaida, A. J., Grissinger, M., Urbanski, B., and Mitchell, J. F. *PCA drug libraries: Designing, implementing, and analyzing CQI reports to optimize patient safety.* URL `http://www.ismp.org/profdevelopment/PCADrugLibrariesforwebce.pdf`. (Last accessed July, 2013). [cited at p.99]

[Vic03] Vicente, K. J., Kada-Bekhaled, K., Hillel, G., Cassano, A., and Orser, B. A. *Programming errors contribute to death from patient-controlled analgesia: case report and estimate of probability.* Canadian Journal of Anesthesia, **volume 50, no. 4**:(2003) page 328332. ISSN 0832-610X. [cited at p.4, 5, 99]

[Wan11] Wang, S., Lin, C.-J., Wu, C., and Chaovalitwongse, W. A. *Early detection of numerical typing errors using data mining techniques.* IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans, **volume 41, no. 6**:(2011) pages 1199–1212. ISSN 1083-4427. [cited at p.32]

[Web12] Webster, J., Eslambolchilar, P., and Thimbleby, H. *From rotary telephones to universal number entry systems: can the past re-shape the future?* In: *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, UbiComp '12. ACM, New York, NY, USA. ISBN 978-1-4503-1224-0, pages 596–597. [cited at p.17]

[Wes11] Westbrook, J. I., Rob, M. I., Woods, A., and Parry, D. *Errors in the administration of intravenous medications in hospital and the role of correct procedures and nurse experience.* BMJ Quality & Safety. doi: 10.1136/bmjqs-2011-000089. [cited at p.4]

[Wis11] Wiseman, S., Cairns, P., and Cox, A. *A taxonomy of number entry error.* In: *Proceedings of the 25th BCS Conference on Human-Computer Interaction*, BCS-HCI '11. British Computer Society, Swinton, UK, UK, pages 187–196. [cited at p.29, 112, 127, 138]

[Wis12] Wiseman, S., Cox, A., and Brumby, D. *Designing for the task: what numbers are really used in hospitals?* In: *Proceedings of the 2012 ACM annual conference extended abstracts on Human Factors in Computing Systems Extended Abstracts*, CHI EA '12. ACM, New York, NY, USA. ISBN 978-1-4503-1016-1, pages 1733–1738. [cited at p.56]

[Wis13a] Wiseman, S., Cox, A. L., and Brumby, D. P. *Designing devices with the task in mind: which numbers are really used in hospitals?* Human factors, **volume 55, no. 1**:(2013) pages 61–74. ISSN 0018-7208. PMID: 23516794. [cited at p.74]

[Wis13b] Wiseman, S., Cox, A. L., Brumby, D. P., Gould, S. J., and O'Carroll, S. *Using checksums to detect number entry error.* In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13. ACM, New York, NY, USA. ISBN 978-1-4503-1899-0, pages 2403 – 2406. [cited at p.34]

[Zha04a] Zhai, S., Kong, J., and Ren, X. *Speed-accuracy tradeoff in fitts' law tasks: On the equivalency of actual and nominal pointing precision.* Int. J. Hum.-Comput. Stud., **volume 61, no. 6**:(2004) page 823856. ISSN 1071-5819. doi:10.1016/j.ijhcs.2004.09.007. URL `http://dx.doi.org/10.1016/j.ijhcs.2004.09.007`. [cited at p.114]

[Zha04b] Zhang, J., Patel, V. L., Johnson, T. R., and Shortliffe, E. H. *A cognitive taxonomy of medical errors.* J. of Biomedical Informatics, **volume 37, no. 3**:(2004) pages 193–204. ISSN 1532-0464. doi:http://dx.doi.org/10.1016/j.jbi.2004.04.004. [cited at p.5]

[Zha12] Zhai, S. and Kristensson, P. O. *The word-gesture keyboard: reimagining keyboard interaction.* Commun. ACM, **volume 55, no. 9**:(2012) pages 91–101. ISSN 0001-0782. doi:10.1145/2330667.2330689. URL `http://doi.acm.org/10.1145/2330667.2330689`. [cited at p.19]