INTERACTIVE NUMBERS
A GRAND CHALLENGE

Harold Thimbleby
Swansea University
Wales, SA2 8PP

ABSTRACT

Numbers are used everywhere, but today numbers are mostly created and used interactively; they are not just passive written objects. People interact with numbers in almost every area of life. However, different styles of interactive number have different design trade-offs, particularly when human error is considered. To date, human error in interactive numbers has hardly been explored, resulting in most computer support for interactive numbers (from calculators to medical devices) being mediocre. Interactive number systems should be usable (effective, efficient, etc), free of idiosyncrasies and be demonstrably free from unacceptable levels of risk, particularly in environments where there are several different interactive number entry systems. We argue that defining "good" interactive number systems is indeed a serious challenge, but that substantial progress is being and can continue to be made, perhaps eventually resulting in an international standard for interactive numbers with solid empirical evidence of its value.

KEYWORDS

Interactive numbers; user interface design; safety-critical user interfaces; medical systems; spreadsheets; calculators.

6. INTRODUCTION

Written around 1800 BC, the Rhind papyrus claims that

> "accurate reckoning is the entrance into the knowledge of all existing things and all obscure secrets"

Indeed, numbers are the bedrock of civilisation. Mathematics has progressed since the earliest records of the denary number system, due to the Egyptians in 3100 BC. Notable steps to modernity include Leonardo Fibonacci of Pisa, who introduced Arabic notation (i.e., numbers written in modern Western digits, 0123456789) to the west with his *Liber Abaci* (The Book of Calculation, 1202); Gottlob Frege's *Die Grundlagen der Arithmetik* (The Foundations of Arithmetic, 1884); then Kurt Gödel and Alan Turing's arithmetisation of facts (1937), arguably the foundational result that allows computers to "do anything." On the more practical side, Luca Paccioli's *Summa de arithmetica, geometria, proportioni et proportionalita* (Everything about Arithmetic, Geometry, and Proportions, 1494) introduced double-entry book keeping, an idea that revolutionised accounting, and which lead directly to modern spreadsheets, via Dan Bricklin's VisiCalc (1977) to today's familiar Microsoft Excel (introduced in 1985).

History thus records a very long gestation of numbers as written objects (numerals) through to the modern concept of *interactive numbers*, a concept this paper introduces and explores. Compared to the widespread interest in written numbers and notations, to date very little attention has been paid to interactive numbers as such, with only a few exceptions, including the abacus, counting boards, slide rules and, in modern times, digital calculators, cash machines and of course computers. We shall show that interactive numbers are a surprisingly complex and rich area, and due to their ubiquity and use in high dependability applications, an area highly deserving of serious work: interactive numbers are a significant and promising research area. We contend that with more attention to interactive numbers, many everyday and critical systems could be even better.

Interactive numbers are created through interaction rather than just appearing in a fixed, written form; the new term is intended to focus attention on the non-trivial process of transferring a number from a human (perhaps though intermediaries and perhaps involving calculation) to a system.

With an interactive number we are explicitly concerned with how the user creates the number, considering the process as broken down into steps and possible errors and error corrections. For example:

- Using a spreadsheet, a user may key in the number 2.3. This is not something "that just happens" but involves selecting a cell in the spreadsheet, clicking, pressing a sequence of keys on the keyboard, then clicking elsewhere on the spreadsheet to complete the number entry. Each user action is *actively* handled by a computer program — unlike the simple case of writing a number on paper, where the paper does essentially nothing other than hold ink. Note that in a spreadsheet, a cell may not contain any number before the user starts to enter a number.
- Using a television (TV), the up/down keys and number keys can change the channel number. Typically pressing digit *n* will change to channel *n* (which in turn is mapped by the TV into a UHF frequency as defined during device set up). But if the user presses another digit, *m*, "quickly," the channel will be $10n+m$, not *m*, or unless $10n+m$ is greater than some small number (typically 30) the channel will be *m*. On some TVs, if *n* is pressed the TV will display "*n*–" but not immediately change channel; if the user immediately keys *m*, the display will become "*nm*," that is, showing the Arabic numeral for the number $10n+m$ (assuming $10n+m<30$) and the channel will change, or the user can wait, in which case — depending on the design — either the channel changes to *n* or the display "*n*–" simply disappears and nothing further happens. These interaction rules are further complicated to account for the user keying up/down during number entry. Note that in a TV, unlike a spreadsheet cell, there is always a concept of "current number," namely the current channel.
- In a hospital, a consultant may think of an appropriate drug dose for a patient and records this in an electronic patient record (EPR), hopefully without uncorrected errors. The pharmacy is informed, and prepares a drug. A nurse then programs an automatic delivery system (typically an infusion pump) to deliver the drug at the stated dose. The pharmacy may prepare the drug in standard sizes, meaning that the nurse will have to perform a calculation. Since there is a calculation involved, some hospitals will require two nurses to independently agree on the calculation; other hospitals think this is not as reliable as using just one nurse who therefore takes more care over the calculation as they are the only person responsible for the number. The calculated number is then entered in the drug delivery system, which will run Dose Error Reduction Software (DERS) to try to trap errors. Unfortunately, if the DERS, itself a complex sub-system, has been set up with the wrong drug, it may give a false sense of security. The patient may also have an active part in dosing, and be able to increase the dose on demand (perhaps for pain control), and a nurse may from time to time adjust the dose depending on the patient's response to the drug. In short, then, a number (sometimes a formula) has been transmitted in a very complex way from the consultant's head to the patient's infusion device, with numerous opportunities for interaction — and error.
- Aviation accidents are frequently caused by problems with interactive numbers. The Air Inter Flight 148 in January 1992 performed a "controlled flight into terrain" killing 87 people in part because the autopilot (flight control unit, FCU) was probably in vertical speed mode instead of flight angle mode; the 33 the pilot entered was interpreted as 3,300 feet per minute descent not the intended 3.3 degree angle of descent (at the plane speed, equivalent to about 800 feet per minute). Following the crash, the FCU on A320s was modified to display vertical speed to four digits and angle to 2 digits, to make them less confusable.

In contrast to these interactive scenarios, it is conventional to think of numbers as static. Thus a written number, say 23.4, is viewed as a static representation of a number, and the process by which it was created or written down is of very little interest (except to school teachers). Ergonomics (human factors) has invested considerable research into readable numbers and good font design, notably helping develop OCR (optical character recognition) fonts that balance machine and human readability.

In some fields, notably medicine, great care is taken to ensure that numbers are written so that they are legible and unlikely to be misread.

Thus the Institute of Safe Medication Practices (ISMP, 2006) recommends that written numbers do not have "naked" decimal points (as in ".5" — because this may be misread as 5) and must not have trailing

zeros after the decimal point (as in “1.0” — because this may be misread as 10). Such rules help ensure that drug prescriptions are more dependable. The ISMP urge that “computer software vendors [should] make changes in electronic order entry programs” to follow their rules.

The ISMP and ISO rules are naturally considered as static rules: thus, under the ISMP rules, a written number is valid or invalid as an ISMP number. But what happens when a user keys in an interactive number, say 2.05? It is not obvious how and when to apply the rules: “2” is a valid ISMP number, “2.” isn’t, “2.0” isn’t, and finally “2.05” is. Worse, the ISMP rules are designed to reduce human error, but human error is unavoidable, and eventually users will create erroneous numbers. How should they be handled? Some interactive number systems display “0.” when they are switched on, and the display does not change if the user keys “.” (a decimal point); thus if the display shows “0.” the user cannot tell what pressing say “5” will do — the display will change to 0.5 or 5., depending on the mode the device is in. This is a simple example of the subtleties of interactive numbers.

When the user wants to enter 2.3 and that is what they do, the process may seem trivial. However, humans make slips, and occasionally a user intending to enter 2.3 will key in 3.2, 2..3 or some other sequence that is erroneous. What happens next is a question for interactive number entry. As it happens almost all interactive systems ignore the user’s error even when it is an obvious syntax error, like two decimal points (H.Thimbleby, 2010). Syntax errors are often ignored, treating the number as 0 or 2.3 as if nothing had happened. Some systems do use “data validation” (both syntactic and range checks) and may reject an error like 2..3 and hence prohibit the user from proceeding; nevertheless doing this is likely to increase the user’s stress and the likelihood of their making further errors! Unfortunately many systems do no validation (H.Thimbleby & Cairns, 2010).

To enter a number such as 2.3 the user employs an Arabic numeral keyboard, perhaps in telephone or calculator layout (and both layouts have minor variations). Which keyboard layout is better? In an environment that has both, is the “better” keyboard really better, or does it introduce more transfer errors? Or does variety encourage the user to pay more attention? Simply, we do not know.

We do know that users make slips, such as transposition errors (e.g., keying 2.3 as 23.). We know that very few interactive systems detect even obvious slips, such as two decimal points. We know that validating input to ISMP rules, would halve the out by ten error rate (where the entered number is ten times out from the intended number; this occurs with 2.3 entered as 23. — which happens to be a number with a naked decimal point). It is estimated that interactive numbers — not just handwriting — using ISMP rules would halve the fatality rate from miscalculations of drug doses (H.Thimbleby & Cairns, 2010).

Although they are fast, it is not obvious that Arabic numeral keyboards are the best way to enter interactive numbers — for some meaning of “best.”

Arabic numerals will be “best” for some applications (e.g., time paced copy typing) but we do not know the criteria because not enough research has been done. It may seem that Arabic numerals are obviously best for cases when numbers are used as names (examples being security codes, telephone numbers, credit card numbers), but the tradeoffs have not been explored. Even though we know that words are easier to remember, why are PIN codes numbers rather than words? It has, hitherto, seemed too obvious to need empirical research — and it would be expensive (because larger keyboards would be needed) to discover that numbers are suboptimal. Sadly, the cost of user error is often pushed on to the user, so until more research is done there is no incentive to improve the user interface. In high consequence, mission and safety critical areas, we think these widespread assumptions need reviewing and basing on solid empirical foundations.

One alternative is the “up/down” key approach, where pressing a “Λ” (or similar) key increases the number, and pressing a “V” key decreases the number. Like Arabic keyboards, there are many variations on this theme: for example, holding the Λ key down may cause the number to increase continually, and holding it down for longer may cause the number to increase faster. There may be several up and down keys, each with different rates (for example, changing different digits of the displayed number). Again, we do not know which styles are “better,” and if so for which applications.

Oladimeji *at al* (2011) have performed eye-tracking experiments on entering numbers. For Arabic style keyboards, 91% fixation time is on the keyboard and 9% on the display; in contrast for up/down interactive numbers it is 25% and 75% respectively. Users give much more attention to the display in an up/down system. Crucially, their unnoticed and uncorrected error rate is 6 times lower.

Users of up/down interactive number systems pay attention to the number they are entering, and moreover they have an “error correcting strategy” for normal use too: for example, to enter 95, they might quickly enter 100 then decrement it to 95. It seems clear that for any number entry where it is important that

the intended number is entered should be based on an up/down interface. Of course, unlike Arabic keys, up/down keyboards do not lend themselves to copy typing (typing while looking at text to be copied), and they may be slower. There is clearly a tradeoff, though 5 times the time to enter a number (6 seconds longer) compared to reducing the risk of death in a hospital (or an aircraft flying into the ground) might seem a good tradeoff to make.

Part of the time Oladimeji's experimental participants spent with the up/down interface was experimenting with it — making sure how it worked. In general, experimenting with an interface is a factor that would be expected to improve the reliability of interaction and (b) reduce with experience of consistent user interface design. Conversely, the lack of time spent experimenting with the deceptively "familiar" Arabic style of user interface means that users will be unfamiliar with the idiosyncratic ways these systems interact, especially how they handle errors. H.Thimbleby (2010) has shown an example where a user corrects an error to enter "5" but the system handled the user's error-correction in an unusual way and recorded "0.5" The device log therefore showed the incorrect value 0.5, so *obviously the user* made the error, as the log should have shown 5. We would argue that the device design (or a bug in the device implementation) led to the user being wrongly blamed for the incident.

All up/down interfaces we have explored have complicated features, such as time dependencies and hidden state. We have therefore designed a "rational" incremental up/down user interface, carefully following best design practice. Interestingly, to do this raises new design questions we do not know the answers for.

The science of human-computer interaction has ignored the devil in the details; it has assumed number entry is trivial and not worthy of attention. We have yet to prove that our design is formally correct, certainly an essential step towards improving dependable interactive numbers. In fact, it may be that the design is unique (up to certain trivial variations), and thus a *de facto* standard for improved interactive numbers.

We have also developed a principle-driven user interface for Arabic number entry; ironically, the current paper describing it was rejected by referees referring to it as merely "incremental" research! However, it is again noteworthy that a rigorously defined user interface even for something as "simple" as interactive numbers raises questions that there are no answers. More research is needed; conversely, it is increasingly apparent that all interactive number designs so far developed fall far short of the design standards that we should aspire to. Interactive numbers have been taken for granted.

See H.Thimbleby (2000) and W.Thimbleby & H.Thimbleby (2008) for wider discussion of number entry, calculators and mathematical user interfaces in general (one would be wrong in thinking they would obey mathematical laws).

W.Thimbleby's (2010) user interface uses a touch screen, and explores interactive numbers at a lower-level than ususual, for instance allowing "ink editing" to edit a 1 into a 4 by overwriting the stroke ∠. In an experiment comparing a conventional Arabic keyboard calculator and his interactive number interface, W.Thimbleby (2010) had *no* errors, compared to about 60% for the conventional calculator.

7. SELECTED ISSUES IN INTERACTIVE NUMBERS

2.1 Bugs in Design or Problems in *any* design rules?

The ISMP (2006) rules should apply to interactive systems and they forbid naked decimals. Thus as the user keys "1.2" what do the rules say should be displayed after the decimal point has been pressed but before the 2 has been pressed? If the user enters an ISMP-invalid number such as "2.0" what should be displayed? As mentioned above, the rules assume that numbers are static and not interactive. Yet a system implementor has to make decisions at this level — what can be done?

Time outs are an obvious solution. After a time *t* of inactivity, the display will comply with ISMP rules. Unfortunately, for any time t selected, one can envisage cases where the time is too short or too long. Clearly a system should not reset if it interferes with the user; thus *t* should be longer than a typical interaction. Ten seconds? Yet a handover (e.g., one nurse replacing another) can be done in under 10 seconds, so the second nurse will experience the partially-completed interaction of the first, when it would have been preferable to have done the time out so the second nurse could start with a clean sheet. Many other scenarios are possible. However easy they are to implement, time outs do not solve the interaction problems; they are a proxy for

knowing when the user has lost track of the system mode. Proximity sensors or other techniques may be better. Reducing the hidden state of the user interface will also help.

In short it is clear that the simple ISMP rules are not adequate as they stand for interactive numbers. Programmers have to implement interactive numbers, and the rules (i.e., program) has to work at a finer level of detail than ISMP envisaged.

Either there is an interesting research programme to determine the best interactive number rules, or (possibly) no such rules exist that are completely consistent. In the meantime, developers will continue to implement suboptimal interactive number systems — there is no choice, as we do not know how to do it!

2.2 Numbers versus Numerals

It is usual to distinguish between numerals and numbers; the numeral being a concrete representation of the abstract number value. Confusion arises because we cannot speak about a number without representing it in some way. Thus 27 is an Arabic numeral that represents the number value 27, the same number value as 26+1 in fact. In Roman numerals, the value 27 would be represented as XXVII. As a binary numeral, it would be 11011, and as an English numeral "twenty seven."

Since the user obviously creates a numeral, why do I use the terminology "interactive number" rather than "interactive numeral"? The reason is that the user is thinking of a number (which if you asked them, they might say in English or write down in Arabic numeral notation) and their intention is to get the number into the system they are using. They don't want the system to have the numeral "twenty seven" they want it to have the value that is represented by twenty seven.

Of course many systems use Arabic notation for numerals, so the user will come to little harm if they think of a number as its Arabic numeral. However there are occasions when the differences are subtle and in some applications important. Thus, at first sight 027, 27 and 27.0, while different numerals, appear to represent the same number. Computer systems may think otherwise; Mathematica makes the difference 27!–27.0!, which you might expect to be zero, to be almost 10^{14}; or in Java 27/4=6 but 27.0/4=6.75. In many programming languages, 027 (which seemingly has innocuous leading zeros) is 27 in octal, hence 23 in decimal, 4 less than the 27 programmer may have thought.

In some applications there may be no conventional representation of the number. A light dimmer would be an example: the user interacts with the dimmer to adjust the brightness of a light. The user perhaps does not think of this as an interactive number, but the user's interaction adjusts the power going to the light. An electrician could connect a meter, and the meter would show the numerical value of the brightness. If the user so wished, they could remember this number and set the light to the exact same brightness on another occasion. In fact if the meter was permanently connected the user's interaction would very obviously be interactive numbers. Some dimmers give the user a fixed number of choices, perhaps six: the user can interact to set the light strength anywhere from 0 (off) to 5 (full on).

2.3 Discrete versus Continuous Interaction

There is a considerable literature on number representation and processing (precision, resolution, stability, etc) in the numerical methods literature. We do not need to repeat it here except to note that it has concerned itself with number processing in a computer, without regard for the user. Much of the literature is relevant to interactive numbers however.

Interactive numbers can be *continuous*, taking on values that can, for the user, be imperceptibly close to each other, or *discrete*, taking on clearly distinct values. I've defined the terms from the user's point of view; continuous numbers may be simulated on a digital computer by working to a high precision (e.g., fixed point, rational or floating point numbers) and conversely, real numbers may be simulated (e.g., on a calculator) by 8 digits of discrete numbers that are all clearly distinct to the user. Sometimes continuous numbers will be simulated by whole numbers; for example, the accelerator pedal in a car (if implemented by digital electronics) converts the angle of the sensor into an integer, which in turn controls the throttle and hence the RPM and speed of the car (which is what the user is concerned about); to the user, both RPM and speed will appear to be continuous even though they are controlled by small integers.

Sometimes continuous numbers are made to seem like discrete numbers. A car radio volume control may be a knob with detents that give the user the impression of discrete volume levels. Presumably the detents improve the user experience when turning the knob.

Users interact with discrete numbers by causing step changes, for instance pressing the key 7 typically multiplies the number by 10 and adds 7 (so if the display is 2, pressing 7 changes it to 27). Users interact with continuous numbers by turning knobs, moving sliders, holding buttons down for an interval while the number changes.

Users tend to come to a discrete number interface with a clear goal of the number they wish to enter. Users tend to come to a continuous number interface with a view to finding the best number by experimenting.

The behaviour of a user interface may depend on what has happened before. If everything relevant to the user is displayed (e.g., the number itself, perhaps plus a cursor) then the user interface is *declarative*. Otherwise, a discrete interface has *modes* and a continuous interface has *hysteresis*. Of course, one can simulate the other, and sometimes it may be clearer to partition the behaviour of a single interface into both modes and hysteresis.

The mathematical theories generally most appropriate for analysis of discrete and continuous systems are *graph theory* (including finite state systems) and *control theory*.

2.3 Low Uncorrected Error Rates, Low Error Rates

With interactive numbers, we are not so much interested in user errors, as in the errors that the user makes that do not get corrected: *uncorrected errors*. Most uncorrected errors are *unnoticed errors*. Noticed errors that are corrected will obviously slow the user down, so a design tradeoff is to balance low uncorrected error rate (which is good) with loss of speed (which is generally bad). It is of course possible for a non-error to be accidentally "corrected" and hence create an error, that itself may or may not be corrected; more likely is for an attempted error correction to fail to successfully correct a noticed error. Clearly eye fixation with the number display is likely to be essential for dependable corrections.

In some applications, correct numbers are more important than speed, whereas in some, the numbers being "good enough" (however that is defined) faster is more important. Interestingly, a user interface that gives the impression of having a very low error rate may encourage the user to miss errors when they do occur: a higher error rate may make the user more vigilant, and more skilled at error correction.

It is usual to distinguish between uncorrected user error and the magnitude of the harm that results; simply quantifying uncorrected errors is insufficient. If the numbers are monetary values, then typically any uncorrected error is unacceptable; whereas if the numbers are drug doses, then a relative error greater than some value (say 5%) would be unacceptable; other domains will have different criteria. A domain may additionally have *notifiable errors*, such as the number being out by a factor of ten or more.

User error rates during interactive number entry tests in the laboratory may be as low as 1–2% (Oladimeji *et al*, 2011), which hinders obtaining significant statistical results. If empirical tests compare different styles of interactive number entry (e.g., independently rotating digit dials versus dials that carry) the incident rates at the boundary cases will be even lower, perhaps 0.01% to 0.1%.

2.4 The Richness of Interactive Numbers…

Some interactive number systems are dedicated to particular domains (e.g., cash machines), and others are fairly generic (handheld calculators). Numbers also appear in many places where they are really names (e.g., telephone numbers, PIN numbers), for scales (e.g., the Richter scale), in historical contexts like "2/–" ("two shillings" or £0.1), and in contexts with "check digits," as in International Standard Book Numbers (ISBNs), etc. Mobile phones handle telephone numbers, and therefore have to handle emergency numbers (999, 192, 911, etc), and — an interactive number issue — decide how to allow users to dial emergency numbers when a phone is otherwise disabled yet not dial it [often] by accident (e.g., from random knocks to buttons in a handbag). How numbers should be written is a complex subject too, itself not without controversy. Thus large numbers have thousands separators — unless they are page numbers or dates. And so on.

Most numbers are entered assuming that a dimensioned value is being entered. SI conventions on the use of k, M, n multipliers are clear, though there is conflict with ISMP guidelines (e.g., because a handwritten μ

is confusable with *m*; and the solidus "/" may be misread as a 1; ISMP argues litres should be abbreviated L not l which would make 5 L look like 51; etc). Although most medical infusion pumps expect numbers in mL per hour, a nurse will have calculated the rate on a general purpose handheld calculator unaware of number dimensions, and hence unable to detect interactive dimension errors (H.Thimbleby, 2008).

A complete and consistent coverage of design issues for all domains is clearly impossible — calculators may be used for innovative interactive number uses beyond any reasonable design brief (e.g., many calculators can be set up so that pressing "+" makes them count events; this is a different sort of number entry approach than the usual 0–9 number entry envisaged by designers). Equally, a paper such as this is unable to cover all potentially relevant design issues: interactive numbers touch on many areas, and a complete review of issues is impossible — particularly in a short paper. However, the following topics deserve mention: number notations (e.g., scientific, SI, financial, …); sets of preferred numbers that are within a specified relative error (e.g., the E6 preferred numbers, 10, 15, 22, 33, 47, 68 are within 20% of each other); and so forth.

Static number representations and fonts have been widely studied. Surprisingly, hard-to-read displays, such as seven-segment number displays, are widely used in safety critical applications, such as aircraft altimeters, personal radiation meters, etc (H.Thimbleby, 2011).

Although the SI/ISO 31-0/ISO/IEC 80000 standards specify that numbers "can be made more readable by separating them into groups, preferably groups of three, separated by a small space" few interactive systems do this. Some systems use the comma, even though this is specifically prohibited because of confusion with the decimal point (which is some countries is a comma). In other words, although there are standards for number notations, they are widely and regularly flouted. In the interactive case, the grouping changes as the user keys more digits: 1 234 becomes 12 345 when the user keys "5" — the space seems to move to the right, and its final position cannot be displayed correctly until the user has finished (or reached the decimal point).

Numerical Notation by Chrisomalis (2010) covers the history of static number notations in more detail than we need for this paper.

8. INTERACTIVE NUMBERS AS A GRAND CHALLENGE

Grand Challenges are big, worthwhile topics with recognized value to achieve. Typical Grand Challenges are to manage the nitrogen cycle, provide worldwide access to clean water, reverse engineer the brain, and more specialized but nonetheless strategic challenges like designing a verifiable programming language compiler. Well known Grand Challenges that have been achieved are to get man on the moon, to decode the human genome, and to eradicate smallpox. Grand Challenges unite a research community because they are worthwhile, and with concerted effort, achievable.

Interactive numbers is a "small" topic that in principle seems might, with a research programme, be completely specified, thoroughly evaluated, and moved into an international standard. Considerable benefits would ensue: every interactive number system would become easier and more reliable to use, and transfer errors would be reduced — presumably systems that looked the same would be the same. Yet as this paper showed, interactive numbers are deceptively simple, and a rigorous, empirically justified standard is a long way off. However, given the importance of numbers, particularly in safety- and mission-critical systems, a standard (no doubt with many subsections for various specialized domains) remains as a very worthwhile grand challenge.

Often Grand Challenges have spin offs. Thus the technologies that got man on the moon are much more widely used, for instance in launching satellites and in astronomy. Interactive numbers are a special case of HCI; they show that minute attention to detail has dividends. In researching interactive numbers, we will discover new fundamental ideas about interaction — numbers are just one form of state, and *all* interactive systems are about manipulating state.

The benefit of a Grand Challenge is focus on an achievable, worthwhile goal. Interactive numbers are ideal: unlike almost any other area of human computer interaction, they are focused; they have clear problems; the problems are soluble; and progress will benefit everyone. In contrast, the more "exciting" areas of research are often exciting precisely because they are so distracting and hard to pin down!

9. CONCLUSIONS

Since numbers are used everywhere in society, from gaming to international finance, from healthcare to nuclear power plants, from aircraft to climate change, from alarm clocks to GPS coordinates. Numbers are used in computer applications *known* to be buggy (spreadsheets, medical devices, etc), even small advances in the science and engineering of interactive numbers can be expected to make a major impact for good. It is time to take interactive numbers seriously. Moreover, our initial results, briefly discussed above, show that progress is certainly possible.

There is considerable scope to improve both Arabic number entry and incremental number entry; more software engineering research is needed; more empirical research is needed; and more application by programmers and developers to what is, in fact, a non-trivial but focused problem. In short HCI should take interactive numbers seriously. Amateurish *ad hoc* solutions — as almost all currently are, even on safety-critical systems, should no longer be acceptable, and we should be able to rigorously explain why they are unacceptable.

If it is to be called incremental research, then our field, HCI, indeed needs more incremental research and not just exciting novel research; every improvement in interactive numbers will benefit everyone (even a tiny increment multiplied up by millions is a big improvement), and benefitting people is why we pursue any research program in human-computer interaction anyway. Our goal, indeed a Grand Challenge, is a rigorous standard for interactive numbers, completely formally specified (with off the shelf definitive implementations), empirically justified in its effectiveness, and with guidance on design tradeoffs (e.g., for optimizing speed, accuracy, error rate, and manufacturing costs, etc).

ACKNOWLEDGEMENTS

Paul Cairns, Abigail Cauchi, Andy Gimblett, Andreas Holzinger, Paolo Masci, Patrick Oladimeji and Will Thimbleby have all made essential contributions to the research summarised here. The research has been funded by EPSRC under grants EP/G059063/1 and EP/F020031/1.

REFERENCES

S. Chrisomalis (2010) *Numerical Notation*, Cambridge University Press.

ISMP, Institute for Safe Medication Practices (2006) *List of error- prone abbreviations, symbols and dose designations*. See www.ismp.org/tools/abbreviations

P. Oladimeji, A. Cox, H. Thimbleby (2011), "Number Entry Interfaces and their Effects on Errors and Number Perception," submitted.

H. Thimbleby (2000) "Calculators are Needlessly Bad," *International Journal of Human-Computer Studies*, 52(6): 1031–1069.

H. Thimbleby (September+October, 2008) "Ignorance of interaction programming is killing people," *ACM Interactions*, pp52–57.

H. Thimbleby (2010) "Think! Interactive Systems Need Safety Locks," *Journal of Computing and Information Technology*, 18(4):349–360.

H. Thimbleby & P.Cairns (2010) "Reducing number entry errors: solving a widespread, serious problem," *Journal of the Royal Society Interface*, 7(51), 1429–1439.

H. Thimbleby (2011) "don't use 7 segment displays," *Proceedings BCS HCI 2011*, in press.

W. Thimbleby (2010) *Drawing from Calculators*, PhD Thesis, Swansea University.

W. Thimbleby & H. Thimbleby (2008) "Mathematical Mathematical User Interfaces," in Proceedings Engineering Interactive Systems 2007/DSVIS 2007, *Lecture Notes in Computer Science*, 4940:519–535, Springer Verlag.