Ignorance of Interaction Programming Is Killing People

Harold Thimbleby | Future Interaction Technology Lab (FIT Lab), Swansea University | harold@thimbleby.net

Almost a century ago, the April 9, 1929, issue of the *International Herald Tribune* reported the death of three young brothers. All of them had been given a dose of thallium acetate 10 times what was intended, because of a decimal-point error.

Decimal-point errors occur regularly. For instance, on October 7, 1998, *The New York Times* reported the death of a 10-month-old from a decimal-point error. In May 2001, the Canadian Institute of Safe Medication Practice (ISMP) reported two deaths caused by decimal-point errors: In two separate cases, .5 mg of morphine was misread as 5 mg. The ISMP report mentioned that decimal-point errors were among the first safety issues the Institute had dealt with when it was founded almost 10 years ago. We are still risking such errors every day.

What Is Interaction Programming?

The title of this article mentions "interaction programming," a term I've introduced to distinguish the programming aspects of interaction from the more-often-emphasized human aspects [1]. Human factors and design, together with user-centered processes, are often taken to be all there is to interaction design, but the hidden partner is the details of how things work when they are used. As they say, the devil is in the details, and this is a matter of programming.

This article shows that the programming matters a great deal. A crucial point is that good interaction programming has to be engineered into a device's design by good programmers; it cannot be established by inspection after it is working.

A very simple example is the Cardinal Health Alaris GP infusion pump, a new model introduced in 2006. I have one, and its firmware failed, so the manufacturers replaced it. The replacement has a new user interface quite different from the old, but of course the physical ergonomics are identical. Although there are some obvious differences between the old and new user interfaces, the exact differences (all of which affect users) cannot be established by inspecting the device. Interactive programs are too complex for unaided human comprehension; instead, good user-interface-design requirements must be engineered into programs by rigorous, formal processes.

User-centered methods and processes are essential, and, quite rightly, are emphasized by the usability community, but they are not sufficient to assure safe interaction. For too long user-centered methods and programming have lived in different worlds—programmers discount human factors, and usability people discount programming. Users do not understand design; neither they nor interface designers can articulate the full intricacies of computerized problems. Yet programmers think their own programs, so intuitive, so easy to demonstrate, need no hard work to become usable. But all of us need to work together.

These ideas will become clearer by exploring interactive medical devices.

Details Matter

National agencies make detailed recommendations on how to write drug dosages: Always write fractions like 0.2 mg with a leading zero, never have a trailing zero, as in 1.0 mg (it might be read as 10, not 1), and so on. There are rules for not confusing micrograms and milligrams (μg badly written could be confused with mg, causing a factor of 1,000 error). Write milliliters as mL, not as ml, which might be confused for m1. Write slash in full as "per," that is, write "mL per hr," not mL/hr, so the / won't be confused. Don't use unnecessary decimal precision: 20.4 mg might be read as 204 mg. And so on. Unfortunately, little if any of this basic life-saving advice seems to have been picked up by manufacturers of interactive medical devices.

A typical interactive medical device allows users to enter numbers in almost any format, with or without decimals, misleading zeros, and all without any warnings whatsoever. A human factors study of one pump [2] found that three out of five registered nurses were "partially or completely confused" over using the decimal-point key (the user interface doubles up the decimal point with an arrow key that is used for menu selection). A paper on another pump noted that its user manual says that it works like a calculator when in fact it does not [3]: if a user enters 0.0.5 on the pump, it is taken as 0.5, whereas on a typical calculator, the same button presses would be taken as 0.05, a very different value. This difference could clearly lead to serious problems. Neither pump nor calculator reports any error when more than one decimal point is entered.

The Alaris pump mentioned in the introduction has no numeric keys, so it cannot suffer from decimal-point errors as such. Instead, it has four buttons to increase and decrease the current number by 1 or by 10. Unfortunately, the close proximity of the buttons might mean a nurse presses the 10 instead of the 1. Here, what is intended as a user-interface accelerator has created a hazard analogous to the decimal-point problems of conventional numeric keypad user interfaces. Overall, this may be better or worse—one would have to do experiments to find out. A potentially worse problem is that different approaches (increment/decrement versus numeric keypad) create their own problems: most hospitals have many types of device, and correct operation of one may be deadly if transferred to another. In short, we need *very detailed* standards for user interfaces so that there are no unnecessary proliferations of interaction styles.

Decimal-point errors are one of the simplest drug-calculation errors to understand, one of the longest consistently recognized problems in the area, and, arguably, the easiest to do something about. Yet nothing seems to be happening. Well, one might then ask, is it a significant problem?

Medical errors in hospitals in a given year cause about as many deaths as AIDS, car accidents, and breast cancer combined [4]. Clinicians accept as routine using workarounds, such as switching a device off and on to recover from errors—often losing data (e.g., drug dose to date) in doing so. Indeed, many near misses are not reported because they do not lead to adverse clinical incidents. Often hospital procedures or training are blamed for not accommodating device design, rather than the other way around. If a device "operates as designed," it is often assumed to be designed correctly, even if (to more perceptive eyes) the incident is a symptom of bad design, a "system-induced user error."

A Fatal Overdose

In 2006 a patient received a fatal overdose of fluorouracil, a chemotherapy drug. Here's a summary of how it happened, based on the investigation [2]. The nurse went to the hospital pharmacy with the drug order and returned with a labeled bag of diluted fluorouracil and a printout of the dose details. The nurse's task was then to calculate how to program an infusion pump to deliver the drug at the appropriate rate. The relevant numbers and units are 5,250 mg of fluorouracil diluted to 45.57 mg per mL, to be delivered over four days. This is not an easy problem for anybody to work out, even without the many simultaneous jobs that nurses have to do as well.

The nurse had to calculate the rate to be delivered, 5,250/45.57 mL over 24×4 hours; he or she should have done this calculation:

$$\frac{5{,}250}{45.57} \Big/ (4 \times 24)$$

The nurse attempted the calculation using a calculator, and a second nurse double-checked the work as a routine precaution. It's a simple calculation, as things go—for instance, a calculation for a dose of gentamicin (an antibiotic) is based on patient weight, gender, and height, and involves powers, as well as many constants and conditionals.

It is easy to take the design of calculators for granted, but we already know that calculators ignore many errors. So let's look more closely at how one was used (though the report does not give details, presumably because it assumes calculators "just work").

Calculators Are Mad, Bad, and Dangerous

The nurse would have pressed a sequence of buttons to perform the calculation. For example, the keystrokes AC AC 5250 ÷ 45.57 ÷ (4 × 24) = will obtain the correct result, 1.2. However, it is likely that the nurse did not have a calculator with brackets, and instead had to do AC AC 5250 ÷ 45.57 ÷ 4 ÷ 24 = . What nurse knows that repeated division is equivalent to dividing by a product? Far more likely, then, the nurse would have calculated 4×24 on paper or used the calculator to store the result in the calculator's memory. He or she would then need to do AC AC 5250 ÷ 45.57 ÷ MRC = to get the answer.

How can one work out 4×24 and store it in the memory? A basic calculator has a memory, but many calculators do not have a "store in memory" key; instead, they have an "add to memory" key, M+. To store a number to memory, then, the memory must already be zero, otherwise the number stored will be wrong. If the nurse starts to calculate 4×24 before zeroing the memory, it is almost impossible to store the result correctly.

To get the drug calculation right, the nurse must do the following: AC AC MRC MRC 4 × 24 M+ 5250 ÷ 45.57 ÷ MRC =. The buttons AC and MRC must both be pressed at least twice at the start, otherwise the nurse risks the wrong answer being calculated.

In computer science terms, what the nurse has just done is called "compiling"; the nurse has converted, that is, compiled, a calculation into a sequence of "machine code operations"—button presses—to do it. To compile correctly, which is crucial to get the right answer, the semantics of the target machine (the calculator) must be defined; but we know many calculators are very different (and, worse, mathematically wrong) despite even looking alike [5]. Clearly, compiling is a difficult task for any user, and indeed one can imagine it is especially difficult for people trained as nurses rather than as computer scientists. We do not know from the incident investigation whether the nurses got the compiling wrong, were using the wrong numbers, or "simply" missed a step in their calculation. But requiring nurses to do such a complex operation as compiling for such badly specified devices as handheld calculators is manifestly risky. The cognitive load on the user will not have helped their vigilance and ability to detect errors. Whatever the causes, we do know they made an unfortunate calculation error they did not detect.

Although compiling on a calculator is very complex, it is still generally easier than doing the calculation with pencil and paper. One of the main reasons compiling is so complex is that calculators are designed to do any calculation—they are more powerful than any nurse or doctor needs. If the nurse makes a mistake, perhaps pressing – instead of ÷, no calculator will complain; it has no idea what calculation the nurse is trying to do. It will just provide the wrong answer.

If the design of calculators is inappropriate for medical calculations, it is even more remarkable that the infusion pump did not help, as it—unlike a calculator—was specialized to medical problems. Its design should have been based on a task analysis and potential user errors. An infusion pump contains microprocessors, and one could easily be designed to take concentration, duration, and so on from the nurse and do the sum itself. Some are, of course, but not this one (the ones that do, so-called "smart pumps," cost much more despite differing only in their programming).

Interactive Medical Devices Are Bad Too

In the 2006 fatality, both nurses failed to divide by 24 hours per day, so they agreed the dose rate was 28.8 mL per hour instead of the correct 1.2 mL per hour. The pump could have told the nurse that at that rate the drug supply (which the pump knows) would take about four hours to be

used. This would not have been the four days the nurse expected. Instead, the infusion pump merely asked the nurse to confirm what they had entered. They had entered 28.8 in error, so the pump asked if they had meant to enter 28.8. Unfortunately, having made the error, that *was* what they thought they wanted to enter.

Since the pump was in use exclusively on a chemotherapy ward, it could have checked that dose rates were appropriate for standard drugs; more than 30 grams per day for fluorouracil should have raised warnings (1 gram a day is a high adult dose for fluorouracil). Unfortunately the pump provided no such checking.

The infusion pump was an Abbott AIM Plus. In the mode where the nurse should enter mL per hour, the display option is "mL" without the "per hour," which is incorrect (see Figure 1). Moreover, the HELP button provides information on only two of the three options and does not give help for the incorrectly labeled option!

The pharmacy computer printed the label on the fluorouracil bag, which the nurse used to get the numbers for the calculation. The label confusingly included many numbers, 1.2mL/hr, 28.8mL/24h, 1312.5mg/24h … 15 numbers in all, not counting the date and patient-identification details. The numbers on the label break many recommendations: 1.2mL/hr rather than the correct 1.2 mL per hr (the space before mL is required to help avoid the m being misread as 00), and showing a number pointlessly to five significant figures, and so on. Worse, in my opinion, the numbers were not organized in any way that related to the pump's requirements. The bag label appears not to have been designed to help the nurse who has to use it.

Both nurses incorrectly calculated 28.8, yet this incorrect number had also been printed on the label, which would have provided confirmation bias for the nurses and distracted their attention from relevant detail; indeed, the cognitive load of compiling a complex calculation would have reduced their error-detecting vigilance in general.

The ISMP report [2] commissioned a small human factors study of the Abbott pump: it identified numerous problems. Why aren't devices made consistent with best clinical practice, so that operator training becomes simpler, rather than the other way around? Why does the report [2] say in its recommendation 10A that nurses should be trained that "mL" on an infusion pump means "mL per hour"? Why does recommendation 10B ask purchasers (hospitals) to do human factors studies of pumps? The same answer to both questions is that for the time being manufacturers—and national regulatory processes—can't be relied on, and hospitals therefore have to train nurses to cope with bad design. That also means that when things go wrong, as they do, that the nurses or the training has failed: it's then a very short step to blame the nurses or their management for the consequences.

Alternatives Are Possible

I spent a day programming an Apple iPhone to explore ways of improving things. With my prototype you can hold in your hands a working system that avoids some of the problems described above. It can be downloaded from harold.thimbleby.net/health.

The iPhone stimulates many ideas: for example, it has a camera and could photograph the drug barcode and check that it was what was expected; it could require a second nurse to check the calculation; and so on. On the other hand, the iPhone is a new approach, so it might not work as well as expected without further development.

Using a conventional calculator, a dose calculation would report very few errors—perhaps an accidental division by zero. It would just display "E" (and a wrong number) when there is an error. Unlike conventional calculators, the iPhone provides a clear explanation and, importantly, no number that could be misinterpreted is displayed. Using the iPhone for the calculation above,

potentially 52 errors can be detected, and some are detected during incomplete steps, such as when the nurse is entering 45.57 but has not yet entered the decimal portion of 57.

On the iPhone, a nurse cannot easily do the wrong calculation, whereas on a conventional calculator, it is easy to hit + instead of ÷ and never notice. The iPhone has no operators, and therefore the user cannot employ the wrong ones. The iPhone also uses correct units (mg and so on) and checks that they are used consistently; a conventional calculator has no idea about units and cannot help the user avoid errors related to them (say, mixing up milligrams and micrograms).

To make numbers easier to read, the iPhone shows a clear decimal point with the decimal part smaller, as in 45•57 (see Figures 3 and 4), and large numbers are shown with commas (another recommendation the fluorouracil bag ignored), as this reduces confusion between numbers like 100000 and 1000000. (I don't currently require users to enter commas; it would be an interesting study to see if their use would reduce errors.)

If the calculator communicated with the electronic patient record, all numbers could be automatically provided—hence, correct—or they could be confirmed by the nurse rather than entered manually. This would avoid transcription errors. If protocol requires nurses to be responsible for calculations, the iPhone could show a checksum for the correct answer: the bag label would say something like "if you don't get X5Q [the checksum], you've got the wrong answer."

Conclusions

It is astonishing that a life-threatening problem that has been recognized for a century has had such little impact on interaction design. Why are basic errors ignored by interactive medical devices? As the iPhone showed, better interaction programming is easy to explore.

We could save many lives if we made people aware that poor interaction programming is a significant factor in medication incidents. Lawyers who represent patients and clinicians need to know more. We already have many good recommendations to improve design [6]; this article has argued that these recommendations should also be applied to the details of interactive design.

Investigatory bodies, analyzing incidents, must include people trained in HCI. This has already started, and the role of ergonomics and human factors is increasing, but expertise in interaction programming is essential too. It should be normal for manufacturers to employ programmers with appropriate postdoctoral specialist qualifications—just as pharmaceutical companies do.

Regulatory bodies, too, should get the skills to stop problematic designs from being approved. Interaction is complex: program specifications and source code must be checked using formal tools, otherwise inconsistencies and other problems will not be detected (this is a fundamental theorem of computer science). This article talked about "simple" problems with numbers, but no usability study can check that *all* numbers, both well-formed and erroneous, are handled correctly. (And number entry isn't the only design feature that needs checking.) In short: a usability study can *help* check that a design is appropriate for users and their tasks, but the entire design *must* also be checked by formal methods. Quality assurance has to be done primarily by using rigorous manufacturing processes, not later by regulatory bodies or by hospitals—or by users finding the bugs.

There are many more ideas; changing culture is never easy, and it will require many approaches. Lives depend on us. I've started to put some resources together at http://harold.thimbleby.net/health

{**Sidenotes**}

[1] Thimbleby, H. *Press On: Principles of Interaction Programming*. Cambridge: MIT Press, 2007.

[2] Canada. Institute for Safe Medication Practices. *Fluorouracil Incident Root Cause Analysis*. www.ismp-canada.org. 2007.

[3] Thimbleby, H. "Interaction Walkthrough: Evaluation of Safety Critical Interactive Systems," DSVIS 2006, The XIII International Workshop on Design, Specification and Verification of Interactive Systems, Springer Lecture Notes in Computer Science, edited by G. Doherty and A. Blandford, 4323:52–66, 2007.

[4] Kohn, L. T., J. M. Corrigan, and M. S. Donaldson eds., *To Err is Human*, National Academy of Sciences, 2000.

[5] Thimbleby, H. "Calculators are Needlessly Bad," *International Journal of Human-Computer Studies* 52, no. 6 (2000):1031–1069.

[6] United Kingdom Department of Health. *Design for Patient Safety*, 2003.

About the Author

Harold Thimbleby wrote *Press On: Principles of Interaction Programming,* which won the American Publishers Association best book award in Computer and Information Sciences in 2007. *Press On* has more design recommendations for interactive devices, not just medical devices. Harold is a Royal Society-Leverhulme Trust Senior Research Fellow, and the work here was also supported by EPSRC Grant EP/F020031. See harold.thimbleby.net.

Figure 1. The infusion pump used in the fluorouracil incident. The pump is small and gives the patient full mobility during the treatment. Here, the nurse needs to enter mL per hour but has to use Option 3, which is apparently asking for mL! Note that the up arrow key doubles as the decimal point.

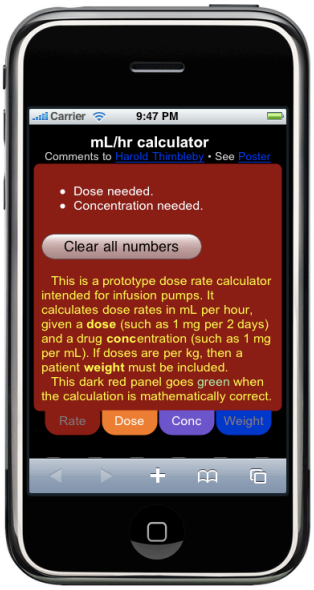

Figure 2. The prototype dose calculator running on the iPhone (it also works on desktop Web browsers). The opening screen is red and shows that a dose and drug concentration have not yet been provided. The tabs at the bottom of the screen allow the user to choose which numbers to enter; they allow users to enter numbers in any order, unlike an ordinary calculator, where changing order would create errors.

Figure 3. Entering the drug concentration, using the keypad. The screen scrolls up, and the numeric keyboard appears when a number field is tapped. The "Rate" tab is red, indicating outstanding errors, one the errors at this point is that the user has not finished entering the concentration.

Figure 4. Once all numbers are entered correctly, the main screen goes green and summarizes the dose details. It also confirms how long standard sizes of drug will last and what the daily dose is.