Software design issues in medical device user interfaces

Paolo Masci
Queen Mary University of London
(paolo.masci@eeecs.qmul.ac.uk)

NordiCHI2014, October 26th, 2014
Focus of this talk

Software design issues identified in 20+ medical devices in use in hospitals across the US and EU

A recorded video of the demonstration is available on my academic website http://www.eecs.qmul.ac.uk/~masci/trainingMaterial/
CHI+MED research project
www.chi-med.ac.uk

Long-term aim: to transform the design and use of medical devices so as to help clinicians avoid and recover from human error

Combining a variety of approaches
– Mathematical analysis of device designs
– Contextual studies in hospitals
– Lab-based experiments
– Understanding manufacturer’s context
– Public engagement
Impact of our research

• Creating the basis for a new international standard for medical software
  – will enable a more precise and uniform evaluation of usability and safety of medical devices

• Driving the research agenda of regulators
  – new methods and tools for pre-market review and post-market forensic investigations

• Informing procurement decisions
  – hospitals are using our techniques to improve purchasing of new medical devices
The regulatory challenge

- Nearly 2,000,000 medical devices recalled in the last decade in the US alone

- The majority of recalls were due to software defects and user interface issues

- Problems affect all manufacturers
The example of drug infusion pumps

Medical devices used to deliver a medication intravenously at controlled rates and precise volumes

Infusion program is set by clinicians by interacting with the pump user interface
The nature of the demonstrated issues

• **The cause:** software defects

• **What happens:** the device provides inappropriate feedback, ignores key presses, enters new modes without the user’s awareness

• **Potential consequences:** serious use errors (e.g., *missing decimal point errors*) that can lead to patient harm
Software defects

• Coding errors
  – e.g., division by zero, memory allocation errors, …

• Logic errors
  – Unexpected modes
  – Incorrect feedback
  – Ignored key presses
  – …

Several tools and techniques exist for identifying coding errors

Logic errors are harder to identify, as they are inherited from requirements

We will focus on this type of design errors
Our Analysis Method

- Infusion pump
- Usability & Safety Requirements
- Test cases

Mathematical model of user interface software

Automated analysis using formal methods technologies

Logic errors in user interface software
The benefits of our method

• We can verify safety/usability requirements for all software executions and for all user inputs

• We can discover anomalies in the software behaviour and generate input key sequences to challenge the real device in seconds

• We can identify verified solutions that can fix identified anomalies
Demonstration of identified issues
Arrow up and down
Scroll through menus, change setting of numbers from 0-9, answer Yes/No questions.

Arrow left and right
Select data from a scale and switch between digits when numbers are entered. Open a function while pump is running or stopped with the left arrow key.

Yellow LED: Pre-alarm, reminder alarm
Green / Red LED: Infusion occurring / device alarm, operating alarm
Blue LED: Currently connected to SpaceControl

Press to reset single values to zero and switch back to the previous screen/menu level.

Press to open the pump door.

Press to initiate bolus.
Press to turn pump on/off.

Press to link the pump to SpaceControl and to assign a barcode after scanning.

Press to Start/Stop infusion.

Open certain functions and press to confirm values/setting/alarms.
Data entry: instructions presented in the user manual

In the Main Menu, open the rate with ◀ and set it with ⏯.

(from the user manual)
In the Main Menu, open the rate with 🔱 and set it with 🔱.

81200 91200 99999 ...
In the Main Menu, open the rate with ⬆️ and set it with ⬇️.

81200 ➤ up ➤ 91200 ➤ up ➤ 99999 ➤ left ➤ 99999
In the Main Menu, open the rate with up and set it with left.

81200 up 91200 up 99999 left 99999 down 91200
In the Main Menu, open the rate with ← and set it with →.
In the Main Menu, open the rate with \( \uparrow \) and set it with \( \uparrow \).
In the Main Menu, open the rate with \( \downarrow \) and set it with \( \uparrow \).
An accurate specification of the device behaviour

Obtained in our labs by reverse-engineering the real device

Same number entry, different behaviour

Pump 1

Pump 2
Volume: 950ml

Input Key Sequence
1. Left
2. Up 5 Times
3. Left
4. Down

Result from Pump 1

Result from Pump 2
Ignored values
Values are ignored without any warning when input key sequences are not terminated with “OK”
Ventilators have the same issue
Similar devices have opposite behaviours

• This pump automatically accepted values even if the input key sequence is not terminated with “enter”
Decimal point
The decimal point is erroneously ignored

The key sequence

```
1 0 0 . 1
```

is registered as 1001
The decimal point is erroneously ignored

The key sequence

1 0 0 . 1

is registered as 1001 (the value is fortunately rejected in this case because the pump configuration limits the rate value to 999 mL per hr)
Ill-formed values
Fractional numbers without leading zero
(they might be easily misread as integer values)
Integer values with leading zeros
(they might be easily misread as fractional values)
Timeouts
If the user edits a value and pauses for a few seconds before confirming the new value…
...then the device erroneously discards the new value without any warning.
We observed and reproduced the same problem in different types of devices

Patient monitor

Ventilators
Soft buttons
Guardrails Drug Setup morphine

INFUSION MODES

PCA Dose only
Continuous Infusion
PCA Dose + Continuous

> Select an Option
Cannot proceed due to incorrect concentration or dosing parameters.

Remove syringe, verify concentration and reprogram.

Press REPROGRAM
Accumulated air: Enabled
Air-in-line settings: 75 μL
Auto-restart attempts: 0
KVO rate adjust: 1 mL/h
Max rate: 999 mL/h

Select an Option or EXIT
System Configuration - Pump

Maximum rate adjustment

Maximum rate:
999 mL/h
Phacoemulsifiers have similar problems
Increase/Decrease values

Increase/Decrease values
Select an Option

Increase/Decrease values
DISABLED

Scroll options
Wrong information
Guardrails Drug Setup
BD Plastipak 50/60 mL

CONT DOSE: 1 mg/mL
MAX LIMIT: >>>>>
[Conc]: 1 mg/mL

Select Max Limit Options

CHANGE MODE
PRIMARY INFUSION

- RATE: 0.02 mL/h
- VTBI: 2 mL
- Time Left: < 1 minute

-0.1 is less than MIN RATE
0.1
Additional user interface issues
Viewing angle
Similar problems can be reproduced in any device equipped with seven-segments displays

- Infusion pumps
- Infant warmers
- Phacoemulsifiers
- Patient monitors
Understanding the relation between design error, use error and human error
System Failure

- Traditionally, system failure means something is broken

- However…complex systems may “fail” without anything “breaking”

- Often users are blamed when things go wrong
Systematic Use Error

• Use errors are errors committed when operating a system

• People act according to their knowledge of tasks and device

• They have limited resources such as working memory

• A consequence is that user interface designs can lead to systematic use error if those limits are not considered
Example: Post Completion Errors

- Errors related to “tidying tasks”
- People may forget to terminate the interaction because their goal is achieved
- **Example:** the user interface suggests value entered, so the user may forget to confirm the value
Manifestation vs. Cause

- What happened (the erroneous action)
- Why it happened

• Example
  - Manifestation: user enters wrong infusion rate
  - Cause: keying error, hardware defect, badly designed user interface, …
The manifestation

• Broad classes of erroneous actions
  – Too early / too late, Too long / too short
  – Too little / too much / Movement too far / too short
  – Speed of action too fast / too slow
  – Wrong direction / wrong kind of movement
  – On wrong object
  – Actions omitted / earlier actions carried out again
  – last action repeated
  – Order of adjacent action reversed
  – Irrelevant action carried out
The causes

- Knowing the manifestation does not help greatly in re-designing the system

- We need to determine the underlying causes

- In complex situations (such as medical incidents) there are often many causes that lead to disaster when they happen together.
Broad classes of causes of use error

- Person-related causes
  - e.g., erroneous interpretation, distress

- Technology related causes
  - e.g., user interface problems, equipment failure

- Organisation related causes
  - e.g., training, working conditions, managerial
Suggested reading:
“The phenotype of erroneous actions”
by Erik Hollnagel
Summary

• Systematic use errors are often a manifestation of design issues/flaws

• The vast majority of design issues/flaws are due to logic errors caused by wrong or incomplete requirements
Interaction design principles

Paolo Masci
Queen Mary University of London
(paolo.masci@eeecs.qmul.ac.uk)
Example guideline from Human Factors Engineering Standards

Users can become distracted by other tasks, or be interrupted during device use. Therefore, designers should not depend on users to remember information needed to perform a task. It is far better to present to users the crucial information they need to perform the task correctly."

--- HE75:2009, Chapter 4, Section 6.4
A core set of principles
Visibility

• Ability to make relevant information perceptible.

• If this principle is not implemented effectively, users could fail to understand what needs to be done.
The user is asked to select yes/no, but the question is not visible
Saliency

• Ability to make relevant information prominent and easy to notice.

• If this principle is not implemented effectively, users could miss important information presented by the system.
The attention of the user is driven to the wrong places.
Feedback

• Ability to provide sufficient information about what has been achieved.

• If this principle is not implemented effectively, users could fail to understand what has been done or what has been achieved
Feedback suggests value entered
Predictability

• Ability to anticipate the next goals and the effect of actions.

• If this principle is not implemented effectively, it could be hard for users to foresee the consequences of interactions with the system.
The effect of button presses cannot be predicted from the current observable state
Affordance

• Ability to link system design features with user's knowledge and culture.

• If this principle is not implemented effectively, it could be hard for users to know or learn how to operate the system.
Affordance suggests soft buttons can be used to select channel A, B or C
Consistency

• Ability to support similar tasks using similar operations and similar user interface elements.

• If this principle is not implemented effectively, users could achieve wrong goals or perform wrong actions without realizing it, because they think the goals/actions are appropriate when in fact they are not.
Soft buttons on either side of the screen should behave similarly.

<table>
<thead>
<tr>
<th>Setting</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accumulated air</td>
<td>Enabled</td>
</tr>
<tr>
<td>Air-in-line settings</td>
<td>75 micro-liter</td>
</tr>
<tr>
<td>Auto-restart attempts</td>
<td>0</td>
</tr>
<tr>
<td>KVO rate adjust</td>
<td>1 mL/h</td>
</tr>
<tr>
<td>Max rate</td>
<td>999 mL/h</td>
</tr>
</tbody>
</table>

- Select an Option or EXIT
Forgiveness

- Amount of effort needed to recover from wrong goals or actions.

- If this principle is not implemented effectively, errors and mistakes identified by the user require unreasonable amount of time or effort to be resolved.
After a timeout, the device discards the entered value without warning the user.
A core set of principles

- **Visibility**: Ability to make relevant information perceptible.
- **Saliency**: Ability to make relevant information prominent and easy to notice.
- **Feedback**: Ability to provide sufficient information about what has been achieved.
- **Predictability**: Ability to anticipate the next goals and the effect of actions.
- **Affordance**: Ability to link system design features with user's knowledge and culture.
- **Consistency**: Ability to support similar tasks using similar operations and similar user interface elements.
- **Forgiveness**: Amount of effort needed to recover from wrong goals or actions.
Designing safer medical devices

Paolo Masci
Queen Mary University of London
(paolo.masci@eeecs.qmul.ac.uk)

NordiCHI2014, October 26th, 2014
The regulatory challenge

• The scale of evidence provided in premarket clearance submissions makes it hard to spot latent problems
  – Important details might be omitted or missed
  – Usability aspects are currently under-represented but a crucial factor in safety

• Can we make the argument more precise and succinct at the same time?
  – Rigorous development processes
  – Trusted tool-chains
Development methods promoted by the US Food and Drug Administration (FDA)

- Hazard Analysis
- Requirements
- Model
- Verification
- Verified model
- Code generation
- Verified code
Illustrative example developed by FDA engineers
Illustrative example developed by FDA engineers

1. Usage models encapsulate the common characteristics and behaviour for broad classes of devices (e.g., insulin pumps, PCA pumps, medical ventilators, etc..)
Illustrative example developed by FDA engineers

2. The reference model is checked against given safety requirements
Illustrative example developed by FDA engineers

3. Test cases are generated to validate the software implementation executed on the real device.
Usage (reference) models for infusion pumps

- Generic Infusion Pump
  - External
    - therapeutic
      - insulin
      - chemotherapy
      - opthalmic
    - analytical sampling
    - anesthesia
    - enteral
    - PCA
    - epidural
  - analgesic
    - PCA
    - epidural
  - implanted
    - therapeutic
      - insulin
      - chemotherapy
    - analgesic
Usage (reference) models for infusion pumps
The FDA has developed a usage model for the controller of PCA pumps (model available at http://rtg.cis.upenn.edu/gip.php3)

Model-based development

- Model of the software
- Verification against safety requirements
- Code generation

Figure 1: The GPCA Model System Architecture
Within CHI+MED, we developed a usage model for the user interface of PCA pumps.
Model-based development of the Generic PCA pump user interface in PVS

“Model-based development of the Generic PCA infusion pump user interface prototype in PVS”,
P Masci, A. Ayoub, P. Curzon, I. Lee, O. Sokolsky, H. Thimbleby
in SAFECOMP2013, , Intl. Conference on Computer Safety, Reliability and Security

“Verification of interactive software for medical devices”
P Masci, A. Ayoub, P. Curzon, M.D. Harrison, I. Lee, H. Thimbleby
in EICS2013, ACM SIGCHI Symposium on Engineering Interactive Systems
A generic pump user interface (?)

Which user interface should be used as a reference model?
• Different user interfaces have different pros and cons

How to promote innovation with safety in mind?
• Manufacturers should be allowed to present novel user interfaces if they have evidence that they enable safer interaction
A generic set of safety requirements (?)

Examples of safety requirements for PCA pumps

SR1: Clearing of the pump settings shall require confirmation

SR2: The flow rate shall be programmable
A generic pump user interface

A generic reference model specified in terms of concepts illustrated in the usability/safety/security requirements
Safety and Usability Requirements

Expressed as logic statements between concepts

SR1: Clearing_settings $\Rightarrow$ Require_confirmation
A correct interpretation (?)

Clearing settings = …
  • *Press cancel when the pump is not infusing?*
  • *Select “clear settings” from the main menu?*
  • …

Require confirmation = …
  • *Display confirmation message and press Ok button?*
  • *Implicit after a timeout?*
  • …

There is no single correct interpretation
  • Correctness depends on intended use, workflows, use environment, user population, etc.
Step 1: provide an interpretation (and argue about its correctness)

Press cancel when the pump is not infusing
Step 2: verify that the agreed interpretation is satisfied by the device
We mechanised the approach within the PVS and IVY verification systems
Proof obligations

PVS automatically generates the formula that needs to be verified on the device model to demonstrate that the requirement is met

Example proof obligation

```
IMP_reference_model_th_R1_Axiom_TCC1: OBLIGATION
  FORALL (st, st0, st1: gpcau_i_state):
    (gpcau_i_init?(st) IMPLIES
      0 <= stgetDisplay AND stgetDisplay <= 99999
      AND 0 <= stvtbi AND stvtbi <= 99999)
    AND ((0 <= st0getDisplay AND st0setDisplay <= 99999 AND 0 <= st0vtbi
      AND st0vtbi <= 99999 AND st1 = click_up(st0) OR %...)
      IMPLIES 0 <= st1getDisplay AND st1setDisplay <= 99999
      AND 0 <= st1vtbi AND st1vtbi <= 99999);
```
Successful application of the approach with two commercial infusion pumps

Chevron-based layout

Navigation-based layout
Details of the approach can be found in our papers

“Model-based development of the Generic PCA infusion pump user interface prototype in PVS”,

“Verification of interactive software for medical devices”
On-going work with the FDA

- Development of specialised analysis tools
  - can be used by regulators to challenge premarket submissions and during forensic investigations
  - can be used by industry to improve device design early in the development process

- Development of training material
  - to inform procurement decisions
  - to warn clinicians about current issues

- Creating the basis for a new reference standard for medical user interface software
  - will enable a more precise and uniform evaluation of usability and safety of medical devices
Specialised analysis tools
(https://github.com/thefrogfather/pvsio-web)

Realistic layout

User actions \downarrow \quad \uparrow \quad User interface response

Mathematical model of software
Generic architectures

- Architecture independent of any hardware or features
- We are using it to anticipate software-related hazards
Automatic verification of design principles: the example of predictability

- A prediction model captures core concepts of the design principle
- The device model captures the device behaviour
- Any divergence between the execution of the two models is an indication of a potential issue

Prediction Model (captures the user’s knowledge)  
Device Model (captures the device behaviour)
Predictability principle in PVS

"The observable state of the device after performing an action must be identical to the state predicted by the user on the basis of observable information"

up_button_predicable: THEORY
FORALL (st: Device_State):
    observable_state(device_up(st))
    = user_up(observable_state(st))
Example analysis results

Predictability fails in 3 cases:
- Display is 99999
- Display is 1
- Display is 0.1

The root cause:
- Navigation keys change mode of operation depending on the history of past interactions (which cannot be inferred from info displayed by the device)
Example recommendations for users

Verified use strategies to avoid predictability issues with the analysed devices

Safe strategy for device B

- Always use left/right keys to move the cursor and clear the device memory: this will avoid unexpected behaviour of up/down buttons

Unsafe strategy for device B

- Remember the last $n$ observable device states
  (what is a safe value for $n$?)
Example guidelines for designers
Applicable to any device

Avoid silent mode changes
– Avoid modal behaviour – prefer to stop interaction and notify users rather than changing mode
– If a mode change is needed, provide sufficient feedback

Check the behaviour at the boundary cases when using memory in data entry systems
– The strategy for storing numbers could lead to inappropriate behaviours
Recent Publications with the FDA


3 - Formal Verification of Medical Device User Interfaces Using PVS. P. Masci, Y. Zhang, P. Jones, P. Curzon, H. Thimbleby In ETAPS/FASE2014, 17th Intl. Conf. on Fundamental Approaches to Software Engineering, Grenoble, France, April 2014


6 - A generic user interface architecture for analyzing use hazards in infusion pump software. P. Masci, Y. Zhang, P. Jones, H. Thimbleby, P. Curzon In Medical Cyber Physical Systems Workshop 2014 (MedCPS), Berlin, Germany, April 14, 2014