User Interface Software Errors in Medical Devices: Study of U.S. Recall Data

Yi Zhang, Paolo Masci, Paul Jones, and Harold Thimbleby

Abstract

The current work assessed U.S. medical device recalls during 2012–15, with the goal of understanding the impact and nature of user interface (UI) software errors in medical devices. Based on information from the Food and Drug Administration’s public and internal recall databases, 423 (~140/year) medical device recalls were identified as resulting from UI software errors, which accounted for nearly one-half of recalls caused by software errors during the same period. A total of 499 UI software errors were identified as the root causes of medical device recalls, and a detailed classification of those errors (into 20 categories) was established. This error classification can be used by device manufacturers, end users (e.g., healthcare providers), and regulatory authorities to raise awareness of the type and impact of UI software errors. The classification also provides stakeholders with an evidence-based challenge to assess and improve the quality of UI software in medical devices.

Mitigation of use errors is a top priority in the design of medical devices (e.g., infusion pumps, ventilators) and for health information technology systems (e.g., electronic health record systems). In many cases, use errors associated with medical devices are induced by, among other design defects, errors in the design or implementation of the user interface (UI). UI software, which governs the interaction between a device and its user, can increase the chance of use errors and in turn pose threats to patient safety if it does not appropriately manage design flaws and implementation mistakes.

This article presents a systematic and detailed analysis of three years (September 2012 to August 2015) of Food and Drug Administration (FDA) medical device recall data, with the goal of assessing the prevalence, categorization, and safety implications of UI software errors in medical devices. The analysis identified 423 medical device recalls that were caused, partially or entirely, by UI software errors, accounting for 5.44% of total recalls studied or 46.33% of recalls resulting from software errors. In total, 499 UI software errors were observed as the root causes of device recalls. Based on a generic UI software architecture, a hierarchy of UI software error categories was established to capture and classify the nature and impact of the observed errors.

These findings can provide device manufacturers with deeper insights into known UI software errors and help them better prioritize resources and efforts to reduce similar errors in devices. The findings can also be used by healthcare providers and regulatory authorities as a reference to raise awareness and understanding of UI software errors and to better evaluate the quality of UI software design in future medical devices.

Key Takeaways

- The authors assessed U.S. medical device recalls during 2012–15 and identified 423 (~140/year) recalls resulting from user interface (UI) software errors, which accounted for nearly one-half of recalls caused by software errors during the period of study.
- A classification of the identified UI software errors (into 20 error categories) was established, as a reference to help stakeholders assess and improve UI software in medical devices.
- The identified UI software errors emerged from a lack of adherence with widely recognized human factors design principles or as consequences of simple design/implementation errors, according to the authors. Detecting the former type of errors depends on comprehensive hazard analysis and/or carefully designed user studies, while sufficient software testing and verification are effective at detecting the latter.
Medical Device Recall and FDA Recall Databases

In the U.S. market, medical device recalls refer to corrective or removal actions taken by manufacturers, voluntarily or required by the FDA, to reduce the safety risks or remedy the violation of laws caused by their devices. Depending on the severity level of health risks involved, medical device recalls are classified as Class I, II, or III, if the devices present a reasonable chance to cause serious injury or death, minor injury, or no health impact to patients, respectively.

To inform the public of devices under recall and their potential risks, the FDA publishes medical device recalls on its public recall database on a regular basis. Each recall record in the database includes a collection of narrative fields describing details of the recall, among which the following two fields offer the greatest value to the current study:

- The Manufacturer Reason for Recall (MRR) field, which briefly describes the malfunction that the recalled device demonstrated during field use, the possible safety risk, and, occasionally, the root cause(s) of the malfunction
- The FDA Determined Cause (FDC) field, which provides the categories of device problems, as determined by the FDA, that have caused the recalls

A limitation of the FDA public recall database is that information included in records is often brief and sometimes incomplete, which can impede a full understanding of the recall scenario.

Additional information often can be found in the FDA’s internal database, the Recall Enterprise System (RES), which is accessible by the FDA only. The RES database maintains regulatory documents related to medical device recalls, including Corrections and Removals (CR) reports that the recalling manufacturers submitted, as well as other supporting documents to justify the scope and strategies of the recall proposed by manufacturers. For example, the CR report includes a root cause section that provides a technical description of the issues in device design, engineering, or manufacturing that triggered a recall.

On rare occasions, the information in the RES may still be insufficient to support accurate identification and classification of UI software errors resulting in device recalls. For example, a manufacturer may simply indicate a software error as the root cause of a recall without providing further details. For these scenarios, if a determination could not be made as to whether the recall was due to software errors or UI software errors, the recall was classified as an “unknown recall” or “undetermined software recall,” respectively, in the current study.

FDA medical device recall data, especially publicly available recall data, has been the subject of numerous academic and industrial studies that have attempted to reveal different categories of medical device problems (e.g., software errors9–12) and their impact on the safety of medical devices in general or on specific types of devices13–15. However, none of these studies has focused on UI software errors.

UI Software and UI Software Errors

Various definitions for software errors exist, with slight differences. This study adopted the terminology used by Simone9 and defines the term software errors as design and implementation flaws and mistakes introduced to the software in a medical device at any point in its development process, from requirement definition, to architectural and detailed software design, and to software implementation and deployment. Errors in the manufacturing software for medical devices, such as software applications for device development, verification and validation, and packaging, are not considered.

Further, this study defines the term UI software as the portion of software governing a medical device’s interaction with users, while the rest of the software in a device is referred to as control software. Control software is responsible for regulating the mechanical and electrical parts of the device to deliver the expected medical functions or for achieving the intended medical purposes if the software itself is a medical device.

A generic UI software architecture was defined in our previous work16 and is depicted in Figure 1. In the current study, this generic architecture—rather than a detailed implementation of the UI software, which may vary substantially across
devices—is used to identify and classify UI software errors in medical device recalls. This allows us to focus on how UI software errors affect interactive functions, rather than where they physically reside in the software. It also helps us to clearly understand how errors in different functional components of the UI software propagate across the rest of the device and, eventually, affect human-device interaction.

**Dataset**

The medical device recalls considered in this study were those announced by the FDA on its public recall database from September 1, 2012, to August 31, 2015. By querying the FDA’s public database, we retrieved a total of 7,771 device recalls during this period, with more than 2,200 recalls occurring during each year (see the first row of Table 3 for details). For brevity of expression, September 2012 to August 2013 was considered year 1, September 2013 to August 2014 year 2, and September 2014 to August 2015 year 3.

**Analysis Method**

To analyze the recalls that were retrieved, a two-phase process was carried out. The first phase identified software-related recalls. In this phase, we implemented a semiautomatic filtering process to quickly eliminate recalls that were not likely caused by software errors. This allowed the next phase of manual review to focus on recalls that warranted more careful study.

The second phase of the analysis identified and classified UI software errors causing recalls that passed the first phase. In this phase, information from the FDA’s RES database was examined manually, in order to identify recalls that were caused by UI software errors. UI software errors observed during the analysis then were classified based on the generic UI software architecture (Figure 1).

**Identification of Software-Related Recalls**

The filtering process in the first phase included an FDC-based filter and a keyword-based filter. Overall, this process eliminated 4,738 recalls (60.97% of total recalls) from subsequent manual analysis.

**Filter 1: FDC filter.** As noted above, each recall posted in the FDA public recall database included an FDC field documenting the category of problems determined by the FDA to be the cause of the recall. A total of 44 categories of problems could be assigned to the FDC field, many of which did not appear to be related to software errors (e.g., labeling, packaging, storage, sterilization). A total of 19 of these categories were considered irrelevant to our study, and a filter was applied to eliminate them from further analysis.

**Filter 2: Key-Word-Based Filter.** The second filter was based on key words. These words were manually selected to identify software-related issues within the recalls. Each recall was analyzed for the presence of these key words, and those deemed to be related to software issues were considered for further analysis. This filter helped to refine the list of recalls to focus on those that were more likely to be caused by UI software errors.

---

**Figure 1.** Generic functional model of user interface software. Interaction control logic defines the procedures through which the user interacts with the device, coordinates with the control software to respond to user inputs, and determines the content and timing of information and feedback to be presented to the user. Drivers for input devices are responsible for detecting user actions on input devices, such as press/release of buttons, rotation of knobs, and touch events on touchscreens. Input interpreter handles complex user input events, such as long-press actions on input widgets, mapping between touchscreen coordinates and input widgets rendered on the touchscreen, and continuous user actions on input devices, such as joysticks and pedals. Output render defines how feedback is presented to the user, such as the format and layout of on-screen information. Drivers for output devices command output devices (e.g., displays, speakers) to present feedback to the user in a manner defined by the output render. Adapted from reference 16. Abbreviation used; I/O, input/output.
developed to eliminate any recall if its FDC field fell into one of those categories.

Of note, to avoid potential omission of recalls caused by UI software errors, we were conservative in selecting irrelevant FDC categories. For example, we preserved the Use Error category, so that recalls in this category could be further examined to determine whether the use errors involved were induced by UI software errors.

Filter 2: keyword-based filter. This filter further eliminated recalls that contained no computer- or use-related keywords in the MRR field of public recall records, based on the assumption that a recall is unlikely caused by (UI) software errors if its description does not involve any of these keywords. This assumption is reasonable, as supported by the study of Alemzadeh et al.\textsuperscript{10}

Table 1 summarizes the set of keywords used in the filtering process. The set includes the same keywords proposed by Alemzadeh et al.,\textsuperscript{10} as well as three additional keywords: \textit{mode}, \textit{display}, and \textit{statistics}. These keywords were added based on our experience that they often appear in the description of medical device recalls resulting from use errors.

Accuracy of the filtering process. Before applying it to the entire recall dataset, we conducted an evaluation of the filtering process to determine whether it would eliminate any recalls caused by software errors. First, we manually analyzed all 2,724 recalls in year 3, based on the information from the FDA’s public and RES databases, to determine root causes. This manual analysis identified 284 recalls as caused by software errors. The filtering process then was applied to these 2,724 recalls, which eliminated 1,650 recalls. The remaining 1,074 recalls included all 284 recalls that our manual analysis identified as being caused by software errors. This evaluation confirmed that the filtering process did not falsely eliminate recalls caused by software errors and, hence, was suitable for application to the entire recall dataset.

Identification and Classification of UI Software Errors
Recalls survived from the filtering process were further analyzed using a manual process to identify and classify UI software errors, according to the following two steps.

<table>
<thead>
<tr>
<th>Category</th>
<th>Keywords</th>
</tr>
</thead>
<tbody>
<tr>
<td>Software</td>
<td>Software, application, function, code, version, backup, database, program, bug, java, run, upgrade, mode, statistics</td>
</tr>
<tr>
<td>Input/output</td>
<td>Sensor, alarm, message, screen, signal, interface, monitor, connect, button, scanner, key, speaker, wireless, terminal, communication, display</td>
</tr>
<tr>
<td>Other</td>
<td>Error, system, fail, verification, self-test, reboot, web, calculation, document, performance, workstation</td>
</tr>
</tbody>
</table>

Table 1. Keywords for filtering recall data

Categorization Rules

<table>
<thead>
<tr>
<th>Defective Functional Components</th>
</tr>
</thead>
<tbody>
<tr>
<td>If the software error leading to a recall affects handling of user input and:</td>
</tr>
<tr>
<td>Causes misdetection of simple input events (e.g., click, press/release).</td>
</tr>
<tr>
<td>Causes erroneous interpretation of complex input events (e.g., continuous input from joystick).</td>
</tr>
<tr>
<td>Affects user input functionalities (e.g., data entry, login).</td>
</tr>
<tr>
<td>If the software error leading to a recall affects feedback presented to the user and:</td>
</tr>
<tr>
<td>Causes malfunction of output devices (e.g., speaker not functioning).</td>
</tr>
<tr>
<td>Causes user feedback to be presented in an erroneous manner (e.g., degraded quality of medical images).</td>
</tr>
<tr>
<td>Causes incorrect timing, location, and content of user feedback.</td>
</tr>
<tr>
<td>None of the above; however:</td>
</tr>
<tr>
<td>The software error affects the performance and availability of the user interface.</td>
</tr>
</tbody>
</table>

Table 2. Rules for associating observed software errors with user interface (UI) software functional components

Step 1: identify recalls caused by UI software errors. The information available in the RES database was independently reviewed to determine whether each recall was caused by UI software errors. If disagreement occurred, the authors discussed the recall until consensus was reached.

Table 2 summarizes the criteria for deciding whether a recall was caused by UI software errors and, if it was, how such errors were associated with the abstract functional components in Figure 1. Of note, there are
UI software errors that cannot be associated with a specific functional component but instead lead to degraded performance or unavailability of the UI. We classified such errors as "general UI software errors."

Table 3 summarizes the numbers of recalls identified as being caused by software errors and/or UI software errors. As noted above, when the information in the RES database was not sufficient to identify or classify UI software errors, the corresponding recalls were labeled as either unknown or undetermined software recalls (Table 3).

**Step 2: classify UI software errors.** The UI software errors identified in step 1 were classified into different categories to capture their nature and impact on device-user interaction. In particular, UI software errors associated with each functional component in Figure 1 were clustered into the same category if they affected the same UI functionality. For example, all UI software errors causing clinical/device status data (except for medical images) to be rendered incorrectly on the display were clustered into the category "incorrect rendering of clinical/device data."

The selection of UI functionalities affected was driven by the observed UI software errors. Some of these functionalities were specific to certain types of devices (e.g., rendering of medical images in computed tomography [CT] or magnetic resonance imaging [MRI] devices) or to specific technologies (e.g., barcode readers), while others were generally applicable across different device types (e.g., the category for issues in the design and configuration of alarms).

When necessary, error categories with excessive occurrences of UI software errors were further subcategorized based on manifestation of the errors. For example, the category "incorrect rendering of clinical/device data" was divided into three subcategories that differentiated how the errors in this category affected different aspects of rendering clinical or device data.

This data-driven classification process established a hierarchy of 20 error categories (Table 5), and 17 subcategories. This structured hierarchy can be used as a starting point toward developing a comprehensive UI software error taxonomy.

### Results

Table 3 summarizes the numbers of recalls caused by software errors and UI software errors. During the three years considered in the study, 11.75% (913 of 7,771) of the recalls were determined to be caused by software errors in the subject devices.

**Impact of UI Software Errors**

A total of 423 recalls were caused, partially or entirely, by UI software errors, accounting for 5.44% of total recalls or 46.33% of recalls due to software errors. An 12 additional recalls were caused by software errors, but we could not accurately determine whether these errors were in the UI software.

The majority (96.47%) of the recalls caused by UI software errors were Class II, meaning that the recalled devices presented reasonable odds of causing minor injury to user or patient. The remainder of the recalls (14 total) were equally divided into Class I and III.

As shown in Figure 2, the types of medical devices affected most by UI software errors were (1) medical imaging systems (e.g., CT and MRI devices), (2) picture archiving and communication systems (i.e., devices for storing, transmitting, and viewing medical images), (3) cardiovascular devices (e.g., external defibrillators or cardiac monitoring systems), and (4) radiation therapy devices. These devices commonly depend on sophisticated UI software to enable complex operational procedures and intricate device-user interactions, which explains why they are more prone to UI software errors than other devices.
Classification of UI Software Errors

The information from the FDA’s public and RES databases revealed 499 UI software errors as the root causes of medical device recalls. In many cases, a device was recalled because of more than one UI software error (which is why the rightmost column of Table 4 adds up to more than 100% of the recalls considered).

Table 4 provides the breakdown of these UI software errors into the functional components of the generic UI software architecture, where interaction control logic and output render are affected the most by UI software errors. Errors in these two components account for 80.16% and 12.63% of the observed UI software errors, respectively. This suggests that more scrutiny should be given to the design, implementation, and verification of these two components.

To support a better understanding of the nature and distribution of observed UI software errors, we further classified the errors into 20 categories (Table 5). The numbers of occurrences of errors in every category also are listed in Table 5. The full hierarchy of UI software error categories and subcategories can be found at https://online-docs.herokuapp.com/recalls.

The remainder of this section defines these error categories and explains their impact on the safety of medical devices, with examples provided for illustration.

Error Categories Associated with Interaction Control Logic

Incorrect coordination with control software components. The largest category of errors within interaction control logic were due to coordination errors between the UI software and control software. A total of 87 errors were found in this category, primarily manifesting as follows:

- The UI software failed to fetch the correct, complete, and up-to-date clinical and device status data from the control software to decide user feedback. As a result, the user’s ability to monitor the operation and clinical outcome of the device was affected. Examples of these errors included interruption to the process of fetching data from the control software due to errors such as multithread programming mistakes, failure to fetch the latest patient monitoring data for display, failure to monitor and report unavailability of clinical/device data, misidentification of the source of data (e.g., identifying the data from sensor A as being from sensor B), failure to fetch and display certain data critical for the user to monitor the device (e.g., radiation output level), use of cached data rather than the latest clinical data to decide user feedback, and calculating the device status based on the status of on-screen widgets rather than the information from the control software (e.g., reporting the treatment as being in process...
when the “start treatment’ button is pressed, but the treatment has not yet started).

- The UI software failed to send the control software the commands entered by the user or trigger it to respond to user commands. Example errors included the UI software violating the interprocess communication protocols and sending incorrect user commands to the control software or the UI software failing to trigger the control software to recalculate the dosage of radiation therapy when the user made change to the treatment plan, resulting in a mismatch between the dosage displayed on screen and the dosage actually delivered.

Incorrect or missing interaction options or on-screen instructions. A total of 84 UI software errors were related to providing the user with incorrect options and instructions to interact with devices. These errors primarily manifested as follows:

- Making incorrect interaction options available to the user. For example, the UI software allowed the user to disable alarms that were critical for safe device use or to clear device error warnings and proceed with the treatment. In many cases, incorrect interaction options were not purposefully introduced but were due to inadvertent coding flaws in the interaction procedures. For example, an infusion pump was recalled because it disabled all safety checks on the infusion program entered by the user if the user pressed the “hold” button twice. This behavior did not appear to be intended by the developers.

- Making interaction options available to the user at an incorrect time. For example, the UI software did not process any user input during system reboot but failed to disable the buttons on the front panel. As a result, the pressing of these buttons by the user during the reboot process may have overflowed the input buffer and caused the system to freeze.

- Incorrect instructions presented to the user. Example errors included the labels on soft keys being incorrect due to translation errors and on-screen instructions being incorrect regarding the tests needed for the current in vitro sample.

- Interaction options not being available to the user when they should have been, potentially affecting the user’s ability to operate the device. Example errors included a radiation therapy machine not providing the emergency stop button on its remote control console and the text window through which the user entered inputs being mistakenly deactivated.

Incorrect determination of the content and timing of user feedback. We observed 72 errors in deciding the content or timing of

<table>
<thead>
<tr>
<th>Component</th>
<th>Categories of UI Software Errors</th>
<th>No. Occurrences</th>
</tr>
</thead>
<tbody>
<tr>
<td>I/O device drivers</td>
<td>Erroneous recognition/registration of user input</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>Driver errors cause I/O devices to be frozen or irresponsible</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>Barcode reader decoding errors</td>
<td>4</td>
</tr>
<tr>
<td>Input interpreter</td>
<td>Erroneous interpretation of complex user input</td>
<td>3</td>
</tr>
<tr>
<td>Output renderer</td>
<td>Medical images are displayed using incorrect size, format, quality, or colors</td>
<td>32</td>
</tr>
<tr>
<td></td>
<td>Incorrect rendering of clinical/device data</td>
<td>29</td>
</tr>
<tr>
<td></td>
<td>Incorrect rendering of on-screen widgets</td>
<td>3</td>
</tr>
<tr>
<td>Interaction control logic</td>
<td>Incorrect coordination with control software components</td>
<td>87</td>
</tr>
<tr>
<td></td>
<td>Incorrect or missing interaction options or on-screen instructions</td>
<td>84</td>
</tr>
<tr>
<td></td>
<td>Incorrect determination of content and timing of user feedback</td>
<td>72</td>
</tr>
<tr>
<td></td>
<td>Incorrect design or configuration of alarms</td>
<td>48</td>
</tr>
<tr>
<td></td>
<td>Incorrect data entry functionalities</td>
<td>42</td>
</tr>
<tr>
<td></td>
<td>Incorrect save functionality</td>
<td>27</td>
</tr>
<tr>
<td></td>
<td>Incorrect login mechanism</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>Incorrect handling of simultaneous user inputs</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>Incorrect cancel/undo functionality</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>Incorrect design or configuration of audio notifications</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>Incorrect timeout mechanism for user inactivity</td>
<td>3</td>
</tr>
<tr>
<td>General UI software errors</td>
<td>Incorrect integration with the running environment</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>Incorrect integration with the rest of device software</td>
<td>7</td>
</tr>
</tbody>
</table>

Table 5. Categories of user interface (UI) software errors observed. Abbreviation used; I/O, input/output.
user feedback, which resulted in incorrect, incomplete, misleading, or out-of-sync feedback being presented to the user. These errors were divided into five subcategories:

1. Incorrect user feedback in different areas of the display or across different control consoles. For example, a medical imaging system displayed the session time and other critical parameters on the main screen but not on the remote control console.

2. Incorrect on-screen auxiliary information. Medical devices often provide auxiliary information, such as units of measurement, clinical references, and the order and orientation of medical images, to help the user better understand the displayed information. We observed 14 errors related to presenting auxiliary information to the user, such as left and right labels being incorrectly associated with medical images on the screen, thereby causing the user to incorrectly interpret or understand the displayed information.

3. Incorrect processing or management of correct clinical/device status data. Errors of this type incorrectly processed or managed the clinical/device status data received from the control software when such data were presented to the user. For example, seconds were rounded up to minutes when displayed, the information to be displayed was incorrectly translated to a foreign language, test results that should have been filtered out were displayed to the user, certain information critical for the user to monitor the treatment of the device (e.g., the technique of radiation exposure for radiation therapy systems) was excluded from user feedback, and the queue of information to be presented to the user was incorrectly managed, so that the information from one patient was displayed for another patient.

4. Incorrect update of the content of user feedback upon user actions. For example, when the user rotated the screen of the smartphone, the UI software incorrectly reset the recommended drug dosage displayed on the screen to zero, which might go unnoticed and therefore be accepted by the user.

5. Incorrect timing for user feedback. For example, the UI software allowed medical images to be exported before they were finalized, causing clinicians to diagnose using incorrect images. Another example was a lab test machine giving the user a green light to load the next sample when it was still processing the current sample, therefore resulting in incorrect association between samples and test results.

Incorrect design or configuration of alarm functionalities. Alarms have been adopted by most devices as an important safety feature attracting user attention to critical events, such as deterioration of patient condition or device failures. We observed 48 errors in design and configuration of the alarm system, including low-priority alarms incorrectly disabling or overriding high-priority alarms, pausing one alarm resulting in all other alarms being erroneously disabled, paused alarms never returning, alarms not being displayed on every user console if one console restarts, a lack of alarms for critical system errors, and alarm messages being too brief or incorrect. All of these errors affect user ability to observe and understand alarm events in a timely manner.

Incorrect data entry functionalities. Users depend on correct data entry functionalities to configure devices, set up clinical processes, and take notes. We observed 42 errors that affected the user’s ability to enter the intended data. They were grouped into the following six subcategories:

1. Missing or incorrect check of user inputs. Errors of this kind cause the UI software to accept user input that is invalid or inappropriate for device operation. For example, a device was recalled because it allowed the user to enter an output energy level outside the device’s capability. Another example was the UI software allowing the user to assign a region of interest (ROI) on the displayed medical image with the same name as other ROIs, which can lead to mistakes in subsequent analysis of the image.

2. Incorrect deletion. For example, when the user deletes clinical data currently displayed on screen, the same deletion could be mistakenly applied to other...
data opened in the background. Another example is that when the user deletes the input value without entering a new one, the UI software incorrectly sends the deleted value to the control software without informing the user.

3. Incorrect editing. For example, because of the wraparound behavior of the data entry system, pressing and holding down the decrement button could erroneously lead to entering the maximum value.

4. Incorrect selection. For example, if the user uses the mouse to select a patient name from a dropdown list and then uses the arrow keys on the keyboard to select another patient, the actual patient name selected could be the one selected with the mouse.

5. Incorrect unit or format conversions of user input. The UI software might incorrectly convert the data entered by the user (e.g., incorrectly converting dates entered in a "dd/mm/yyyy" format as "mm/dd/yyyy").

6. User input being applied without the user's confirmation. For example, when the user enters a new patient ID, the UI software might automatically use it to override the current clinical data without the user's awareness or confirmation.

Incorrect save functionality. We observed 27 errors related to saving user edits and annotations on the displayed patient treatment plans or diagnoses. These errors lead to either loss of data or data being saved for wrong patients.

Examples of these errors include the UI software saving the physician’s edits on patient prescriptions to cache files but not to the database, potentially resulting in edits being lost; user annotations on one medical image being erroneously saved to another medical image when multiple images are displayed at the same time; user edits not being saved when the user closes or navigates away from a display window; and an image currently under view being erroneously saved to the end of the new series when a new image series arrives.

Incorrect and inappropriate login mechanisms. We observed 16 UI software errors in the design of login and access control mechanisms; they included:

- Imposing inconvenient user login details that may delay patient treatment. For example, a device forced the user to start from service mode and enter passwords after an AC power disconnection.
- Imposing inappropriate security policies that affect users’ access to the devices. For example, errors in antivirus software in one device erroneously allowed users to access the device without having to log in, while another device was recalled because its security settings prohibited legitimate users from remotely accessing the device.

Incorrect handling of simultaneous user inputs. We observed eight UI software errors in handling simultaneous user inputs that resulted in critical system failures. Example errors included the device locking up when the operator pressed two or more buttons simultaneously; the device losing functionalities if the user changed the treatment plan and, at the same time, released the foot pedal to start the treatment; and images freezing on screen when the the mouse was used to zoom in/out and, at the same time, keys on the keyboard were pressed.

Incorrect cancel/undo functionalities. Cancel and undo functionalities provide medical devices with important measures for accommodating and recovering from use errors in operating devices. Typically, the former is used for revoking a user command and the latter for rolling back to a previous device state before a use error occurs. We observed seven errors in these two UI functionalities. For example, if the user presses the cancel button after entering therapy parameters that violate safe limits, the UI software will stop checking all subsequent user inputs, and if the undo button is pressed during medical image readout, all acquired images will be erroneously deleted.

Incorrect timeout mechanisms for user inactivity. We observed three errors in the design of automatic timeout mechanisms (e.g., locking screen), which are designed to prevent unexpected device-user interaction during user inactivity. For example, if a device does not provide a timeout mechanism that clears the displayed clinical recommendations after a period of user inaction, then the on-screen recommendations might no longer be appropriate, or if the user exits the timeout
Incorrect order due to sorting errors, and truncated, lab test results being displayed in large numeric values being displayed as formats. Example errors included fractional or which often are in numeric or graphic data (e.g., motor speed, output energy levels), cal measurements, lab test results) and device patient demographic information, physiologi-

In addition to medical images, we observed images.

Incorrect handling of the width and height of and displaying images as squeezed due to screen" rather than in their actual sizes, the display to freeze, displaying images as "fit mic errors in rendering images that caused a 6-bit color scale rather than 8-bit, algorithm errors in rendering images in a 6-bit color scale rather than 8-bit, algorithmic errors in rendering images that caused the display to freeze, displaying images as "fit to screen" rather than in their actual sizes, and displaying images as squeezed due to incorrect handling of the width and height of images.

Incorrect rendering of clinical/device data. In addition to medical images, we observed 28 errors in rendering other clinical data (e.g., radiologists, oncologists) rely to correctly diagnose and treat patients. Rendering medical images incorrectly certainly would affect correct clinical decision making for these users.

We observed 32 errors in which the quality, size, format, quality, or color of medical images displayed to the user were affected. Examples of these errors included displaying images in a 6-bit color scale rather than 8-bit, algorithmic errors in rendering images that caused the display to freeze, displaying images as "fit to screen" rather than in their actual sizes, and displaying images as squeezed due to incorrect handling of the width and height of images.

Incorrect rendering of on-screen widgets. On-screen widgets include device controls (e.g., soft buttons), input components (e.g., text fields), and navigational tools (e.g., scroll bars) necessary for the user to operate the device and browse the information presented on the display. (Note: A set of common on-screen widgets can be viewed at www.useability.gov/how-to-and-tools/methods/user-interface-elements.html.) Errors in correctly rendering on-screen widgets on the display (three of which were observed in this study) affect the visual appearance of the widgets (e.g., size, shape, status), which in turn hinders the user’s interaction. An example error would be a scroll bar for a text field showing that the user has reached the end of the text when more text is to be viewed. As a result, the user is likely to overlook the remaining text.

Error Categories Associated with Output Render
Medical images might be displayed in an incorrect size, format, quality, or color. In many types of devices, medical images are an important type of feedback on which users (e.g., radiologists, oncologists) rely to correctly diagnose and treat patients. Rendering medical images incorrectly certainly would affect correct clinical decision making for these users.

Error Categories Associated with Input/Output Device Drivers
Erroneous recognition/registration of user input. Eight errors were found in the driver or firmware for input devices (e.g., keyboard, touchscreen, foot pedal), resulting in erroneous recognition or registration of user actions. In addition to classic software errors in distinguishing stuck keys and legitimate key presses, we also observed errors in the driver of other types of input devices (e.g., touchscreen) that caused failures in detecting legitimate user touches or distinguishing them from accidental events (e.g., liquid drops).

Driver errors cause input/output (I/O) devices to be frozen or irresponsible. We observed seven errors in the driver or firmware for I/O devices that caused them to become inoperative or to function incorrectly. Consequently, the user lost the means to correctly interact with the devices. For example, a buffer overflow defect in keypad firmware caused the keypad to lock up and a firmware defect in a speaker caused the speaker to fail to issue audio notifications.

In addition to classic software errors in distinguishing stuck keys and legitimate key presses, we also observed errors in the driver of other types of input devices (e.g., touchscreens) that caused failures in detecting legitimate user touches or distinguishing them from accidental events (e.g., liquid drops).
### Barcode reader decoding errors

Barcode technologies have been introduced to many devices to promote effective device use and reduce data entry errors. Errors in decoding barcodes, however, can void their benefit and cause data entry errors (e.g., associating patients with incorrect treatment orders) that are difficult to detect. We observed four errors related to barcode readers, including the barcode reader adding an incorrect suffix string to the scanned barcode or introducing algorithmic errors in decoding standard EAN-13 barcodes.

### General UI Software Error Categories

- **Incorrect integration with the running environment.** We observed seven errors in integrating the UI software with its running environment, including operating systems (OSs) or other supporting software applications. These errors typically involve interaction functions being disabled. For example, due to incompatibility with the Universal Serial Bus (USB) services provided by the OS, the UI software might fail to manage the display devices via USB connections. Another example would be UI software not being able to launch because the Java Virtual Machine in which it operated crashed frequently.

- **Incorrect integration with the rest of the device software.** Seven errors were found in integrating the UI software with the control software. For example, a device was recalled because its UI software might have created a race condition with the control software when both parties tried to access the device's database simultaneously. In other cases, the operation of the UI software was disrupted by the control software because it consumed most computational resources or inappropriately reset the UI software. All such errors caused the UI to become unavailable or irresponsive.

### Error Category for Input Interpreter

Errors in this category erroneously process and interpret complex and continuous actions that the user performs on input devices (e.g., joysticks, touchscreens) or via on-screen widgets (e.g., scroll bars). As a result, inputs other than what the user intended were registered. For example, an error was observed in which the UI software mistakenly interpreted the user’s action of pushing the joystick to the maximum forward position as pushing it backwards. In another example, when a scroll bar was used to adjust the time value, the UI software incorrectly calculated the position of the slider and the corresponding numeric value. We observed three errors in this category.

### Limitations and Future Work

This study was based on information available in the FDA's recall databases, and to the extent possible, we avoided making our own determination of the root causes of recalls. Thus, the quality of information in these databases, in terms of completeness and accuracy, had a direct impact on the findings and conclusions of this study. Given their legal responsibility, it is unlikely that manufacturers would submit incorrect or misleading information to the recall databases. However, it is possible that the information in the recall databases lacks the level of detail desirable for a study such as the current one, partly because manufacturers might be reluctant to disclose proprietary information or lack the technical expertise to perform thorough root cause analysis for the recalled devices.

Two other factors may have affected the quality of the recall data and, in turn, the findings of this study. The first is the underreporting of medical device problems. Academic and regulatory investigations have revealed that not all medical incidents resulting from device defects have been reported to manufacturers and regulators. This underreporting may cause manufacturers not to initiate medical device recalls when they should. This situation is exacerbated by UI software errors generally being harder to spot and replicate than other device problems.

The second factor is related to the purpose of medical device recalls, which is to protect the public health from defective medical devices. Therefore, the information submitted by the recalling manufacturers focuses...
on the health risks of device problems and the strategies for correcting them; it does not necessarily focus on the root causes of these problems. In some cases, we could not determine whether the recalls were caused by UI software errors, thereby affecting the accuracy of our findings.

The findings described here represent a first step toward improving the quality and safety of UI software in medical devices. Further work is needed to advance these findings and help manufacturers develop effective measures for detecting and mitigating UI software errors.

Conclusion

The current study reported findings from analysis of three years of FDA medical device recall data. This study confirms the prevalence of UI software errors in medical devices and their impact on device-user interaction and, in turn, device safety. Of note, these UI software errors emerged from a lack of adherence with widely recognized human factors design principles or as consequences of simple design/implementation errors. Although detecting the former type of errors depends on comprehensive hazard analysis and/or carefully designed user studies, sufficient software testing and verification are effective at detecting the latter. For example, our previous work suggested that formal methods and other rigorous software engineering methods can be used to avoid or eliminate, by design, many UI software errors, which otherwise are hard to detect and mitigate.19,20

These findings can serve to raise the level of awareness among stakeholders on the importance of correct and appropriate UI software design to the safety of medical devices (and many other safety-critical systems). The detailed classification of UI software errors described here also can help stakeholders to better assess and improve UI software design in future devices.

Funding

Dr. Masci is funded by the European Regional Development Fund through the Operational Programme for Competitiveness and Internationalisation COMPETE 2020 Programme, within project POCI-01-0145-FEDER-006961, and by National Funds through the Portuguese funding agency FCT as part of project UID/EEA/50014/2013. Dr. Thimbleby is a See Change Fellow in Digital Health, funded by See Change, Scotland.

References


Make medical devices safer,
one step at a time.

Writing Human Factors Plans & Reports
for Medical Technology Development

For more information and to order this book, please visit www.aami.org/store.

Product Code: HFP and HFP-PDF