Communications of the IIMA

Volume 9 | Issue 2 Article 6

2009

Integrating Human Computer Interaction Testing into the Medical Device Approval Process

Jim Nindel-Edwards
Microsoft Corporation

Gerhard Steinke
Seattle Pacific University

Integrating Human Computer Interaction Testing into the Medical Device Approval Process

Jim Nindel-Edwards
Microsoft Corporation
USA
jimne@ieee.org

Gerhard Steinke
Seattle Pacific University
USA
gsteinke@spu.edu

ABSTRACT

Medical devices that utilize computer software are becoming common place in today's health care environment. In this paper we examine device failures in the area of the human computer interaction—a failure of the interface between the hardware/software in the medical device, and the person using the device. We make the case that human computer interaction testing—starting early on in the product development life cycle—should be required before medical devices are approved by the FDA. Use of human computer interaction testing of medical devices can improve device quality and user experience, and most importantly, has the potential to reduce serious health care outcomes.

INTRODUCTION

Medical devices that utilize computer software are becoming common place in today's health care, ranging from insulin pumps, to devices that dispense drugs, and those that monitor heart rhythms. Many medical devices have been used successfully to provide better patient care, often with costs savings, and in most situations actually address some areas of human driven errors (Berman, 2004).

At the same time software to program and/or control medical devices can, and have, introduced errors that affect patient outcomes. The errors are in part due to the software or a failure of the interface between the hardware and software in the medical device, and the person using the device. Errors in medical devices raise questions concerning the strength of the design and testing of human computer interactions when developing medical devices. Sufficient human computer interface testing should ensure that most errors are identified before products are released.

The FDA has an approval process for medical devices with differences depending on the criticality level of the application of the medical devices. It appears that human computer interaction testing is not a clearly required component of the FDA medical device approval process. We believe that human computer interaction testing should be required before medical devices are approved by the FDA. The challenge is in specifying criteria to help a manufacturer determine what is "sufficient" human computer interaction testing.

Medical Device Failures

A few examples of problems with the human computer interface of medical devices will provide some background for this paper.

The Therac-25 device was developed to provide radiation to cancer patients. The presence of numerous flaws in the software led to massive radiation overdoses, resulting in the deaths of three people (Leveson, 1993). It was poor user interface controls that caused prescription and dose rate information to be entered improperly (McQuaid, 2009).

A problem with a flow control knob is cited by the FDA (Sawyer, 1996): "A physician treating a patient with oxygen set the flow control knob between 1 and 2 liters per minute, not realizing that the scale numbers represented discrete rather than continuous, settings. There was no oxygen flow between the settings, yet the knob rotated smoothly, suggesting that intermediate settings were possible. The patient, an infant, became hypoxic before the error was discovered. Human computer interaction testing of this device should have identified this problem."

A volumetric infusion pump is a medical device that delivers intravenous fluids and medicine to patients. The Baxter Colleague triple channel infusion pump generates fault codes under the condition of changing a fluid supply at the same time the supply goes to zero (an understandable and appropriate behavior). Unfortunately this fault also stopped the other two channels from continuing to operate, causing a life threatening situation for patients. At least 9 patients' deaths have been attributed to miscommunication between the care giver and the software that runs these medical devices (Infusion Pump Recall, 2009).

The *Journal of the American Medical Association* (Koppel, Metlax, Cohen, Aboluck, Localio, Kimmell, & Strom, 2005) reported on an examination of a hospital computer system and suggested that computers increased the error risk. They said, "…the problem is that hospital computer systems are not designed for the way real-life hospitals work." It appears that their recommendation is for more human computer interaction testing.

Further examples can be found in an article published by Dick Sawyer (1996). And the complexity of user - computer interaction is growing: The deployment of medical devices to the field such that "Caregivers and clinical engineers ... are becoming lost in a swirl of technology, and [we] face unanticipated interference between devices" (Lee & Pappas, 2006). It is clear that some medical devices have failed, at least in part due to the software in the devices.

Human Computer Interaction Testing

When there is an accident involving computers, Marc Green says, "First, the computer interface should be evaluated for adequacy of design. Faulty design could be construed as negligence on the part of the designers".

Human computer interaction testing is being commonly used in many commercial settings to form better human computer interface scenarios (Gosbee & Ritchie, 2007). Popular software developed by organizations such as Microsoft and Adobe, as well as web site applications are expected to be tested for human computer interaction. Similar to human factors engineering used in building non-computerized devices, interaction design can be extremely valuable to hone in on robust command and control of medical devices used to supply care to patients (Sawyer, 1996). The requirement for medical device manufacturers to adopt the use of interaction design in the development of medical devices could – literally – be a life saving event.

The users of medical devices are both patients and medical professionals. This means the human computer interaction testing needs to include both types of users. Patients come with all levels of background and experience with technology. Medical professionals naturally are very patient focused, not necessarily medical device focused. Human computer interaction testing needs to meet the needs of all potential users.

FDA Standards for medical devices

The Federal Drug Administration has established standards for medical devices and classifies these based on criticality levels of the device usage, ranging from Class I (least critical) to Class III (life support/life sustaining) (FDA, 2006):

- Class I – Devices used in non-life sustaining tasks, subject only to "General Controls".
- Class II – Also non-life sustaining, but subject to both "General Controls" and "Special Controls" (FDA, 2006a).
- Class III – Devices that sustain or support life, subject to both types of controls and may not be introduced to market without FDA oversight and pre-approval requiring in many cases clinical trials.

Class III medical devices require the most scrutiny, and accordingly are the devices that cost the most to develop, test, and support. It is interesting to note that Class II devices can, and are, applied in situations where if not used, or tested appropriately, can result in serious injury and possible patient mortality. And yet the approval process for Class II is significantly reduced from Class III devices. Also, from the FDA perspective, it seems that the difference between hardware and computer controls with associated software is not material. The devices are simply classified as I, II, or III, based on usage. Perhaps the testing could also be based on the complexity of the device and the associated software—the more complex, the more testing should be required.

FDA Recommendation for software development

There are software standards published by the FDA in 2002 that provide guidelines and requirements for the software used in medical devices, regardless of class (Quality System Regulation, 2006). Various FDA documents specify that the "general controls" associated with software in medical devices follow existing standards for requirement driven software projects, e.g. the classic waterfall model overlaid by a risk based analysis termed "Level of Concern". This is summarized in the *Guidance for the Content of Premarket Submissions for Software Contained in Medical Devices* documented in the following decision matrix (FDA, 2006):

The criticality of devices is divided into three classes: Class I, II and III. The software is divided into three areas as well: Minor Concern, Moderate Concern and Major Concern (see Table 1). There is some relationship between the two. That is to say that a medical device could be a class III (highest standards) yet the software associated with the device be only a "minor concern" if the software component was actually peripheral to the Class III rating. This in fact was the situation with the first model of the Therac as the hardware provided the protection and the associated software was of "moderate concern". When the hardware interlock was removed the software moved from "moderate" to "major" without the appropriate review of the software component.

Table 1: Software documentation.

SOFTWARE DOCUMENTATION	MINOR CONCERN	MODERATE CONCERN	MAJOR CONCERN
Level of Concern	A statement indicating the Level of Concern and a description of the rationale for that level.		
Software Description	A summary overview of the features and software operating environment.		
Device Hazard Analysis	Tabular description of identified hardware and software hazards, including severity assessment and mitigations.		
Software Requirements Specification (SRS)	Summary of functional requirements from SRS.	The complete SRS document.	
Architecture Design Chart	No documentation is necessary in the submission.	Detailed depiction of functional units and software modules. May include state diagrams as well as flow charts.	
Software Design Specification (SDS)	No documentation is necessary in the submission.	Software design specification document.	
Traceability Analysis	Traceability among requirements, specifications, identified hazards and mitigations, and Verification and Validation testing.		
Software Development Environment Description	No documentation is necessary in the submission.	Summary of software life cycle development plan, including a summary of the configuration management and maintenance activities.	Summary of software life cycle development plan. Annotated list of control documents generated during development process. Include the configuration management and maintenance plan documents.
Verification and Validation Documentation	Software functional test plan, pass / fail criteria, and results.	Description of V&V activities at the unit, integration, and system level. System level test protocol, including pass/fail criteria, and tests results.	Description of V&V activities at the unit, integration, and system level. Unit, integration and system level test protocols, including pass/fail criteria, test report, summary, and tests results.
Revision Level History	Revision history log, including release version number and date.		
Unresolved Anomalies (Bugs or Defects)	No documentation is necessary in the submission.	List of remaining software anomalies, annotated with an explanation of the impact on safety or effectiveness, including operator usage and human factors.	

The key reports associated with the FDA specifications are listed in the first column above:

- Software Requirements Specification (SRS),
- Architecture Design Chart,
- Software Design Specification (SDS),
- Traceability Analysis,
- Software Development Environment Description,
- Verification and Validation Documentation,
- Revision Level History, and
- Defect Reports.

The key deliverable is the Verification and Validation (aka V&V) documentation. While a common term in the software industry, we'll reflect here on the FDA's definition (FDA, 2002):

- Software verification provides objective evidence that the design outputs of a particular phase of the software development life cycle meet all of the specified requirements for that phase. Software verification looks for consistency, completeness, and correctness of the software and its supporting documentation, as it is being developed, and provides support for a subsequent conclusion that software is validated. Software testing is one of many verification activities intended to confirm that software development output meets its input requirements. Other verification activities include various static and dynamic analyses, code and document inspections, walkthroughs, and other techniques.

- Software validation is a part of the design validation for a finished device, but is not separately defined in the Quality System regulation. The FDA considers software validation to be "confirmation by examination and provision of objective evidence that software specifications conform to user needs and intended uses, and that the particular requirements implemented through software can be consistently fulfilled." In practice, software validation activities may occur both during, as well as at the end of the software development life cycle to ensure that all requirements have been fulfilled. Since software is usually part of a larger hardware system, the validation of software typically includes evidence that all software requirements have been implemented correctly and completely, and are traceable to system requirements. A conclusion that software is validated is highly dependent upon comprehensive software testing, inspections, analyses, and other verification tasks performed at each stage of the software development life cycle. Testing of device software functionality in a simulated use environment, and user site testing are typically included as components of an overall design validation program for a software automated device.

An interesting point here is the statement "that software specifications conform to user needs and intended uses". This assumes that the Software Requirements Specifications (SRS) in and of itself is correct, and Verification and Validation (V&V) assures the device software matches the specifications. What if the SRS itself is flawed in one or more areas? The very best we can say is that we've assured (through the V&V process) that the software perform well to the requirements (SRS) without highlighting the underlying flaw in the SRS.

So we have two areas to examine:

- Does the software perform according to the requirements?
- Are the requirements indeed accurate and complete?

It seems obvious that the latter question should be addressed first. The software quality assurance perspective on this problem is to “drive quality upstream” by verifying the correctness of the requirements, in this instance the SRS and Software Design Specification (SDS). Then the Verification and Validation process can assure that the system meets requirements which have also been tested.

Recommendation: Include Interaction Design Testing in the Development of FDA Approved Devices

The techniques to ensure the requirements are accurate and complete is human factor based design and its counterpart in the software world: human computer interaction design. Driving quality “upstream” would suggest that we require that both human factors analysis (hardware) and interaction design (software) techniques be used not only in the V&V of medical devices, but even earlier in the development of these devices. The software perspective suggests that examining the device and software interface before the device is deployed can identify problems before a device is released to market.

Human computer interaction design can even be a potential advantage to the speed of development and deployment of medical devices as software, by its very nature, can be more easily adapted in the development process than hardware devices. Human computer interaction design call for creating prototypes of human computer interfaces and then testing them for usability. This approach can directly address two of the challenges associated with complex medical devices software, especially the use of "Model Based Development", and User-Centered Design (Lee & Pappas, 2006). Leaving user-computer interaction to the V&V phase to catch design problems suggests that the traditional design techniques are adequate to cover all Human Computer Interaction (HCI) problems, yet the evidence belies that assumption. The use of Interaction Design techniques directly addresses some of the important and even life critical, issues. Having improved (a quality measurement) the initial human device interface the V&V effort can then focus on whether the device is fulfilling its mission.

The only mention of human factors in the FDA’s guidance document (FDA, 2002) relates to "Unresolved Anomalies" (e.g. defects). The recommendation does not include the use of either human factors or interaction design early in the hardware/software development effort—the very place where the payback is the greatest. The critical step of giving instructions (some would say programming) the devices to dispense critical care is essential to the effective use of these devices. This is the role of the human machine interface, both in the “programming” of the device and the feedback given from the device to the human, both to “understand” the instructions, and the on-going “status” reporting that the “instructions” are being followed. Making certain that the human computer interaction is an accurate and safe experience is a reflection of a high quality medical device.

Medical devices put a heavy responsibility on the design team responsible for the development and implementation of medical devices. The medical device design teams is not are not licensed care givers but computer hardware, and software developers. Theirs is the job of both discerning not only the requirements and anticipated uses of medical devices, but the potential for boundary, outlier, and simply misuse of these devices. It is worthwhile to repeat Marc Green's comment: "Faulty design could be construed as negligence on the part of the designers."

The goal of interaction design, and its associated testing, is to verify at the start of the design phase the usability of the yet-to-be built software. Concentrating on what, and how, a critical heath care application is intended to respond is certainly a factor in the development and delivery of functionality. Interaction design and its associated "testing" of medical device performance can assure that the device delivers its critical care under the anticipated circumstances and continues to delivers critical care when circumstances are less than ideal. Any of us can suggest that if events had only gone as we expected them to unfold, everything would "be ok". Interaction design allows us to "what if" the unexpected happens and test whether the systems (device plus actor) respond appropriately.

A challenge for the FDA as well as a medical device manufacturer is the question: What is appropriate and sufficient testing? How much testing is enough? Often it is impossible to test all possible conditions and situations. Certainly the testing should relate to the FDA's definition of Class I, II and III devices. Perhaps there should also be some factor as to the complexity of the software in a particular device, with more complex software functions calling for more testing. We would also suggest that an independent organization verify the testing performed by the manufacturer and provide some level of certification as to the testing that has been performed.

CONCLUSION

Human computer interaction has been successfully used in many commercial settings to form better human computer interface scenarios. Similar to human factors engineering used in building non-computerized devices, human computer interaction design testing can be extremely valuable to hone in on robust command and control of medical devices used to supply critical care to patients. The FDA should require medical devices manufacturers to test the interaction design during the development of medical devices. In addition, we suggest an independent certification for human computer interaction testing. Further research is necessary to determine certification levels and their relationship to the criticality levels of the medical devices.

REFERENCES

Berman, A. (2004). Reducing medication errors through naming, labeling, and packaging, *Journal of Medical Systems*, 28(1), 9-29.

FDA (2002). *General Principles of Software Validation; Final Guidance for Industry and FDA Staff*, http://www.fda.gov/downloads/MedicalDevices/DeviceRegulationandGuidance/GuidanceDocuments/ucm085371.pdf

FDA (2005). *Guidance for the Content of Premarket Submissions for Software Contained in Medical* Devices, May.

http://www.fda.govhttp://www.fda.gov/downloads/MedicalDevices/DeviceRegulationandGuidance/GuidanceDocuments/ucm089593.pdf

FDA (2006). *FDA Regulatory Requirements and Approvals,* http://www.cdaservices.com/fdaapproval.htm

FDA (2006a). *Device Classification Panels,* http://www.fda.gov/MedicalDevices/DeviceRegulationandGuidance/Overview/ClassifyYourDevice/ucm051530.htm#regs

Gosbee, J., & Ritchie, E. (2007). Human–Computer Interaction and Medical Software, *Interactions*, July/August.

Green, Marc (n.d.). - Error and Injury in Computers and Medication Devices, http://www.visualexpert.com/Resources/compneg.html

Infusion Pump Recall (2009). http://www.baxter.com/about_baxter/press_room/documents/2009/03_11_09_colleague.pdf

Koppel, R., Metlay, J. P., Cohen, A., Abaluck, B., Localio, A. R., Kimmel, S. E., & Strom, B. L. (2005). Role of Computerized Physician Order Entry Systems in Facilitating Medication Errors. *Journal of the American Medical Association*, 293(10), 1197-1203. http://jama.ama-assn.org/cgi/content/full/293/10/1197

Lee, I., & Pappas, G. (2006). High-Confidence Medicate Devices Software and Systems, *IEEE Computer*, 39(4), 33 – 38.

Leveson, N. (1993). An Investigation of the Therac-25 Accidents. *IEEE Computer*, 26(7), 18-41. http://courses.cs.vt.edu/~cs3604/lib/Therac_25/Therac_1.html

McQuaid, P. (2009). Software Disasters: What Have We Learned? *California Polytechnic State University SQP*, 11(3), ASQ.

Quality System Regulation (2006). *21 CFR Part 820* - http://www.gmp1st.com/mdreg.htm

Sawyer, D. (1996). Do It By Design - An Introduction to Human Factors in Medical Devices, http://www.fda.gov/downloads/MedicalDevices/DeviceRegulationandGuidance/GuidanceDocuments/ucm095061.pdf