Medical Software's Increasing Impact on Healthcare and Technology Management

Barbara Majchrowski

The use of medical software in healthcare is growing, but its impact is not well understood. The management of software within a healthcare facility has unique characteristics when compared with the management of hardware. This uniqueness poses its own risks to patient safety and requires its own management techniques. And, as healthcare becomes more interconnected and dependent on various technologies working together to create a more efficient, cost-effective, and safer healthcare system, it is required that the various layers of software function well with seamless interfaces, excellent reliability, and good quality. This article will explain some of the complexities surrounding medical software and how they can affect technology management issues such as hazards and recalls, software upgrades, and interoperability.

Medical software is pervasive, ranging from the embedded software in medical devices to information systems exchanging medical data. Today, there are very few analog-only medical devices remaining. Most devices have a computerized component. For these devices, it is generally acknowledged that the software provides the majority of the device's capabilities when compared with the physical components (e.g., transducer). While the design of the "box" (e.g., mechanized components) is extremely important, the software typically controls the device's key functionality. Among software's many roles, it can contain logic that sets alarm limits, annunciates alarms, and processes a raw signal via an algorithm to display a recognized measurement that is understood by clinicians.

In several guidance documents for the industry,[1,2,3] the U.S. Food and Drug Administration (FDA) has stated that medical software falls under federally required quality and design controls that apply to firmware, standalone software applications, software for installation in general-purpose computers, accessories to medical devices containing software, and commercial off-the-shelf software used in medical devices. This means that vendors of medical software must meet certain criteria before market (e.g., following good manufacturing practices).

In addition to software-only medical devices, information systems can be classified as medical devices depending on the type of data exchanged and the risk to the patient should the information system fail or malfunction. The FDA has three levels of classification for medical devices, ranging from the lowest risk (Class I) to highest risk (Class III). To illustrate, laboratory information systems are listed as Class I medical devices, and picture archiving and communication systems (PACS) are Class II medical devices. This means that the exchange of medical data—made possible by software code—can and does affect patient care. Should a problem occur with these systems, not only could business inefficiencies result, but also patient harm. As with other medical devices, patient injuries resulting from anomalies with information systems or software-only medical devices should be reported to the FDA.

Furthermore, we may see more types of devices requiring federal oversight as the breadth of FDA authority increases. In February 2008, FDA proposed a draft regulation for a medical device data system (MDDS).[4] An MDDS is defined as software intended to exchange, retrieve, store, display, and/or reformat medical device data without interfering with the medical device itself. Anyone (hospitals as well as vendors) who develops a device falling under the definition of a MDDS would be required to follow Class I medical device controls, including registering the device, following Good Manufacturing Processes (21 CFR Part 820), and complying with Medical Device Reporting requirements (21 CFR Part 803). When the MDDS ruling becomes final, we may see a change in the marketplace that will invariably affect

Barbara Majchrowski, MHSc, P.Eng., is a senior project engineer in the Health Devices Group at ECRI Institute. E-mail: bmajchrowski@ecri.org.

healthcare and, consequently, software management.

Medical Software Complexity

It is clear that medical software is a growing issue for the healthcare industry. While change is forthcoming, facilities must deal with the current state of medical software and its complexities. First, it must be understood that no software is fail-safe. It is not a matter of *if* a software-related problem will occur, but rather *when* a software-related problem will occur. Even a manufacturer with excellent quality systems, effective risk management, and a successful safety culture will release defective software at some point. As a result, testing is a critical step. Although errors can never be reduced to zero, vendors can employ various techniques to minimize the number of anomalies, which could be functional in nature, security weaknesses, or errors such as spelling mistakes.[5]

Moreover, we cannot predict every use case for a device. Test procedures and verification and validation processes can run the device through the lion's share of expected use cases; however, upon release, it is only a matter of time before a user will find a new and unexpected way of operating the device. This could include off-label use, unforeseen manipulation of controls, a multi-step procedure performed out of order, and inappropriate commands, among other deviations. Once released to the public, manufacturers have no control over how a product will be used. As a result, many latent problems (e.g., unexpected performance, device shutdown) are discovered after a device has been marketed.

Software quality also depends on the input obtained during the software development phases. The design of medical software requires that users such as clinicians or clinical engineers provide the requirements of the device. These requirements must be communicated to the software programmers. This requires a common language easily understood by both groups, a task easier said than done.[6] It is a skill to use the English language in a way that can describe the clinical requirements well and can also be translated into software code. This is a critical step since the clinical requirements define the scope of the device or system, which then defines the software requirements.[7]

Also, consider all the different types of software that need to operate collectively. A single device could have firmware (the programmable piece of a hardware component such as a microprocessor), application software (software designed for a user's specific needs), an operating system (software that controls program execution, data management, etc), utility software (software that performs general support such as electronic media formatting), and drivers (software that allow peripheral devices to connect to a larger system).[8] And there may be software that layer on top of the application software such as infusion pump drug libraries (software containing a list of commonly used medications and dosages), which could have multiple versions to cover the adult to neonate patient populations. All of these software components must work well together to create an effective and functioning medical device.

To further complicate matters, healthcare facilities are faced with convergence—the interrelationship between medical and information technologies.[9] Examples of converging technologies include medication management systems, physiologic monitors on the hospital-IT network, the routing of medical alarms to clinician-worn devices (e.g., cell phones), and the exchange of medical data into various information systems and electronic medical records. There are many reasons to embrace converging technologies. It can allow for better communication of medical information, leverage existing equipment within an organization, facilitate work processes, and help improve patient care.[9, 10] However, in relation to software, convergence means that medical software intermingles with nonmedical software, which can increase the likelihood that different coding elements will interfere with one another.

Technology Management and Medical Software

Software-related problems will occur. What we do not know is when, how frequently, what devices, whether one patient or multiple patients will be affected, under what conditions a problem will present itself, how the problem will manifest itself, how severe the problem will be, or what the downstream effects may be. Similarly, we do not know how "easy" the fix will be, how long the hazardous condition will exist (e.g., how soon a software upgrade can be made available), what action might be needed before a solution is implemented, or how inconvenient or risky the temporary action may be.

This uncertainty can be nerve-racking for a clinical engineer or biomedical technician tasked with the job of supporting and maintaining medical technology. And, as the trend toward convergence continues and IT staff members are getting closer to the patient's bedside, they are confronted with the same challenges and mission to

protect patient safety.

Clearly, medical software is an issue that healthcare facilities must deal with. Just as manufacturers must have processes in place to minimize the risks that medical software poses, so must healthcare facilities.

Software-Related Hazards and Recalls

As software in medical devices has become more complex with more lines of code, it's no surprise that the frequency of software-related problems is increasing. This is supported by results shown in Figure 1. ECRI Institute's Health Devices Alerts database is a repository of hazards and recalls reports (sourced from various national and international regulatory agencies, manufacturers, and member hospitals).[11]

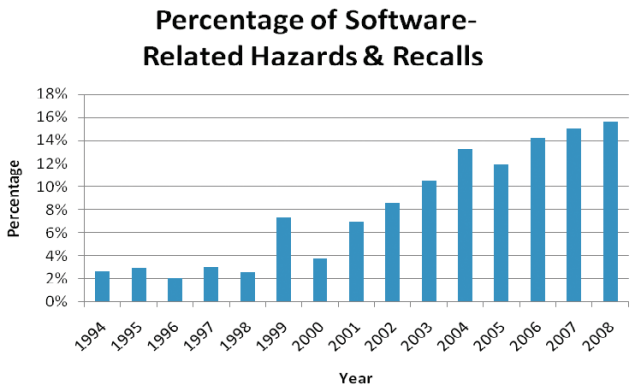

Figure 1. ECRI Institute Health Devices Alerts Database: Percentage of software-related hazards and recalls.

Figure 1 illustrates the results of a search performed on the Health Devices Alerts database using the parameter "*software*" on "active" or "completed" alerts. The results were normalized over a 15-year period from 1994 to 2008. Care was taken to ensure that no duplicate alerts were counted (i.e., when an original report was supplemented with later updates, those updates would not be considered a new and separate problem). The 15-year trend shows an obvious increase in the frequency of software-related hazards and recalls, with a spike in 1999 likely due to the "Y2K" phenomenon.

ECRI Institute organizes hazards and recalls into three categories depending on the risk level: critical, high, and normal priority. Critical-priority alerts are those that, if left unresolved, can result in patient or user death, severe injuries, or permanent health problems. In the first six months of 2009 alone, there were 154 software-related alerts, 13 of which were rated critical priority. (Some alerts were due to both software and hardware problems.) Potential ramifications of these problems included failure to sound alarms, over- or underinfusion of drugs, and failure or interruption in therapy.

To manage the increasing number of software-related hazards and recalls, healthcare facilities need to have good practices in place. Hospitals may need to assess their current medical device hazard and recall management policies and procedures to determine whether their current system can handle software-related recalls. Facilities generally have several contingency plans that can be exercised should a problem occur. For instance, a healthcare organization may consider leasing options for medical devices should their inventory be subject to a recall. Such a solution could work for either a hardware- or software-related recall. However, not all the contingency plans currently in place may be applicable for hazards and recalls concerning medical software such as those affecting the correct transfer of medical data between multiple systems.

Healthcare facilities should plan as best they can for software-related hazards and recalls. This could include activities ranging from good documentation of software within their facility to understanding the hierarchies of software such as the interactions existing between the medical software and other information systems. These actions can help to proactively develop contingency plans applicable to medical software. And, the identification of hazards caused by software failures can be facilitated through vigilant observation of odd or unexpected behavior. These observations, near misses, or incidents should be communicated and handled in accordance to departmental and hospital policies.

Software Upgrades

Software upgrades usually provide new or improved functionality and/or fixes to known software defects. Most corrective actions to resolve software-related hazards (and sometimes hardware-related hazards) result in software upgrades. However, even a software upgrade intended to fix a problem can adversely affect other parts of the system. For example, an upgrade to an access point can cause medical telemetry dropouts.

To effectively deal with software upgrades, several technology management processes must occur, ranging from software version documentation to change management. Although documentation efforts within software version management may seem simple, they can quickly

become complicated. Consider medical devices that are composed of different modules, each containing a different software and/or firmware version. Similarly, consider how software-only medical devices are captured within an inventory record. Good tools and documentation are a necessity. Computerized maintenance management systems should have methods of capturing software versions and the history of software upgrades in a user-friendly, standardized, and searchable manner.[12]

Sometimes a healthcare facility will opt to avoid software upgrades to a particular class of medical devices, for example, to allow for data exchange between the medical device and different systems in the absence of good interoperability. In these cases, excellent documentation is needed to record the reasons for this decision. And, there may be the need to include information from the affected vendor(s) to detail any risks associated with this practice.

Change management ensures that alterations to a system, such as a software upgrade, are performed in a controlled manner. Because medical devices have varying levels of risk associated with their use, change management also needs to be risk-driven.[13] Since technology does not exist in a vacuum, human behavior also needs to be considered. If the upgrade provides new or improved functionality, training for users may be required to avoid or minimize unexpected user performance. If the new functionality will affect a clinician's workflow, an analysis of the downstream effect on related processes should be undertaken. And, as convergence grows and more interrelationships are formed, change management will need to assess the system-level effects of a software upgrade.

Convergence, Interoperability, and Software

When adopting converging technologies, interoperability—defined by IEEE as the "ability of two or more systems or components to exchange information and to use the information that has been exchanged"—becomes a common occurrence.[14] Interoperability and medical software are inextricably linked. The transfer of data is dependent on software, and any defect can directly affect patient safety. If we look only at PACS, several problems have occurred directly related to interoperability. For example:

- When zoomed computed radiographic images were exported to PACS, a system could make inaccurate measurements.
- A PACS software defect could result in the inaccurate display of heart wall motion abnormality scores from data from a cardiac ultrasound scanner.
- An anomaly between a modality and a PACS resulted in patient data being overwritten or matched to an incorrect patient.
- Sections of breast study images would not display on a PACS from certain modalities.[15]

> "Medical software demands the establishment of its own best practices and management strategies within hospitals.

As a result of the interdependencies of systems, healthcare organizations must be cognizant of the correlations and hierarchies between different types of medical software. How will a change, software upgrade, or new connection (and hence new software) affect the system as a whole?

Another aspect to consider is open communication standards. Such standards can help to prevent or mitigate interoperability-related problems when compared to proprietary protocols. Many medical devices still use proprietary communication methods, but there are initiatives under way to bridge the gap. But just because an open communication standard is used does not necessarily mean that seamless compatibility will exist. For instance, Digital Imaging and Communications in Medicine (DICOM) is an open communication standard used to exchange image-related data between PACS and modalities. Even though both a PACS and a modality may be DICOM-compliant does not ensure against incompatibility between the systems.[15] Detailed reviews of DICOM-conformance statements will likely help identify certain communication problems early on or before implementation; however, such an activity will rarely prove useful finding potential software defects.

Software Management: A Collaborative Effort

Medical software is a significant technology management issue. Software-related hazards and recalls, software upgrades, and interoperability are only a few of the areas that require attention. Cyber security, problem reporting, root-cause analysis, and the troubleshooting of software-related problems all need to be addressed by healthcare facilities.

Executive management must promote software reliability within their organizations and support departments such as the clinical engineering and IT departments, who are responsible for medical technology. However, other departments play a role as well, and their involvement

and positive effect on technology management should not be underestimated. Purchasing departments can include wording in request for proposals, purchasing contracts, and maintenance agreements that outline expectations for software upgrades and interoperability requirements, among others. There are industry initiatives that can assist purchasing groups such as the Medical Device "Free Interoperability Requirements for the Enterprise" which provides samples of contract wording specific to interoperability.[16]

Risk management, safety, and quality departments can promote best practices by including medical software into their activities, including root-cause analyses and failure modes and effects analyses. There are guidance documents available that are informative, but most are written primarily for medical device manufacturers.[1,2,3,17,18] The upcoming IEC 80001-1 standard *Application of risk management for IT-networks incorporating medical devices—Part 1: roles, responsibilities and activities* is aimed at healthcare facilities, and it may also prove a useful tool.[19] The standard centers around the risk management activities and responsibilities needed when medical devices are placed on the hospital network to balance priorities such as the safe and effective use of medical devices and data and system security. Its approach will apply not only to the initial incorporation of medical devices onto a hospital network, but also to the ongoing needs of the technology such as software upgrades and cyber security—similar challenges to that of medical software.

Clinicians play a critical role as they help to define the requirements of medical technology. They are also on the front line when a problem occurs. Good reporting systems, good relationships with clinical engineering and IT, and a safety culture are vital to the sustained health of medical technology, medical software, and ultimately the patient.

Medical software has a commanding presence in healthcare, and its impact will only grow as technology becomes more sophisticated and as systems become more interconnected. Medical software demands the establishment of its own best practices and management strategies within hospitals. Failure to do so could result in operational inefficiencies, or worse, harm to patients. ■

References:

1. U.S. Food and Drug Administration. General principles of software validation; final guidance for industry and FDA staff. January 11, 2002. Available at: http://www.fda.gov/MedicalDevices/DeviceRegulationandGuidance/GuidanceDocuments/ucm085281.htm. Accessed July 31, 2009.
2. U.S. Food and Drug Administration. Guidance for the content of premarket submissions for software contained in medical devices. May 11, 2005. Available at: http://www.fda.gov/MedicalDevices/DeviceRegulationandGuidance/GuidanceDocuments/ucm089543.htm. Accessed July 31, 2009.
3. U.S. Food and Drug Administration. Guidance for industry, FDA reviewers and compliance on off-the-shelf software use in medical devices. September 9, 1999. Available at: http://www.fda.gov/MedicalDevices/DeviceRegulationandGuidance/GuidanceDocuments/ucm073778.htm. Accessed July 31, 2009.
4. U.S. Food and Drug Administration. Devices: general hospital and personal use devices; reclassification of medical device data system. Federal Register. 2008 Feb 8;73(27):7498-503.
5. **Kimberland K.** Microsoft's pilot of TSP yields dramatic results. news@sei. 2004;2. Available at: http://www.sei.cmu.edu/news-at-sei/features/2004/2/feature-1-2004-2.htm. Accessed July 27, 2009.
6. **Rakitin S.** Applying system and software engineering practices to clinical engineering. Presented at: AAMI 2009 Conference and Expo; June 6, 2009; Baltimore, MD.
7. **Schrenker R.** What you should know about software engineering. *Biomed Instrumen Technol*. 2005 Sep-Oct;39(5):353-6.
8. U.S. Food and Drug Administration. Glossary of computerized systems software development terminology. August 1995. Available at: http://www.fda.gov/ICECI/Inspections/InspectionGuides/ucm074875.htm. Accessed July 31, 2009.
9. ECRI Institute. Coping with convergence: a roadmap for successfully combining medical and information technologies [guidance article], *Health Device*s. 2008 Oct;37(10):293-304.
10. **Bates DW, Gawande AA.** Improving safety with information technology. *N Eng J Med.* 2003 Jun;348(25):2526-34.
11. ECRI Institute. Health Devices Alerts [database online]. Available at: https://members2.ecri.org/components/alerts/Pages/CPIssues/Issue.aspx?CH=1&ChName=Medical%20Devices&rid=0. Accessed July 27, 2009.
12. **Majchrowski B.** Dialysis, hazards, and medical device software: then and now. *Biomed Instrumen Technol*. 2009 Jan-Feb;43(1):59-62.
13. **Sloane E.** Tackling change management 24x7. October 2008. Available at: http://www.24x7mag.com/issues/articles/2008-10_07.asp. Accessed July 31, 2009.
14. ANSI/IEEE Std 100-1984 IEEE *Standard Dictionary of Electrical and Electronics Terms.* 3rd ed. New York, NY: The Institute of Electrical and Electronics Engineers, Inc; 1984.
15. ECRI Institute. Data-transfer problems between imaging devices and PACS could result in misdiagnosis [hazard report]. *Health Device*s. 2008 Dec;37(12):381-3.
16. Medical Device "Plug-and-Play" Interoperability Program. Medical device interoperability for patient safety: driving procurement changes. October 2008. Available at: http://mdpnp.org/MD_FIRE.php.
17. AAMI. Medical device software risk management [technical information report]. AAMI TIR32:2004.
18. AAMI. Validation of software for regulated processes [technical information report]. AAMI TIR36:2007.
19. International Electrotechnical Commission. IEC 80001-1 Application of risk management for IT-networks incorporating medical devices–Part 1: Roles, responsibilities, and activities. 2009-07-31 [draft standard].