Medical device software: Why is it in critical condition?

Jim McElroy, LDRA

Requirements traceability eliminates gaps in embedded software repeatability between the development and maintenance phases of medical devices.

When it comes to medical devices, manufacturers face conflicting demands. Devices need to provide the expected performance. They need to be quick to market and cost effective. Above all, however, when human health and safety are at stake, they need to be reliable. From the business perspective, manufacturers simply cannot afford a device recall never mind the negative impact this has from a publicity perspective.

As with most safety- and security-critical designs, hardware reliability is not enough to guarantee the safety and effectiveness of the device. The software needs to be properly architected, coded, and tested to deliver accurate and repeatable performance.

Given the complexity of today's designs, achieving this level of quality with new products that don’t have development legacy and history is difficult enough. The challenge is compounded as devices are updated and functionality is added to previous designs. AnFDA analysis of more than 3000 medical device recalls found that 79% of software-based incidents proved to be the result of defects introduced by changes and updates.

Ultimately, the cause of many problems comes down to a single factor: repeatability. The key to eliminating device problems is to analyze the medical device engineering process and determine where the gaps in repeatability occur. These gaps present enormous financial risks to the suppliers of these devices, as well as health and safety risks to patients. Lifecycle traceability, as facilitated by an automated matrix of relationships, provides an effective and efficient method to mitigate these risks.

House of mirrors

A colleague of mine once said, “In the course of every project, it seems, the time comes when you have to shoot the engineer and ship the radio.” Especially with the increasing complexity of embedded devices, product development is all too often left to the wizardry of a few key people, after which the development process is considered history. Moreover, once the “radio” is shipped, the project transitions into maintenance and the product is left to a team without a usable, repeatable process. In such an environment, it's no wonder that updates and changes lead to product failures and recalls.

Product development and maintenance shouldnot be distinct activity silos. They should be integrated.A very important aspect of the IEC 62304 medical standard is that the maintenance process is the mirror image of the development process (see figure 1). Each activity depicted in these process models is based on an objective that states what the activities should accomplish. The resulting activity produces an asset, the specification of a high-level software requirement. This asset next must be validated based on the set of objectives, such as developing high-level requirements for software that comply with system requirements or developing high-level requirements for software that are accurate and consistent.

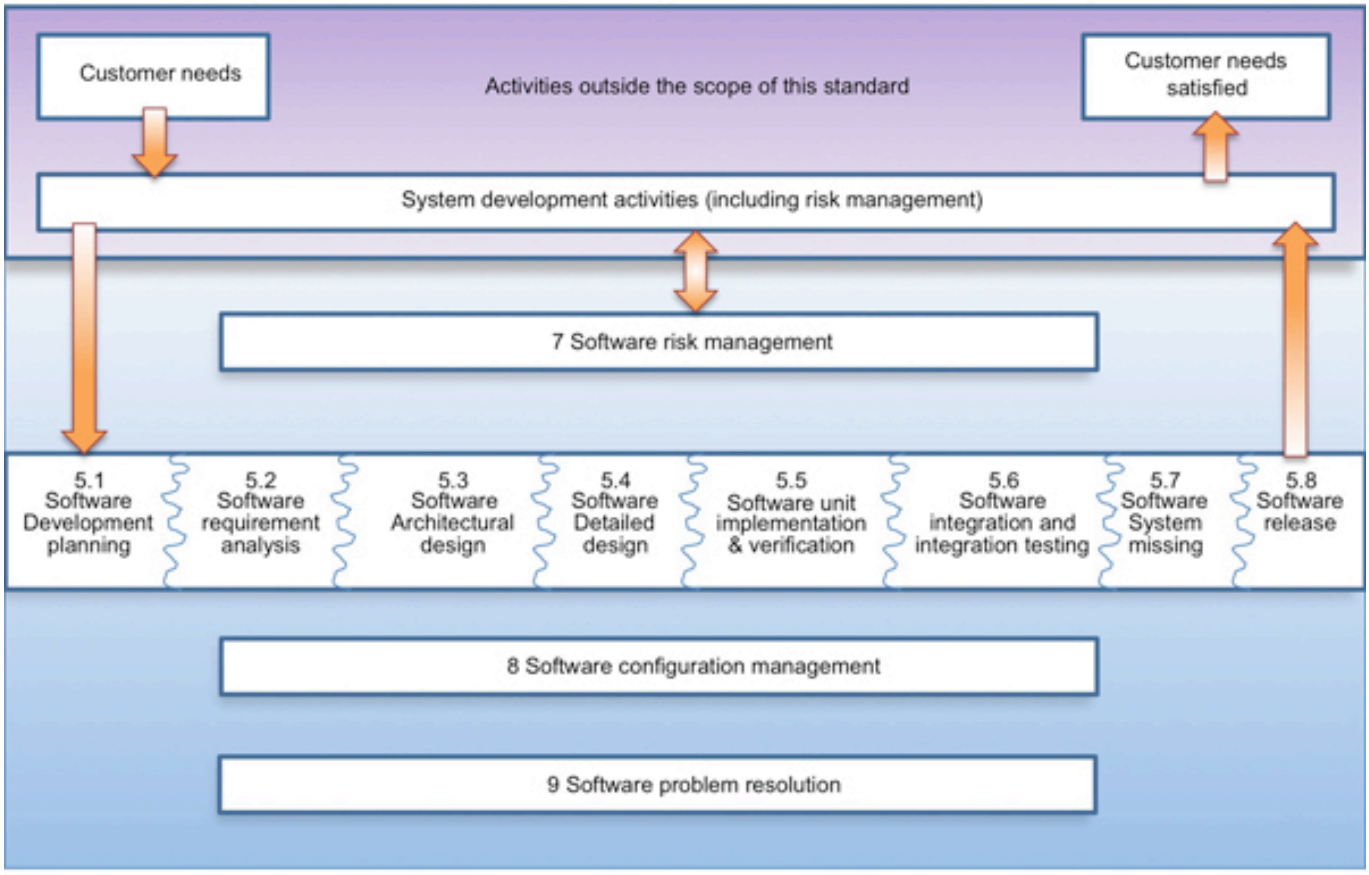

Figure 1. IEC 62304 Development Process Model defines a development process (top) that is the mirror image of the maintenance process (bottom).

Complying with objectives produces many other assets, for example software risk analysis or low-level software requirements. Assets and associated artifacts are what constitute compliance substantiation for meeting objectives, which in turn becomes the basis for product clearance/approvals and in the future hopefully certification. Making this a repeatable process is instrumental to the success of any device manufacturer.

Requirements engineering

Elements such as objectives and assets constitute the requirements for a software development project. Using a technique known as requirements engineering, developers can define these aspects up front and track these elements throughout the development and maintenance process, an approach that ensures repeatability even in the wake of changes and upgrades at any point in the product lifecycle.

The Product Realization Process defined by the Global Harmonization Task Force[1] calls for a Requirements Engineering (RE) process that includes not only the formation, specification and validation of requirements, but also implicitly entails risk management (RM). RM and its related requirements should be based on actual or possible risk events and prevent them from recurring. Accomplishing this goal requires accurate reporting, consistent analysis, and the ability to reestablish the context in which the software was originally produced. These are the characteristics of a repeatable process.

In the context of a product lifecycle, RM is effectively escalated into a System Engineering process. For complex embedded systems, both hardware and software might perform to specification, but in the process of integration, system behavior can go askew. Such anomalies can only be properly evaluated, remediated, and prevented from recurring in the context of a highly repeatable process.

Requirements traceability

The mechanism by which a project can achieve this repeatability is facilitated by a properly structured Requirements Traceability Matrix (RTM). An RTM maintains active links between all product assets and associated validation and verification data. The relationships in the RTM are many-to-many, or N-to-N. These relationships (or links) must be continuously monitored to maintain the viability of the RTM. This monitoring generates a notation called a Suspect Link if a proximate link is altered or removed.

Requirements traceability also demands that its links be bi-directional. For example, any given function in the source code should be automatically linked with its requirement(s) and the validation data, including the risk analysis and the test cases that verified it.The engineers who contributed to its design and implementation shouldalso be identified. Consequently, a by-product of the RTM is an incisive impact analysis for any given proposed change to the source code function.

The core of requirements traceability is to prevent exactly the kinds of errors in updates that lead to the recalls we discussed at the outset. Clearly, the RTM is an essential tool. An astounding reality in most medical device and pharmacological projects, however, is that the RTM function (or its equivalent), relies on an Excel spreadsheet or a manual entry database program to trace the relationship links. A few projects may add a requirements management tool to the mix, producing the resulting dilemma of trying to link two information silos. In either case, these non-integrated, manual approaches create the opportunity for key omissions or errors, which works against the overall goal of repeatability.

It is important to have a viable solution to meet the enormous challenges of establishing a repeatable process and assuring the integrity of the RTM. A purpose-built tool can perform the task effectively and efficiently, ultimately saving in development time and product performance over the long run.

Arguably, the major problem in developing medical device software is characterized by a lack of symmetry between development and maintenance phases of a product

lifecycle. This compromises the ability of a team to effectively incorporate customer feedback, especially from operations such as product trials, and predict the reliability of newly defined and implemented product solutions. Moreover, this lack of repeatability prevents medical device suppliers from realizing potential profits from product variants and extensions through such innovations as product line development and requirements reuse. Requirements traceability provides a key element to a repeatable high quality solution, enabling developers to assess, monitor, and track the effects of changes and deliver reliable products to market efficiently and economically.

References

1. 11th Conference of the Global Harmonization Task Force (2007).

About the author

Jim McElroy, Vice President of Marketing at LDRA Technology, is focused on expanding LDRA business in the embedded software verification market by improving developer productivity and software quality in critical application development. Before joining LDRA, McElroy held executive-level marketing and business development positions with Green Hills Software, Telelogic North America, and I-Logix as well as holding industry-level software development positions at Lockheed Martin and Raytheon. McElroy has a Master of Science in Computer Science from Fitchburg State College and a Bachelor of Science in Computer Science from the University of Massachusetts.

Source URL (retrieved on *09/14/2014 - 11:49pm*):
http://www.ecnmag.com/articles/2012/10/medical-device-software-why-it-critical-condition?qt-most_popular=0