

A software prescription

Developing safe and effective software for medical devices.

By **Steffan Benamou** and **Jim McElroy**.

Medical devices are growing in complexity as the market demands more feature rich and cost effective solutions. As complexity increases, device manufacturers need to reduce development time, while creating a safe and reliable product.

The endoscopic medical device market is an example; if endoscopic surgery can be performed more quickly, the patient is likely to recover faster and surgeons and staff can perform more procedures in the same period of time.

In order to achieve these higher level goals, device manufacturers need to use the right software development tools in conjunction with a reliable and secure operating system (OS).

The challenges involved in developing safe and effective Class II and Class III medical devices (see box) in a timely fashion are wide ranging, with concern for the patient and the device operator at their heart. And developers face the conflicting pressures of time to market windows, competition, development costs, supplier costs and regulatory compliance issues. To cope with these pressures, medical software developers are moving to industry proven development tools, operating systems and software development practices.

Until 2009, most medical device software was developed using tools and processes that may not have been optimised for the application. However, the IEC 62304 standard provides a framework of life cycle processes, activities and tasks for the design and maintenance of medical device software (see fig 1). Companies like Stryker Endoscopy are using industry standards such as



IEC62304 to aid software development and maintenance in order to create safe and effective devices as quickly as possible without sacrificing product quality or patient and operator safety.

A few core elements contribute to the successful development of complex medical devices. The first step is the effective use of standards but, in addition, the proper selection of hardware and OS technology is fundamental to the success of the project as a whole.

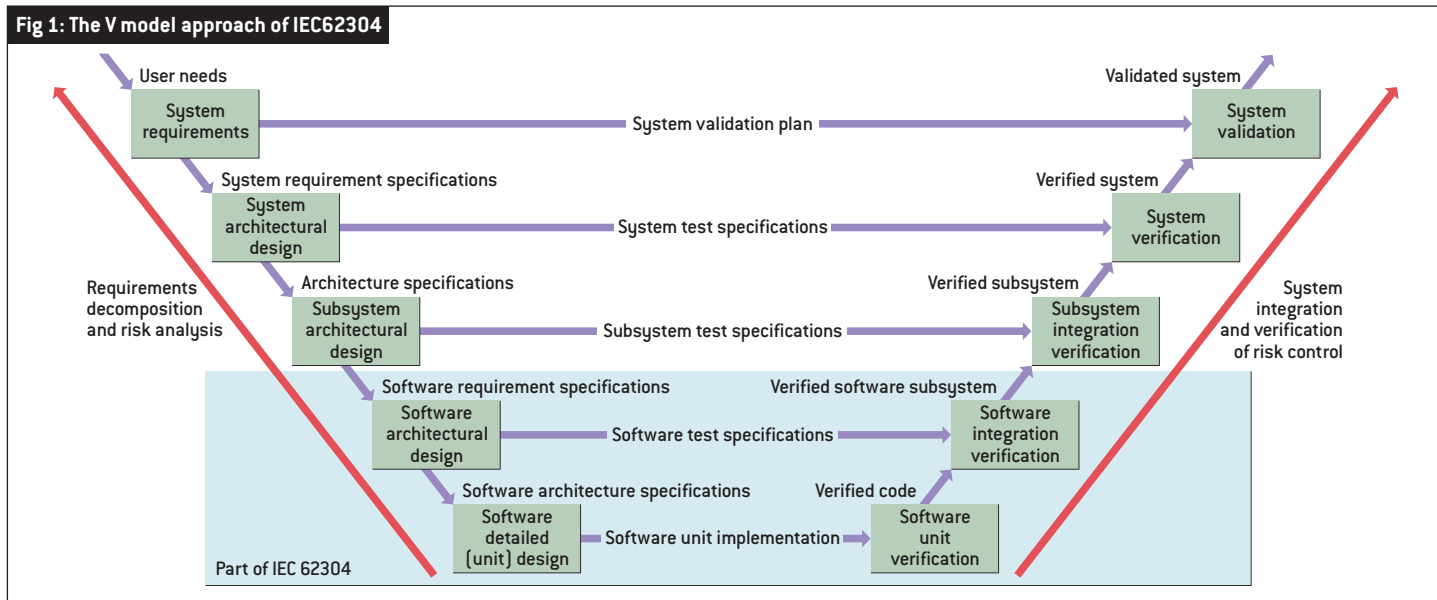
Stryker chose Green Hills Software's INTEGRITY real time operating system and, because software applications above the application layer are complex, the team used MULTI software development tools to create, test and verify the applications to ensure patient safety.

In addition to providing a solid framework for software development, IEC 62304 implicitly enables proper architectural software design by forcing

the development team to 'safety classify' all software items. Software items are classified as 'A', 'B' or 'C', based on their potential hazardous effect on the patient or operator, with 'C' having the most detrimental effect. Using these classifications, the development team can conceptually separate critical and non critical applications. This separation can be realised at run time by using the OS's separation kernel architecture to partition the various applications. As a result, the system is safer and more reliable, since a failure in one partition cannot affect an application running in another. In addition, multiple applications can share the same processing resource, saving money on the overall bill of materials by eliminating the need for separate physical hardware resources.

For this architectural design, IEC 62304 helps break the system down into software units, better preparing the

Fig 1: The V model approach of IEC62304



system for unit and system verification. It is important to note the enabling OS technology also provides a clean way for applications in separate address spaces to better communicate with each other.

The chosen OS uses a true real time scheduler that supports multiple priority levels. This allowed a rate monotonic algorithm to be used to assign priorities to all the tasks, making the most of the ability to schedule the system and ensuring critical time constraints were met. In addition, the team used the OS' partitioning architecture to create a governing health monitoring application.

Through the interaddress space communication platform provided by the OS, the health monitoring application could communicate with all running applications to keep an eye on the status of critical and non critical tasks in the system. Faults can be checked for continuously and, if necessary, critical tasks can be put into safe states.

The team also used partition separation to create an address space dedicated to handling communication to and from other Stryker devices in the operating theatre. Using a proprietary interface bus, Stryker products can communicate with each other to allow for seamless integration, data sharing and universal control from a central location.

By implementing this as a partitioned address space, software and hardware become modularised and can be ported to other devices that may be developed in the future.

Code quality is influenced by many factors, including the quality of the engineers writing the code, the development tools and the development practices followed. Under IEC 62304, each unit has its own verification process. In accordance with best practice, the team used a static analysis tool throughout software development, which eliminated a number of obscure bugs. Examples of such bugs included buffer overflows, resource leaks, and NULL pointer dereferences.

Moving forward, the team is using the IDE's static analysis tool, which provides tight coupling with the debugger. This provides the ability to perform static analysis automatically during compile time, instead of performing it manually later. While static analysis tools contribute to higher quality code, they are not sufficient for ensuring the quality of the application.

The design team also used the IDE's code profiler tool, which gives a report of line by line code execution coverage, specifying which lines of application code have been exercised. The team used this

Medical device classification

The US Food and Drug Administration defines Class II devices as those for which general controls alone are insufficient to assure safety and effectiveness, but which should perform as indicated without causing injury or harm to patient or user. A Class III device needs premarket approval through a scientific review to ensure the device's safety and effectiveness. Class III devices are usually those that support or sustain human life, are of substantial importance in preventing impairment of human health, or which present a potential, unreasonable risk of illness or injury. In the EU, four classes of medical device are defined – I, IIa, IIb and III. Performance of all devices except those in Class 1 must be verified by a Certificate of Conformity issued by a Notified Body.

to design unit tests that exercised all elements of the application code.

All documentation is accessible and generated automatically. As is typical in embedded software development, just a few bugs take the most time to fix. For difficult situations, the debugging environment enabled the team to resolve issues related to task priorities, as well as low-level issues.

The Stryker medical device was developed using Green Hills' Platform for Medical Devices, which features the INTEGRITY real time operating system. For code creation and debugging, the team used the MULTI integrated development environment with Green Hills; compiler technology, EventAnalyzer, TimeMachine debugger and the SuperTrace probe. The environment enabled the team to reduce development time significantly while increasing the efficiency of the review and approval processes. The fielded medical device itself is likely to enable surgeons to perform higher quality procedures in a more timely fashion.

Author profiles:

Steffan Benamou is a staff engineer with Stryker Endoscopy. Jim McElroy is director, industry business development, with Green Hills Software.

www.stryker.com www.ghs.com

