IBM Software
Life Sciences

Thought Leadership White Paper

Challenges and opportunities for the medical device industry

Meeting the new IEC 62304 standard

IBM

Contents

Executive summary

The recent IEC 62304 standard for medical device software is causing companies worldwide to step back and examine their software development processes with considerable scrutiny. While software development and testing is still an integral part of overall system design and production, the IEC 62304 standard focuses on software as a separate life-cycle process with specific needs for risk management and safety assessment.

With IEC 62304, the world has changed—country by country—for medical device manufacturers. This doesn't mean, however, that complying with IEC 62304 must slow down your medical device software development. By applying best practices guidance and process automation, companies have a new opportunity to improve on their fundamental business goals, while getting through regulatory approvals faster, lowering costs and delivering safer devices.

This paper will explore what IEC 62304 compliance means for manufacturers in some detail, and also describe the larger context of systems and software engineering best practices at work in many of today's most successful companies.

Changes in the medical device field

The IEC 62304 standard points to the more powerful role that software plays in the medical device industry. Once hardware was king. Older systems used software, of course, but it was not the main focus, and there wasn't much of a user interface. Software was primarily used for algorithmic work. Not to overly generalize, but the focus of management was on building hardware that worked correctly; software was just a necessary element of overall implementation. Now with complex UIs, easy-to-use-at-home medical devices and systems, etc., software has taken on a much more vital role.

Today, software is clearly where the consumer value lies. The embedded software is creating competitive differentiation for manufacturers. Thus, software developers are beginning to play a greater role in the design, architecture, and functionality of medical devices. Tried and true software development methods are also being woven into the overall product engineering process. And the more flexible (some might say "agile") those methodologies are, the better. That's because requirements change more fluidly than ever before. Rigid requirements are a thing of the past. While still widely used, waterfall methods can no longer simply freeze requirements, create the corresponding code, then test against the requirements and hope (often optimistically) that the finished product maps to early expectations. These methods must recognize that changes occur to requirements over the course of the life cycle, and adopt the appropriate levels of flexibility in order to cope.

Rapid changes in supply chains and changes in requirements midstream during a given product delivery cycle are causing some degree of chaos for engineering teams, including software development teams. Now, the IEC 62304 standard requires traceability (the ability to ensure requirements map to each element in the software code) in the software delivery process. But as software teams working in this bold new world seek to adopt agile techniques to better meet deadlines and expectations, they are struggling to embrace the demand for traceability, which is simply not the "sweet spot" for agile development.

So the question is, how do we introduce more rigor to the development process and meet the needs of this new industry standard?

What is so hard about software?

Coordinating people, processes, and tools is never easy. So why does a greater role for the software component make the effort even harder? The primary difficulty has to do with the nature of software itself. While the best thing about software is that it is soft (i.e., relatively easy to change), this is also its riskiest attribute. Unlike bridge construction, most software does not deal with natural phenomena where laws of physics or materials provide a well-understood framework. Instead, most software is constrained only by human imagination. The quality of software is judged not by precise mathematics and physical tolerances, but by the degree to which it satisfies a user's expectations, which can be highly subjective.

For these reasons, agile and iterative delivery methods enforce frequent stakeholder review of the working software under development, which helps guide software projects toward more satisfactory outcomes. But, as noted earlier, software practitioners following the more "pure agile" methods—such as Scrum or XP—may find that the new emphasis on requirements in IEC 62304 demands a more formal, less "agile-feeling" life cycle. Pure agile methods need to be scaled up with additional guidance from configuration and requirements management practices, at a minimum. Modeling your architecture is another great way to scale up your process to the requirements expected of IEC 62304.

The good news is that this standard doesn't require a specific software development and delivery process. You can use your traditional "waterfall" methods if you like; you can use the Unified Process (UP) or one of its flavors, such as the IBM® Rational® Unified Process (RUP); or you can tailor your agile development practices to conform to IEC 62304 requirements.

The following section discusses the best new approach for handling the software component in modern medical devices, especially in light of the IEC 62304 standard.

Disciplined agile delivery: Flexibility with more structure

As many agile software development methodologists are quick to point out, agile techniques were never meant as an excuse to be undisciplined. Beginning with the agile manifesto in 2001, the emphasis on results over process does not mean that following the process is unimportant; rather it simply means that agile teams keep the ultimate goal in mind, using frequent iterations (stages during which functioning software can be tested and demonstrated to stakeholders) and frequent course correction as the project proceeds. At the same time, some of the "high ceremony" work flows used in more traditional methodologies—such as architecture, configuration and change management, and traceability—were set aside as cumbersome sub-processes not needed by small, co-located teams working closely on an hourly basis.

Over time, these agile methods—XP and Scrum are two of the best known—proved successful, and more traditionally minded software development organizations began to take notice of the results that agile teams were achieving. Yet the problem was clear: how do teams that need a little more discipline (for a variety of reasons such as culture, service level agreements, compliance, safety, and contract restrictions) adopt agility to achieve these same results? In many cases, these are larger teams who must manage the scale of operations defined by a geographically distributed workforce.

Enter Disciplined Agile Delivery (DAD). At its essence, DAD provides teams of any size many of the benefits of traditional methodologies while retaining the results-oriented spirit of agile development. Of course, it will not always be the case that a development team working on the software components for a medical device will be a large team, but the traceability required by the IEC 62304 standard demands more than a purely agile approach, and DAD offers a solution. A full discussion of DAD[1] is beyond the scope of this paper. But suffice it to say that the overall quality management concerns introduced by the new the IEC 62304 standard are addressed by DAD.

Meeting the specification: A look at some of the details

Traceability, reporting, architecture, requirements management—all these "high ceremony" attributes of more traditional iterative processes can be applied to a medical device

team's agile software methods without slowing down the project. This additional rigor is what the IEC 62304 standard demands. The real question becomes: what key elements do you need in any process, and how are those elements linked together? This section will discuss some of those elements. This is not a comprehensive list, but it covers several of the essential requirements and how they affect software delivery teams. For a complete list please refer to the actual IEC 62304 document.

The IEC 62304 standard states that requirements analysis must be part of the software development process. The requirements analysis discipline establishes the high-level requirements of the system being designed, then derives lower-level requirements from those until the process produces requirements with sufficient information to enable coding; these lower-level requirements detail the complete system, including potential faults and interfaces between systems. These can be developed iteratively, in an agile process, but the IEC 62304 standard demands that they must be documented. Requirements also need to link to other phases of the process, including the software architecture, test cases, and so forth. The general idea is that someone can look at a requirement and understand what should be implemented and what tests must be performed to prove the requirements are met. Requirements typically are written by systems engineers as simple text early in the design process as they capture ideas on paper.

Another discipline required by an IEC 62304 guided process is architectural design. This turns the requirements into a coherent architecture so developers can understand how the requirement will be met and ensure there aren't any overlaps or holes in the requirements as described. Graphical images often are used to help development teams visualize an architecture emerging from the requirements. The graphics should map to the actual code, which provides the means for traceability from requirements all the way to the code.

A key part of the overall process is failure mode and effect analysis (FMEA). FMEA is a powerful tool for explaining the potential failures to a regulatory agency. It shows how the identified failure points map to the requirements and the tests that need to be run in order to prove that the failure can be handled correctly. Fault Tree Analysis (FTA) is a graphical way to help analyze the system to see where a failure is likely to occur. Typically used during the early analysis phase of development, FTA diagrams show how failures interrelate. FTA diagrams combined with FMEA create a comprehensive strategy for identifying, understanding, and tracking potential failures.

Another important discipline is testing. The IEC 62304 standard discusses both integration and system testing. Integration testing can help ensure that different components actually work together and do not cause unanticipated behaviors. System testing treats the whole system like a black box and helps ensure that high level requirements are met by the system itself. Each testing discipline is critical for meeting the requirements of IEC 62304. They also are quite useful for ensuring that your device works as expected.

While developing reports isn't specifically listed as an IEC 62304 requirement, in the end a report is what needs to be sent to the regulatory agencies. That report needs to contain all the information listed above and explain how they trace between each other to make a comprehensive whole software system for the medical device.

Process steps for IEC 62304 compliance

The techniques described at the end of the previous section are facilitated through software development tools specifically geared for systems delivery, which is often divided into multiple categories: systems engineering, project management, software engineering, and test management, for example. These categories ideally interconnect across the systems delivery life cycle in performing distinct tasks and creating distinct work products.

Typically, hardware and software teams today use similar processes for development. The standard V diagram in Figure 1 shows the typical stages both teams use for analysis, design, implementation and testing. The challenge is that the teams operate separately, limiting their ability to synchronize key steps. What's more, alignment is hindered by the use of different languages and tools. To achieve rapid co-development and the associated business benefits, the hardware and software processes need to be integrated into a unified process.

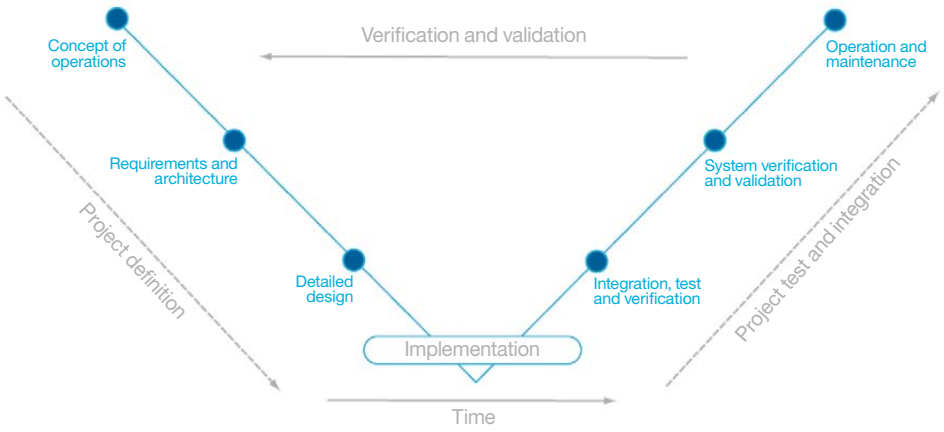

Figure 1: The typical stages both software and hardware teams use for analysis, design, implementation, and testing

This general, ideal process illustrated in Figure 1 provides a kind of map for software and systems development teams to address the needs of a standard such as IEC 62304. Some of this process may be part of your current methods; some portions may be clearly missing; and some may be unclear. The next section describes how to go about your own gap analysis to determine where you need to improve to meet the standard.

Performing the gap analysis

IEC 62304 compliance does not need to slow down your medical device software development. But we do recommend that you perform a gap analysis to see how closely your own process maps to the specifications of IEC 62304. You may eventually hire a third party to help you come into compliance with the new standard, but you should do the gap analysis first, just to take an inventory of your process and its typical artifacts. You may find that your documented process isn't the one you actually follow in practice.

Get started by examining your process on paper. Of course, having a consistent process across groups is helpful for management to understand what is happening. But is this process truly the one you are following? Be honest with yourself, and determine if your departures from the stated process occur across all products, or only some. Try to determine where your software tools are most helpful—where the integrations are strong and consistent across the entire organization; and identify the areas for improvement.

If you aren't meeting your on-paper process, it could simply be that it's good in theory but difficult in practice. Compare both to what the standard requires. Do you follow all the phases of the standard? Do you have the required traceability between the phases?

For example, IEC 62304 standard demands complete traceability. Some organizations use a waterfall process, with traceability only on paper linking requirements to specific locations in the code. This type of tracing can become out of sync with the actual code because, as discussed earlier, software code is by nature "soft" and evolves as development goes on. Therefore, you need some requirements management method that maintains a link to the actual code that addresses each requirement, not simply a line number or some other meta reference.

In the classic waterfall process, requirements are created first, and coding is expected to follow the requirements to the letter. But you need to know how closely the actual code maps to those requirements. Your team may have improved on this mode of traceability with an agile process, which can demonstrate actual requirements being met through frequent testing. Tools with the ability to link through the architecture and support in context help solve this problem, and there are a variety of approaches you can adopt through tool and practice implementation.

You can yourself, in most cases, undertake an initial comparison between IEC 62304 specifications and your actual practices. If necessary, you can then bring in a third party to review the whole process and obtain recommendations for closing any gaps discovered. A third party may recommend changes in your process (even simplifications) as well as recommend specific tools that a) offer strong collaboration across the software development life cycle, and b) are specifically geared to support software and systems delivery.

It is up to your organization to decide what works best, and changes should be carefully considered both in light of the IEC 62304 specification as well as your product delivery culture and history of success. The best tool solutions will be those that assist with incremental adoption of new capabilities to help you avoid wholesale process changes and massive new infrastructure investments.

Let's consider IBM's approach to tooling that can help address the critical needs for IEC 62304 compliance.

Choosing software tools for IEC 62304

With deep expertise in the design and deployment of embedded software throughout the systems landscape, the IBM Rational organization offers a proven solution to the tooling needs of medical device manufacturers. While a combination of tools from other vendors and open source providers can potentially address some of the needs of software teams working in the medical device arena, we will present in this section the advantages of IBM's comprehensive approach.

IBM Rational Solution for Systems and Software Engineering

Without integrations across the system delivery life cycle, systems software teams are left to operate in silos. When silos form, product delivery effectiveness suffers. In order to deliver smarter medical devices that respond to changing market needs and standards, systems and software engineering teams must perform efficiently and manage all the life-cycle work products through collaboration. Figure 2 shows how the IBM Rational Solution for Systems and Software Engineering provides this integrated system life cycle management solution. (Note how the arrangement of specific product capabilities in Figure 2 largely instantiates the workflow depicted in Figure 1.)

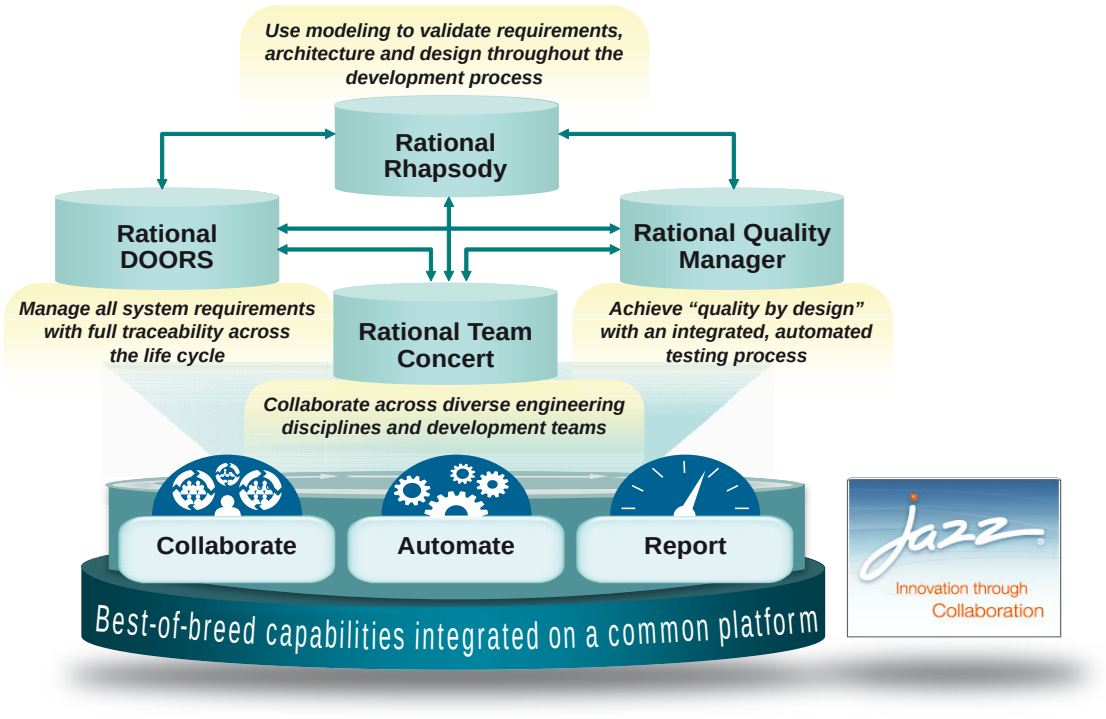

Figure 2: IBM Rational Solution for Systems and Software Engineering

The IBM Rational Solution for Systems and Software Engineering offers a comprehensive life cycle management platform that supports collaborative tasks and helps link the various artifacts developed over the course of the product life cycle. This solution also enacts system delivery workflows and provides task management capabilities to effectively run the system delivery project, and it is enhanced with support for safety, reliability, and security analysis, mapping to CMMI maturity levels[2], and supports medical standards IEC 61508 and IEC 62304.

Using the specific product sets illustrated in Figure 2, the IBM Rational Solution for Systems and Software Engineering provides an integrated method for the following systems and software engineering team roles:

- **Systems engineers.** The IBM solution provides an integrated and collaborative environment for requirements analysis, architecture management, and work, change and configuration management for teams of system engineers. The leading products are IBM Rational DOORS® software and IBM Rational Rhapsody® software for system engineering tasks, integrated with IBM Rational Team Concert™ software for life cycle management of the work products. The integrations with IBM Rational Quality Manager software provide for strong collaboration with system validation teams from the start of the project.
- **Safety engineers.** This group focuses on safety requirements and assurance. The primary products are Rational DOORS software and Rational Rhapsody software with its Safety Analysis Profile for identifying and classifying hazards, faults and safety measures.
- **Reliability engineers.** This group focuses on the reliability of the system as measured through metrics such as MTBF and availability. The primary products used here include Rational DOORS software and Rational Rhapsody software, possibly in addition to some spreadsheet templates provided with the IBM Rational Harmony™ process content.
- **Project, development and test team leads.** Rational Team Concert software and Rational Quality Manager software provide work and plan management for system delivery teams across the project life cycle and help with live transparency through collaboration, automation, and reporting to the system delivery work products and project health.
- **Software engineers.** With Rational Rhapsody software integrated with Rational Team Concert software in the Eclipse IDE, the IBM solution provides a software development solution for software engineers. This integrates model driven development using UML with the Rational Team Concert software capabilities for team collaboration, such as model configuration management, work items, change sets and continuous software build support. This solution also provides traceability to upstream system engineering work products in Rational DOORS software and Rational Rhapsody software, or downstream traceability to system integration and validation.
- **Software and system testers.** Rational Quality Manager software provides a collaborative environment for test planning, construction and execution supporting continuous testing as part of the software engineering teams, as well as test management of system validation and acceptance testing. IBM Rational Test Lab Manager software can help improve the efficiency of system test labs and manage how resources are requested and provided.

Integrations help provide collaboration and traceable requirements

The IBM Rational Solution for Systems and Software Engineering is integrated on the IBM Jazz™ platform for collaborative software delivery. Uniquely attuned to global and distributed teams, Jazz can help transform software delivery by making it more collaborative, productive, and transparent. You can think of Jazz as an extensible framework that dynamically integrates people, processes, and assets associated with software projects. Unlike the monolithic, closed solutions of the past, Jazz is an open platform that supports the IBM Open Services for Lifecycle Collaboration (OSLC) initiative[3] for helping improve

tool interoperability. Products built on the Jazz platform can leverage a rich set of capabilities for team-based software and systems delivery.

Rational DOORS software provides a comprehensive requirements management capability for the IBM Rational Solution for Systems and Software Engineering. It manages stakeholder requirements, system requirements, and decomposed subsystem requirements that engage stakeholders, system engineers, software engineers and testers in a collaborative requirements process.

Together with the Rational Rhapsody and Rational Team Concert applications, Rational DOORS software supports the system engineering teams to capture and link system and stakeholder requirements. Integrating with Rational Rhapsody software, the system requirements are linked to elaboration and specification of use-case and executable requirements models.

Rational Publishing Engine and Rational Insight software

The IBM Rational Publishing Engine application automates the generation of documents, formal reviews, or regulatory compliance and helps to improve productivity and reduce risk and cost. Rational Publishing Engine software integrates with the IBM Rational Solution for Systems and Software Engineering and generates composite documents from the system life cycle repositories. This powerful facility allows you to create custom reports for submission to certification agencies. In essence, the Rational Publishing Engine software can make life easier for development teams, especially to sell across different countries and regulatory agencies.

IBM Rational Insight software adds performance measurement and analysis to the management and reporting solution, which helps improve process performance through reports and dashboards.

Conclusion

A developer immersed in the details of code recently written may have a very clear sense of how that code addresses a specific requirement. But even brilliant code that is not well-documented won't meet the specifications of IEC 62304, since a new level of traceability between requirements and code is now demanded. Yet, your brightest developers may detest the need to demonstrate this traceability, since it has little to do with the ingenuity they have brought to bear on their various coding assignments. That's why it is vital for your tools themselves to show these connections automatically. This can alleviate manual reporting and dramatically reduce the possibility of human error that invariably is part of a manual process.

An improved process, based on the principles of agile development and the added strengths of modeling, architecture, and sophisticated requirements management capabilities, gives your teams the process rigor needed to meet the IEC 62304 standard. With automated reporting, this process also makes it far easier for your sales teams or those in charge of handoffs within the supply chain to demonstrate IEC 62304 compliance.

For more information

To learn more about IBM Rational Solution for System and Software Engineering for the medical device industry, please contact your IBM marketing representative or IBM Business Partner, or visit the following website: **ibm.com**/software/rational/solutions/healthcare/

Additionally, financing solutions from IBM Global Financing can enable effective cash management, protection from technology obsolescence, improved total cost of ownership and return on investment. Also, our Global Asset Recovery Services help address environmental concerns with new, more energy-efficient solutions. For more information on IBM Global Financing, visit: **ibm.com**/financing

About the author

Martin Bakal is a senior business development manager with IBM, specializing in embedded devices for a variety of industries. He was formerly with the Modeling Division of Telelogic prior to their acquisition by IBM. He has a BS in electrical engineering and an MS degree in engineering management, both from Tufts University.

[1] For this full discussion, see Ambler and Lines, "Disciplined Agile Delivery: An Introduction," **ibm.com**/common/ssi/cgi-bin/ssialias?infotype=SA&subtype=WH&appname=SWGE_RA_ZV_USEN&htmlfid=RAW14261USEN&attachment=RAW14261USEN.PDF

[2] For more information on the Capability Maturity Model Integration, see the Software Engineering Institute's website at http://www.sei.cmu.edu/cmmi/

[3] For more on OSLC, see the white paper "The business value of open collaboration" at **ibm.com**/common/ssi/fcgi-bin/ssialias?infotype=SA&subtype=WH&appname=SWGE_RA_RA_USEN&htmlfid=RAW14207USEN&attachment=RAW14207USEN.PDF

IBM Corporation Software Group
Route 100
Somers, NY 10589 USA

Produced in the United States of America
June 2011

RAW14268-USEN-00