sagentia

Medical devices: the shift from embedded to connected

Five principles to help software architects walk the line between opportunity and threat

By Paulo Pinheiro and Anthony Hayward, from the Electronics, Software & Systems group at Sagentia

Software architects in the medical device industry have never had it easy. Delivering innovation within a rigorous regulatory environment is no mean feat. This is compounded in the digital age: connected devices offer much potential, but they cannot be fully isolated from external risks. In this paper, Paulo Pinheiro and Anthony Hayward share five fundamental principles to mitigate risk and maximise opportunity in a connected world.

Driving advancements in medical devices for the benefit of patients can be inherently problematic. On the one hand, manufacturers are striving for seamless, frequent throughput of new products and product updates. On the other, their work must gain regulatory approval. In the digital age, the latter is increasingly complex, encompassing cybersecurity as well as user and patient safety. And this raises new challenges for medical device software architects.

Unfortunately, the software architecture that gets a product to market in the shortest timeframe isn't necessarily one that regulators will accept. It takes experience, detailed market & regulatory understanding and skill to develop a solution that marries regulatory approval with speed of delivery.

The rise of connectivity and commodity software presents a wealth of opportunity, but it's a double-edged sword. With discrete, unconnected devices, software code is embedded and contained within a single functional unit. It's possible to own end-to-end code development and control risk with a top-down approach. But with connected devices, graphical user interfaces (GUIs), web apps and mobile apps, there are multiple owners with distinct business, operational and regulatory concerns.

You can't have complete control in this environment. So, you have to weigh up the options. There is a huge range of off-the-shelf software components available that could accelerate development of a medical device, provided by major vendors like Microsoft and Oracle,

one-developer open source projects and every kind of provider in-between. These components are SOUP (software of unknown provenance) as they have not been developed to medical standards. Can you safely use them in your device? Can you afford not to use them, and incur the cost and risk of building everything from scratch?

Most software architects need to use all three options to some extent. The secret is to understand when it is appropriate and beneficial to leverage third party software, while controlling the associated risks.

So, what does it take to achieve this?

1. Upfront decisions are critical: be informed and prudent

Front end software architecture choices are always important, but especially so in the medical device sector. Decisions don't just impact market success. Selecting the wrong operating system or programming language could greatly reduce the chances of regulatory approval and therefore market entry.

Consider the potential risks attached to a surgical tool control system. If architected in a non-real-time operating system, it is unlikely to be approved. Should it freeze, it could have consequences impacting life.

This is an extreme example to underline an important and complex point. The software architect has to choose various technologies, from programming languages and operating systems to databases, data/information exchange protocols and potentially cloud/hosting options. They must make these selections knowing there is no perfect choice, only a defensible choice. Regulation, software longevity, safety, cybersecurity and usability all need to be considered. And the final decision has to balance the benefit of harnessing available, ubiquitous technology with the safety/control of bespoke, tightly guarded code.

2. Safety is king: embrace risk-based decisions

Reconciling the apparently opposing forces of innovation and safety demands a risk-based approach to software architecture. Medical standard IEC 62304 has an A / B / C classification system for medical device software, based on potential hazard(s) that could cause injury to the user or patient. The architect needs to deconstruct system architecture and segregate it so that each software item/unit can be classified appropriately. This enables architects to take advantage of well-tested, ubiquitous software for some parts of the system while deploying higher levels of safety and control in others, such as 'class C' elements where death or serious injury is possible.

Documentation is an important aspect of risk-management. For us as a consulting organisation this is critical, because clients can't properly review an architecture unless we document it effectively. Traceability for segmentation based on the A / B / C software classification, and a clear outline of which part of the system performs which function, is essential. The hierarchy and risk case for each technology selection also needs to be covered, with a robust rationale explaining how risk is managed. For example, certain functions, such as device configuration, upgrade or calibration, might only be used in the absence of a patient. By design, these cannot be inadvertently activated during treatment and therefore a different risk profile would apply.

IEC 62304

The international standard IEC 62304 applies to the development and maintenance of medical device software when software is a medical device in itself, used as a component or accessory of a medical device or used in the production of a medical device.

It requires manufacturers to classify software based on its potential to create a hazard that could result in injury:

- Class A: No injury or damage to health is possible
- Class B: Nonserious injury is possible
- Class C: Death or serious injury is possible

3. Future proofing: consider how your decisions will stand the test of time

In many ways future proofing is oxymoronic. You can never be sure that choices made today will seem ideal a decade from now. This is another area where medical device software differs from standard software; the time horizons are much longer. Nobody expects individual consumer devices – phones, wearable tech, apps – to be in use ten years hence. These products generally follow a three-to-five-year obsolescence cycle at most. Yet time horizons for medical products can extend to ten or even 20 years. In this environment, the software architect needs to distinguish between fads and more significant trends, while allowing scope for future development.

There is no perfect way of doing this but there are some key principles:

- **Beware lock-in:** proprietary software (generally delivered by small, specialist or niche vendors) can be extremely limiting, even if it claims to solve 90% of current problems. Some vendors and consultancies tie clients into IP-encumbered technology stacks. If you follow this path, future development of your own product may be curtailed, as you will be dependent on the third-party upgrade path and product investment.
- **Look for standards-based and open source:** when you've decided which parts of your architecture can make use of commodity software, make choices that are as open as possible. Anything standards-based, in terms of its technology stack and the data standards it can read/write, should give you flexibility and support future development. Try to avoid any unnecessary bias and be as agnostic as possible in your technology choices. There may be good reasons to prioritise some third-party providers (you already have skills and resources inhouse that support a certain technology stack, for example) but don't get funnelled down a particular route without considering potential consequences.
- **Assess ubiquity and belief in the software:** try to assess breadth of use, popularity, length of time in the market and commitment of key industry players to the technology. This isn't easy, but you may be able to find anecdotal evidence through desk research, networking at industry events or by talking to colleagues who have worked for other organisations.

In many ways future proofing is oxymoronic. You can never be sure that choices made today will seem ideal a decade from now.

4. Usability: think about the impact of architectural decisions

It's not just human factors specialists that have to consider the use or potential misuse of medical devices. Software architects' decisions also impact usability, and can't be undone down the line, no matter how talented the UX designer. System performance is largely dictated by architecture, and this impacts responsiveness which is critical to an optimal user experience. Similarly, architectural decisions influence robustness, which has ramifications for both ease of use and safety.

A system's GUI also has to be assessed for usability and compatibility with the wider architecture. The needs of various user groups, and the specific interfaces they will be exposed to, should be carefully considered. A clinician's requirements will be quite different to those of maintenance staff and manufacturers. Even software developers and testers expect a good user experience.

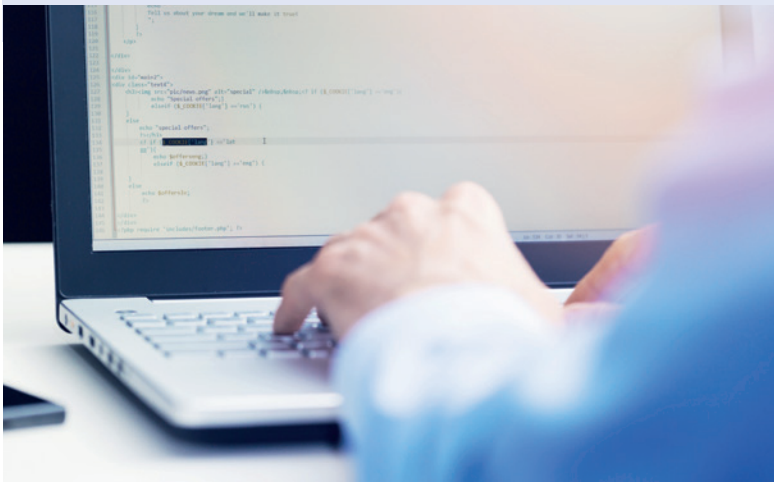
Another important consideration is the platform on which these end users might access the system. Sometimes our clients forget that when we verify the software, we have to do it on a given platform. But even a straightforward phone application has to account for device fragmentation such as different screen sizes and resolutions, different hardware platforms and different versions of the operating system.

When functional features can be impacted by the platform into which the software is deployed, you have to think very carefully about usability. Take a system which displays information to users in real-time. What happens if it's deployed on a hospital PC which doesn't have a real-time operating system? How do you avoid users assuming they are working with the latest data when in fact the machine froze some time earlier? This was in fact a real scenario encountered by one of our clients. We solved it by placing a clock on the screen at all times, so users could immediately detect when the system had frozen, as well as receiving an alert when it unfroze. Sometimes an architectural consideration (need for a real-time operating system) is impacted by an environmental reality (you can't mandate the machine), and therefore the GUI design has to compensate.

5. Software security: build it from ground up

Connected medical devices are inevitably exposed to cyberthreats. In this complex environment, people, devices, software and services interact with the support of globally distributed physical information and communication technology.

> Cybersecurity is a major concern. It needs to be front of mind at the outset when an architect is selecting third party software, and throughout the development cycle.



Since the FDA first introduced guidance in 2014, effective cybersecurity has been a necessary requirement for any medical device using wireless, internet and network connections to exchange health information. This guidance was updated in 2018 to include recommendations for device design, labelling and documentation for premarket submissions of medical devices with cybersecurity risk.

There are well-established risk management frameworks (such as NIST 800-30) for conventional IT

systems. Yet it's widely recognised that little guidance is available for managing cybersecurity risks of medical devices. In 2016, the Association for the Advancement of Medical Instrumentation (AAMI) took a welcome step, moving medical device manufacturers towards a coherent security risk management framework. The Association's report *TIR57:2016 Principles for medical device security – risk management* provides guidance on managing the risks associated with security threats and the impact of these risks on data, confidentiality, integrity and device availability.

The recommendation is that manufacturers establish a companion security risk management process alongside existing ANSI/AAMI/ISO 14971-based safety risk management processes. Safety risk involves evaluating the probability and severity of a hazard leading to harm. Security risk, however, assesses the likelihood that a threat will successfully exploit a device vulnerability. An event of this nature could compromise system confidentiality, integrity, and/or availability.

Finally, device submissions to the FDA may now leverage testing and declarations of conformity to UL 2900-2-1 to streamline product review. The FDA's 2017 recognition of UL 2900-2-1 provides manufacturers and developers with tools to meet its evolving expectations for medical device cybersecurity risk mitigation.

Manufacturers are advised to:

- employ a risk-based approach to the design and development of medical devices with appropriate cybersecurity protections based on ISO 14971
- take a holistic approach to device cybersecurity by assessing risks and mitigations throughout the product's lifecycle
- create an architecture capable of addressing the cybersecurity design recommendations in the FDA guidance: Identify and Protect, Detect, Respond, Recover.

In the face of rapidly evolving cybercrime, regular security upgrades will be required to mitigate emergent threats. Given the expected lifespan of many medical devices, it is vital that they are architected with this as a primary capability.

¬ Conclusion: striking the right balance

Medical device software architects make the early, big decisions: from defining the technological framework and identifying selections that need to be made, to choosing the technology set. Upfront architectural choices impact all subsequent work, and have a significant bearing on whether a project completes on-time and on-budget, or overruns incurring major costs and inconvenience.

In the digital age, medical devices need to make the most of connectivity, while mitigating associated risks. Delivering better outcomes without losing sight of safety and security requires a high level of pragmatism and awareness. The five principles outlined here represent a risk-compass, helping software architects make defensible decisions and confidently walk the line between threat and opportunity. Ultimately, they underpin an efficient, progressive and responsible approach to medical device development.

sagentia

Biographies

Dr Paulo Pinheiro is Head of Electronics, Software and Systems at Sagentia.

Paulo has extensive experience in delivering projects ranging from automated machinery to embedded products in a wide range of regulated industries. He is both an experienced project manager and a software team leader. Paulo brings an extra dimension to his projects through his extensive hands on development experience which leads to successful project delivery.

Dr. Anthony Hayward is a Senior Consultant in the Electronics, Software and Systems group at Sagentia.

Ant has over 10 years' experience in enterprise software architecture, development and implementation. He is a lead software architect and engineer and has worked on a number of major medical system developments including for high-performance genomics software. Prior experience included architecting an integrated e-commerce system for major UK retailers with web, mobile and batch processing aspects. He also worked across a number of other industries including public transportation and government systems.

sagentia

a **science group** company

Sagentia Ltd ¬
T. +44 1223 875200

Sagentia Inc ¬
T. +1 650 931 2585

info@sagentia.com
www.sagentia.com