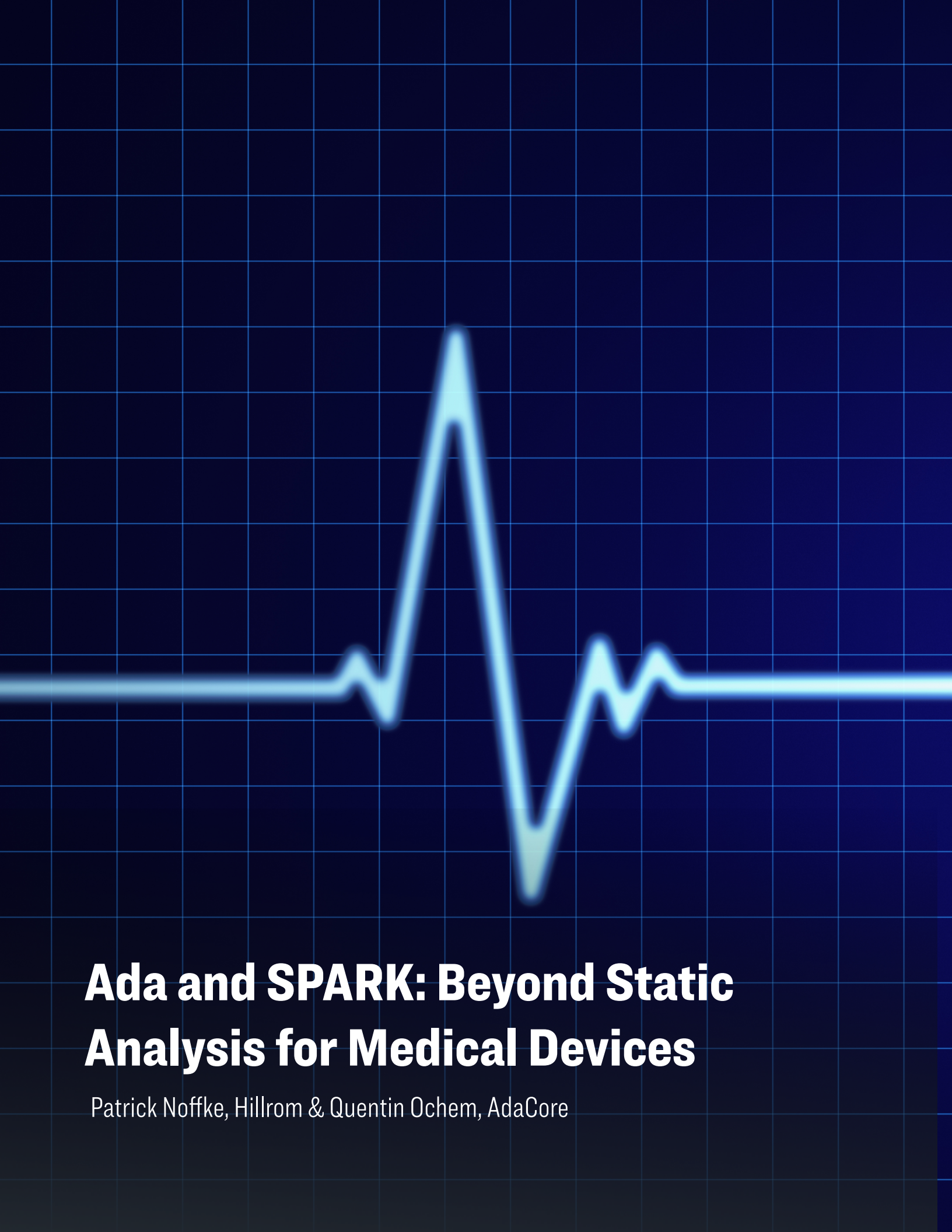

Ada and SPARK: Beyond Static Analysis for Medical Devices

Patrick Noffke, Hillrom & Quentin Ochem, AdaCore

FDA and Medical Devices

The need for safe and secure software — including Medical Device software — is becoming more important as cybersecurity threats in the healthcare sector become more frequent and severe. Constant innovation drives more and more complex life-supporting devices, which can malfunction either from design defects or malicious attacks. In order to mitigate these risks, the Food and Drug Administration (FDA) in the US has published guidance[1] that recommends, in particular, the use of static analysis technologies as part of the software verification process. Similar standards, such as IEC 62304, also promote the use of such techniques to improve overall software quality.

Hillrom Use Case

Welch Allyn® devices — a Hillrom product family — provide numerous solutions to the market that all need to be proven with objective evidence to be safe and perform as expected. One specific example is the electrocardiogram (ECG) software. These algorithms are used in a number of devices such as electrocardiographs, Holter analysis software, stress test systems, and continuous patient monitors. Patient monitors are designed to produce an alarm when lethal arrhythmias are detected. Failure to detect these arrhythmias can lead to patient harm, and thus the software is classified at the highest level of safety, safety class C (per IEC 62304).

In theory, one way to ensure safety is to exhaustively test software. However, this is almost always impractical. The ECG algorithms are exposed to a wide range of input signal morphologies, and it is impossible to test each one. One strategy is to record large databases of ECG representing very diverse signals. These are, however, expensive to collect and still cannot span all possible ECG signal morphologies. Another test source comes from issues reported by customers. However, customers don't have access to the details of the code running on their device, and their reports often lack the level of detail needed to produce useful reproducers.

One way to cope with the difficulty of testing is to add self-checking code in the algorithm and to detect live issues. This is sometimes referred to as "defensive code." However, ECG algorithms are used on a large variety of devices, often with limited CPU resources. As a result, the algorithms must be as efficient as possible to fit within the constraints. The addition of defensive checking code would further constrain the size of the algorithms.

In brief, the ECG algorithm challenge is to make sure that a piece of code can be used with the utmost safety in a wide variety of contexts and a very resource-constrained environment.

Going Beyond Static Analysis with Formal Proof

A complementary technique to database testing is the use of static analysis, which provides a very significant step toward improving software security and safety. However, it's important to understand what traditional methodologies can and cannot do. Because static analysis tools are usually applied after the code has been developed, they are dealing with software that likely has not been designed with analyzability in mind and therefore pose an undecidable problem. Therefore, these tools are usually configured to be pragmatic and useful (they will quickly find some real problems but may miss others) as opposed to comprehensive (thus they will not guarantee to find all instances of the problems they are looking for).

The SPARK technology tackles the problem the other way around. It will require the developer to start from a language and programming patterns that are suitable to static analysis. This means using a programming language that allows the developer to express as much specification as possible, which is why Ada is particularly suitable to serve as a foundation, as opposed to other embedded systems languages such as C.

Using SPARK and Ada, the programmer can much more easily demonstrate complete absence of the kind of problems that other static analysis tools could only partially discover, such as buffer overflows, divisions by zero, out of range assignment, uninitialized variables, or anything that would lead to an exception being raised. But SPARK and Ada bring additional benefits. Because the code is written for static analysis, functional requirements can be expressed at the specification level in the form of boolean pre- and postconditions. The implementation can be proven to meet its specification no matter which path is taken, no matter which values are provided as input.

The demonstration itself is accomplished through a tool called GNATprove, which considers all possible values and all possible paths of execution of a given subprogram (also called function in most languages). It uses state-of-the-art formal proof technologies such as CVC4, Alt-Ergo and Z3 to prove that all stated properties are guaranteed no matter which path or which input.

GNATprove analysis is extremely scalable. While most static analysis tools would see a major performance degradation on large applications, GNATprove analyzes each subprogram separately, attempting to demonstrate the correctness of the output condition (postcondition) against the input condition (precondition), while verifying absence of run-time errors through the control flow of the subprogram. This paradigm not only results in a linear time increase as the size of the application grows, it also allows the developer to concentrate on a small

section of the code at a time, and permits a mixed approach where some parts of the program are verified through proof, and others through testing - or even written in different languages.

Adoption of SPARK for ECG Algorithms

Hillrom has decided to migrate from C++ to Ada and SPARK for ECG algorithm development. This includes both new code as well as translation of some legacy C++ applications to Ada and SPARK. The objective of this adoption is threefold: it is aimed at verifying the safety of the code by formally proving some properties, improving its efficiency by removing dynamic checks, all the while keeping development and verification costs down.

Demonstrating absence of run-time errors in particular allowed removing a number of checks in the resulting software while still guaranteeing absence of overflow. One especially useful feature is Ada's fixed-point capability. Because of resource constraints as mentioned earlier, many target processors don't offer a Floating Point Unit (FPU). Ada has native support for fixed-point arithmetic, which uses machine integers. It was then possible to use SPARK to demonstrate absence of a number of run-time errors, which allowed removal of the associated dynamic checking code. This resulted in direct CPU performance improvements.

Ada and SPARK also helped improve the design of the application. During the initial translation of the code, the team used native integer types, which resulted in many unprovable checks. The solution was to improve the specification of the types, in particular by adding ranges or creating different types for different purposes. This allowed removing further checks and made the code overall easier to read and understand.

The cost savings of the new technology shine in light of the full development cycle that includes verification and maintenance. Fewer problems at integration time and during maintenance mean reduced time-to-market and increased customer satisfaction.

SPARK and Ada here and there

Ada and SPARK are being considered or adopted in various domains, such as other Medical Devices companies[2], Automotive[3], Security[4], Silicon[5] or Space[6]. Beyond increasing the reliability, safety and security of their software, Ada and SPARK adopters are also looking at controlling and reducing costs. Studies from the 90s[7] through last year[8] have consistently demonstrated lower costs when developing software in Ada, in some situations down to almost 40% lower.

Many resources are available on-line for further information about Ada and SPARK. On-line training is provided by the AdaCore Learning site learn.adacore.com, free versions of the Ada compiler and the SPARK prover can be downloaded from www.adacore.com/community, product evaluation can be requested from www.adacore.com/products. Additional resources and use cases on Ada and SPARK can be found on the AdaCore blog at blog.adacore.com .

About Hillrom

Hillrom is a global medical technology leader whose 10,000 employees have a single purpose: enhancing outcomes for patients and their caregivers by advancing connected care. Around the world, our innovations touch over 7 million patients each day. They help enable earlier diagnosis and treatment, optimize surgical efficiency and accelerate patient recovery while simplifying clinical communication and shifting care closer to home. We make these outcomes possible through connected smart beds, patient lifts, patient assessment and monitoring technologies, caregiver collaboration tools, respiratory care devices, advanced operating room equipment and more, delivering actionable, real-time insights at the point of care. Learn more at hillrom.com.

About AdaCore

Founded in 1994, AdaCore is the leading provider of commercial software solutions for Ada, a state-of-the-art programming language designed for large, long-lived applications where safety, security, and reliability are critical. AdaCore's flagship product is the open source GNAT Pro development environment, which comes with expert on-line support and is available on more platforms than any other Ada technology. AdaCore has an extensive world-wide customer base; see www.adacore.com/customers/ for further information.

1 https://www.fda.gov/downloads/MedicalDevices/DeviceRegulationandGuidance/GuidanceDocuments/ucm089593.pdf
2 https://www.adacore.com/press/scandinavian-real-heart-selects-adacore-embedded-software-development-platform-for-revolutionary-artificial-heart
3 https://www.adacore.com/press/denso-spark-automotive-research
4 https://blog.adacore.com/security-agency-uses-spark-for-secure-usb-key
5 https://www.adacore.com/press/adacore-enhances-security-critical-firmware-with-nvidia
6 https://www.adacore.com/press/lasp-selects-gnat-pro-for-clarreo
7 http://archive.adaic.com/intro/ada-vs-c/cada_art.html
8 https://www.adacore.com/uploads/techPapers/Controlling-Costs-with-Software-Language-Choice-AdaCore-VDC-WP.PDF