Demonstration of Timeliness and Benefits of Formal Methods for High Confidence Medical Devices: An Experimental Development of Hand Pendant Software for Control of a Medical Linear Accelerator

Peter Coronado, Varian Medical Systems
John Anton, Lindsay Errington, Kestrel Institute / Kestrel Technology
Bruce Krogh, Carnegie Mellon University

Abstract - This paper recommends the Varian hand pendant as an open experimental platform (OEP) medical device for comparing the benefits of tool-based, specification-based approaches with current best practices. This application provides a realistic test bed to study the issues of gathering and capturing requirements, designing and implementing code, and verifying and certifying software from the standpoint of a device manufacturer and formal methods research organizations. The paper also discusses the feasibility and challenges of using formal methods for medical devices with human safety implications, our experience in attempting to apply formal methods, our thoughts about present challenges in the technology, and our hope for improved safety and business outcomes.

HCMDSS Topics -

a) High-Confidence Medical Device Software Development & Assurance – Caregiver requirements solicitation and capture, design and implementation, verification and validation

b) Certification of MDSS - Quantifiable incremental certification of MDSS, role of design tools

1. Introduction

The state of the art for most medical device companies is to gather requirements in either natural language documents or diagrammatic modeling languages such as UML. Several case studies have shown that formal methods offer an alternative approach and the prospect of delivering code that is higher assurance.[1] Lockheed Martin applied formal methods to the development of code for the C130J and showed that the approach yielded code that was both higher in quality and less costly than standard methods [1]. A head-to-head competition between formal methods and best practices yielded similar results. Two teams were asked to design and code a card-entry security system. Both teams were given the same requirements document and were allowed the same funding and time. One team used Kestrel's Specware™ tool to formally specify and refine the system; the other team used UML and an engineering process certified to CMM Level 4 (one of the top grades in the Software Engineering Institute's classification scheme). The point of the study was a comparison of application reliability achieved with formal methods vs. best practices. The Specware system was significantly more reliable than that of best practices use alone [2]. The results would be even more compelling had the experiment included a maintenance activity, instead of just development. Another study, an elevator scheduling system developed by two groups of undergraduate students, showed that formal methods produce better quality solutions [3].

In spite of such reported successes, adoption of formal methods continues to be slow and controversial. Continuing the work of Hall [4], Bowen and Hinchey [5] attempt to enlighten the community regarding myths held about formal development of software applications. We intend to reinforce their points. In particular, in the context of a real-world medical device application we propose a study of the use of formal methods in the development of hand pendant software for a Medical Linear Accelerator. We hypothesize that this approach will demonstrate practical benefits offered by formal methods, and reinforce Bowen and Hinchey's refutation of claims that *Formal Methods are not supported by tools (Myth 2)*, and *Formal Methods are not supported (Myth 6).*

2. Use of the Hand Pendant in a Medical Linear Accelerator

Varian Medical Systems designs and manufactures medical linear accelerators (*linacs*) used in treating almost every type of cancer in hundreds of clinics throughout the world. Everyday, 100,000 patients are treated with these machines, so safe and reliable operation is paramount.

The linac is a very large machine, has many moving parts, and delivers megavoltage radiation in a radiation-shielded room. The patient is placed on a table and the radiation therapist (RT) positions the machine so that the radiation is directed at the targeted disease site and not at the surrounding healthy tissue. Treated fields range from a few millimeters up to 40cm x 40cm.

The RT utilizes one of two hand pendants to move the parts of the machine to the required positions for the treatment. The hand pendants control the motion of the collimators, gantry, and couch. In addition, they control room lighting and lasers used to position the patient.

The hand pendant consists of a paddle and a handle (see photograph). The handle contains thumbwheel controls for manual variable speed control of the selected axes and motion enable bars that must be

[1] It is possible to integrate formal methods approaches with today's requirements modeling and capture tools, like those based on UML. Since, the expressivity of those tools is not adequate to capture fully what is needed for high assurance devices, additional formal methods work is needed to provide the complement.

depressed for motion to occur. The handle has a keypad that contains data entry, menu selection and special function keys and an LED display.

3. History and Status of the Hand Pendant Software

The hand pendant code consists of approximately 2.5 KLOC of Motorola 68HC805 assembly language. In addition, several natural language artifacts exist including:

- Instructions for use of the pendant
- Pendant software unit test procedure
- Pendant Manufacturing Instructions
- Drawing package of pendant electronics
- Pendant Function & Interface Specification
- Pendant Software Requirements Spec

The earliest version of the pendant software was deployed in 1989 and the latest revision was made in 2001. This code is an example of the challenges described below for *rigid standalone artifacts*. It has been modified by only four programmers in its 16-year history. A single programmer maintained it from 1997-2001 who has since retired. Though the software has been very reliable in field use, the barriers to change it are such that only crisis situations would lead to a decision to do so.

Due to the large number of unstructured and difficult to understand interdependencies in this code, changes in requirements are not feasible. The cost of doing even minor changes is too high.

4. Specware™: an important formal methods tool

Several research groups have been developing tools based on formal methods to support the development and implementation of embedded code. The application of these tools to real product development, particularly medical devices, has been limited. An attractive feature of the hand pendant as an *open experimental platform* is that it is an existing product, meaning that lessons learned through its development can be used to benchmark the application of these tools. It also provides an opportunity to evaluate tools that support the entire development, implementation, and maintenance cycle.

Kestrel Institute and Kestrel Technology are developing the theory, technology and practice of formal software development. Specware is a software development system that supports the mathematically rigorous construction of executable code from abstract specifications. Applications developed with Specware have demonstrated meaningful benefits for three problems that plague industry:

Assurance that code meets specification

Traditional software development systems have at best a rather loose coupling between specifications and code, and tremendous amounts of effort and ingenuity have been invested to address this fundamental shortcoming. Software analysis standards differ, but it is safe to say that most of the effort invested in programming, documentation, code reviews, test suites, monitoring, and customer support is concerned solely with assuring a correspondence between the code and its specification.

Flexibility and Productivity

A second problem with traditional approaches to complex software is that each product tends to be painfully composed from the efforts of diverse developers, and then lives on as a *rigid standalone artifact*. It is hard to adapt components to work together correctly, it is hard to maintain such a system in light of changing specifications and environments, and it is hard to migrate all the specifications, documentation, code, test suites, etc., from one instance of a software family to the next.

Many of these problems are due to the large number of unstructured interdependencies that arise in code - a localized change in the requirements burgeons into an entangled nest of changes scattered throughout the code. Finding and selecting the changes requires consideration of the system as a whole, so the cost of a change grows significantly with the size of the entire system. Especially for medical devices with human safety issues, the cost of verification needed for market clearance can be a superlinear function of the size of the whole system, a fact not lost on manufacturers considering upgrades.

Performance, design changes, and product cycles

Significant performance issues with an existing product sometimes require radical design changes,

but the turn-around cycle from such design changes to re-implementation is typically on the order of months or perhaps years. Traditional software development environments all but preclude systematic exploration of alternative design strategies tightly coupled to hard performance data.

Design and implementation decisions tend to be quite front-loaded and rigid. If end-product testing reveals unexpected performance issues, it can be difficult, approaching impossible, to revise an entire system to accommodate such belatedly discovered information, so everyone lives with the results until the next product cycle a year or more later [6].

5. Recommendations

To address the objectives of the HCMDSS panel, we propose the following actions:

1. Establish an open experimental platform (OEP) for comparing the benefits of tool-based, specification-based approaches with current best practices.
2. Establish an independent organization (NIST would be a natural choice) to specify metrics, to define and monitor experiments, and to report findings to the community. Metrics could include measures of specification correctness and code reliability, performance under constrained resources, time and costs for engineering change orders to be implemented, projected time and cost of market clearance, etc.
3. Create a set of initial experiments for industry and academia:
   - to assess the limitations and effectiveness of the various approaches,
   - to identify challenges for formal methods application,
   - to point out infrastructure (e.g., standard models) and research achievements whose availability would support wider use of formal approaches,
   - to establish a path for introduction of formal methods in medical device software development

The Varian Pendant is a promising candidate for the OEP. Kestrel and Varian Medical Systems are in collaboration now on an effort that could be used as a precursor to an OEP experiment. Varian can provide historical information about the pendant to enable research on newer approaches.

REFERENCES

[1] Peter Amey, *Correctness By Construction: Better Can Also Be Cheaper,* CrossTalk Magazine, The Journal of Defense Software Engineering, March 2002.

[2] Carol Smidts, Xin Huang, James C. Widmaier, *Producing reliable software: an experiment,* Journal of Systems and Software, Vol 6, no 3, April 2002.

[3] Ann E. Kelly Sobel, Michael R. Clarkson, "*Formal Methods Application: An Empirical Tale of Software Development,*" IEEE Transactions of Software Engineering, Vol. 28, no. 3, March 2002.

[4] J.A. Hall, "*Seven Myths of Formal Methods*", IEEE Software, Vol. 7, no. 5, September 1990.

[5] J. Bowen, M. Hinchey, "*Seven More Myths of Formal Methods*", IEEE Software, Vol. 12, no. 4, July 1995

[6] J. McDonald, J. Anton, "*Specware: Producing Software Correct by Construction*", Kestrel Institute Publication, March 2001; also, for more recent documentation, see www.specware.org.