Using Model-Based Design in an IEC 62304-Compliant Software Development Process

David Hoadley, Ph.D.[1]

[1] The MathWorks, Inc.,
Novi, MI, USA
dhoadley@mathworks.com

1 Introduction

IEC 62304 [1] is an international standard (hereafter referred to as the Standard) that specifies software development life cycle processes to improve the safety of medical devices. It defines a series of activities and tasks that are required for medical device software engineering. We will discuss how MathWorks tools for Model-Based Design, which have been used to create high-integrity software [2-4], can be used to create control system and signal processing software in a manner compliant with the Standard.

2 Software development processes

2.1 Software design and implementation

In addition to representing functional behavior, Simulink® can be used to segregate a software design into major structural components, indicate their external properties, and demonstrate the relationship among the components. One application would be to separate risk control measures into a Simulink subsystem or referenced model to more intuitively track them and potentially lower the software safety class for other parts of the software architecture. Simulink Verification and Validation™ provides a facility called the Requirements Management Interface that establishes navigable links between textual requirements and components of models.

Software in safety class C (the greatest potential for harm) of the Standard requires a documented, verified detailed design for each software unit. The Simulink model or subsystem that represents each unit can be an important part of this detailed design.

Additional software standards such as MAAB [5], DO-178B [6], IEC 61508 [7] and MISRA-C® [8], can facilitate validation of the unit design. Simulink Verification and Validation features automated model checks for these standards. Design validation can also include simulations of functional, requirements-derived test cases. Model coverage can be collected during this process to test the completeness of the functional test cases.

Real-Time Workshop® Embedded Coder™ can transform a Simulink model into a well structured, optimized software implementation in the C or C++ languages [9].

2.2 Software unit verification

Test scenarios created to perform functional detailed design verification on the software unit's Simulink model can be used and augmented to perform implementation functional unit verification. Generated code can be compiled on the host, or the computer that the developer is using to design and create software, and invoked as an S-Function in a test harness model for direct model to code comparison (so called software-in-the-loop or SIL testing). In addition, it is possible to perform such tests on target-compiled code via the Embedded IDE Link™ product. This processor-in-the-loop (PIL) technique can be used to help verify that the executable software has the same behavior as the Simulink model when being executed as if part of the final device. Code coverage can be captured during testing to demonstrate that no unintended functionality was introduced.

3 Software risk management

Model-Based Design supports software risk management primarily by the aforementioned capability to link requirements to model elements. This link and Simulink Verification and Validation's Requirements Report allows one to trace from a documented hazard to control measures in the model (design) to the implementation of these measures in a generated software item. The traceability extends to the particular lines of code via comments, and is navigable via the code generation HTML Report and Simulink/Real-Time Workshop's Navigate to Code feature.

4 Software configuration management

The models and data created as part of the software engineering process must be identified and managed along with the resulting software items. Some typical artifacts that may be created during development activities include Simulink models, MATLAB scripts and functions, data dictionaries, generated production code, S-Functions and other user block libraries, simulation input data (test vectors) and results, and generated documentation such as design documents and test results [10].

5 Summary

Model-Based Design can be effective in an IEC 62304-compliant software development process. Links are supported between process artifacts (such as requirements documents) and models, allowing the relationship from requirement to design to implementation to remain strong. See Figure 1 for some common usage scenarios of the tools and the documentation artifacts that they can produce.

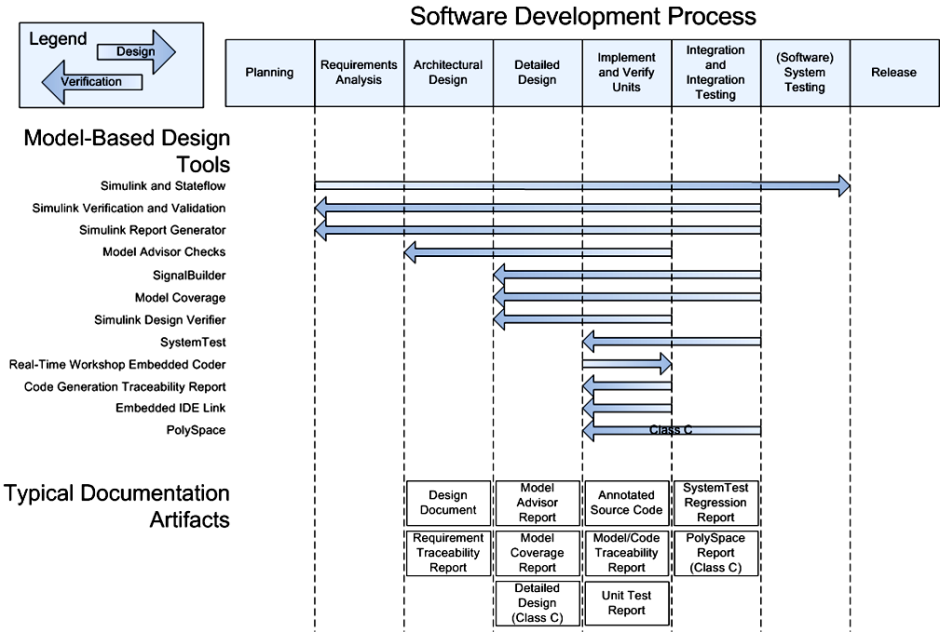

Figure 1 - Applicability of Model-Based Design Tools for IEC 62304-Compliant Software Development

References

1. IEC 62304, "Medical Device Software – Software Life Cycle Processes", International Electrotechnical Commission, Edition 1.0b, 2006
2. "Medrad Ensures Safety of MRI Vascular Injection Pump Using MathWorks Tools", 2004, http://www.mathworks.com/company/user_stories/userstory6313.html
3. "Alstom Generates Production Code for Safety-Critical Power Converter Control Systems", 2005, http://www.mathworks.com/company/user_stories/userstory10591.html
4. "Achieving Six Sigma Software Quality Through the Use of Automatic Code Generation", Bill Potter (Honeywell International), 2005, http://www.mathworks.com/programs/techkits/pcg_tech_kits.html
5. "Control Algorithm Modeling Guidelines Using MATLAB, Simulink, and Stateflow", MathWorks Automotive Advisory Board - Version 2.1, 2007
6. "Software Considerations in Airborne Systems and Equipment Certification", RTCA/DO-178B, RTCA Inc., 1992
7. IEC 61508-3:1998. Int. Standard Functional safety of electrical/ electronic/ programmable electronic safety-related systems - Part 3: Software requirements. 1998
8. MISRA-C:2004. The Motor Industry Software Reliability Association: Guidelines for the use of the C language in critical systems, 2004
9. "Certify embedded systems developed using Simulink and PolySpace products to IEC 61508 and ISO 26262", http://www.mathworks.com/products/iec-61508/
10. "Configuration Management of the Model-Based Design Process", Gavin Walker (MathWorks), Jonathan Friedman (MathWorks), and Rob Aberg (MathWorks), Proceedings of the Society of Automotive Engineers World Congress 2007 (2007-01-1775)