# Testing Medical Software to meet IEC 62304 & FDA Requirements

*by*
*Eur Ing Chris Hills BSc (Hons),*
*C. Eng., MIET, MBCS, FRGS, FRSA*

*As Published in*
*Electronic Specifier*
*January 2015*

PhaedruS SystemS

*The Art in Embedded Systems comes through Engineering discipline.*

## Contents







# Testing Medical Software to meet IEC 62304 and FDA Requirements

Before you can begin to test any software or indeed any system, medical or otherwise, you need to know is what you are testing - you need both a full product specification *and* a full test specification. Ideally the test specifications will have been generated at each level of the requirements and design phases before any code is written. All this requires that you have in place a defined development process. If you are developing medical products, IEC 62304 mandates that you have a process, which it calls a software development plan.

Within this process you will need tools, not just to assist in testing but for the entire development process starting from requirements specification and management. In choosing these tools look for those with a provenance in the medical area, i.e. meeting IEC 62304 and/or FDA documents. An alternative is tools that are aimed at the generic IEC 61508 (Functional Safety) from which descend the standards for railways, automotive, process, machinery, elevators etc. (Figure 1) As these standards have a huge overlap a tool-maker will usually validate against IEC 61508, as the "top level" standard, unless they are aiming only at a specific market. The validation will be from a third party, often TÜV and you should look for a Tool Certification Document/Kit.

Even standards not directly descended from IEC 61508 will reference it: in fact IEC 62304, Annex C 7 says: "use 61508 as a source for good software methods, techniques and tools".

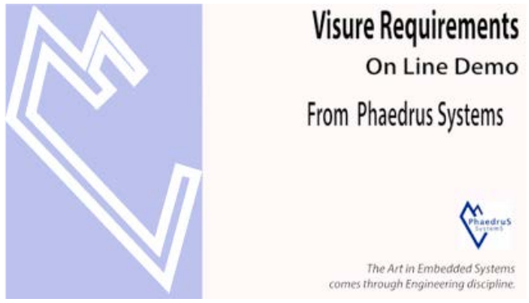
## Managing Requirements

The process starts with establishing your requirements (knowing what it is you are building). See our paper *Requirements are required* http://www.safetycritical.info/library/presentations/Requirements-001.pdf


There are various tools to assist with requirements management which use semantic analysis to automatically flag low-quality requirements containing ambiguity or inconsistencies.

One such is Visure, which meets IEC 62304 and FDA

requirements and has a strong track record in medical projects (as well as in other safety-critical fields). See Video Overview http://youtu.be/gShY3P81cBw




Rigorously defining the specifications and design gives you a solid base to define the test specifications. Your Requirements Management tool should accept the IEC 62304 requirements to ensure that your product requirements satisfy them, and so save you from nasty surprises later. A good requirements management tool like Visure will also automate, reliably and repeatedly, the validation process by generating most of the documents required for validation and, in this case, meeting FDA 21 CDR Part 11.

Assuming you have a good requirements management and tracking system ***and*** that you have actually spent time inputting and refining the requirements ***and*** validated your design before generating the test specifications you can proceed to coding and testing.

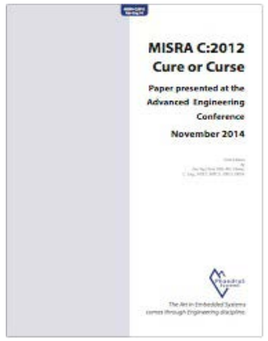
## Static code analysis

There is a testing step prior to compilation! In IEC 61508 Part 7 Table C1, the use of C and C++ is recommended only when used with a language subset, a coding standard and static analysis. (For the difference between a subset, coding standard and style guide see Hills, *Coding Standards: Do We Need Them?* Email *chris@phaedsys.com for a copy*)

MISRA-C, MISRA-C++ (or HICPP from PRQA) are subsets to tame the C and C++ languages. Normally the language subset will be rolled into a single document with the coding standard. We have some papers on MISRA-C

***MISRA-C Curse or Cure*** sets the scene for the problems that caused MISRA-C to be born and the problems caused by MISRA-C

http://www.safetycritical.info/library/presentations/MISRA-C-curse-or-cure.pdf

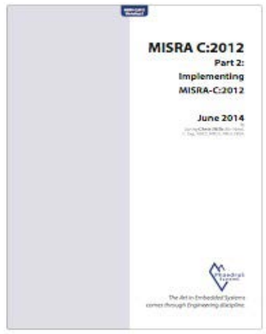



***MISRA-C Why it won't save your project*** looks at some of the reasons why incorreclty using MISRA-C can do more harm than good.

http://www.safetycritical.info/library/presentations/DD-MISRA-C3.pdf

The third paper is Implementing MISRA-C

http://www.safetycritical.info/library/presentations/MISRA-C3-0002.pdf


PRQA has a an alternative o MISRA-c with High ntegrity C++. http://www.programmingresearch.com/resources/white-papers/

A strong static code analyser (SCA) will ensure that the code is semantically and syntactically correct. Where a compiler is *required* to compile all syntactically legal code no matter how pointless or dangerous, an SCA goes much further. When combined with the MISRA-C/C++ rules the SCA will not only enforce the subset but will find semantic problems and theoretically legal but unsafe (and usually unintended) uses of C/C++. Use of a good SCA can cut 30% from a project time and save vast amounts of rework and re-testing.

One of the best SCAs on the market is from Programming Research, the company originally behind MISRA-C. QA·C/QA·C++ will do deep static analysis and enforce both MISRA-C/C++ and your own company coding-standard. The tool will also generate a wide variety of metrics. QA·C has been validated against IEC 62304 and other standards and is widely used in medical developments.





## Dynamic testing

Having ensured the code is statically correct the next thing is to ensure it is functionally correct against the design requirements (does it do what you want it to do?). We now have a myriad of terms: white-box, black-box, unit, dynamic, integration, system and functional testing and a confusing array of coverage methods. Broadly dynamic or functional testing indicates the code is actually compiled and run. Unit, integration and system test cover increasing scope starting from a single function/file moving up the entire system but the boundaries between them are somewhat fuzzy.

White and black box are clearer. Black-box tests only the interfaces to the unit under test without any visibility of the internal code, while with white-box tests you can manipulate things inside the unit under test. White box testing is required for Modified Condition Decision Coverage (MCDC) testing for critical systems. As with SCA the best way of doing this is to automate it with a suitably validated tool.

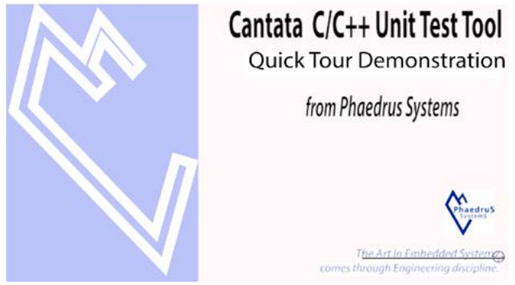
## Validated testing tool

Cantata from QA-Systems is such a validated tool. It is worth reading the Cantata Standard Briefing for IEC 62304:2006 to see where and how the requirements of the standard impinge on dynamic testing. (Available from chris@phaedsys.com )

You have the option to test on the desktop or live on the target. Testing on the desktop is very fast and requires no target hardware. For some tests where there is IO or interaction with hardware live testing on the target is essential. (In some cases a simulator can be used in place of physical hardware.) Do make sure your unit test system can work with your specific target. For example Hamilton Medical had built a platform specifically for their applications.

> "We were unsure as to whether we could get tool support for this platform as it is a unique environment," stated Christian Frehner, the Lead Software Developer of the project. "An out-of-the-box solution would not be possible. We were therefore very impressed when we were informed that a custom part of Cantata could be provided, specifically for this environment. There existed a very professional approach to meeting our particular needs."

To be sure that communications, readings and timing are accurate on the real hardware it is essential that you test on the target hardware. The question is what and how to test? (See Overview Video http://youtube/Exwwm1LYBC0 )




Cantata has several built in profiles for testing against various standards at different levels. For black box testing, generating test harnesses for these standards is virtually automatic particularly on the desktop.

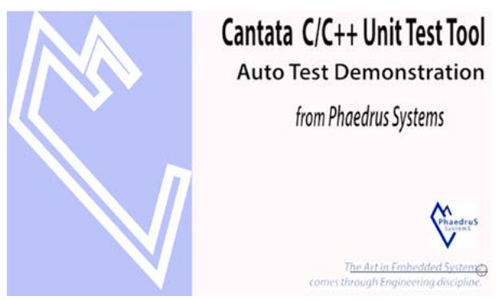
> "Cantata was very easy to get to know," Jutta Luosta, a leading test developer in GE Healthcare Helsinki, explained. "Development testing of sub-systems was really quite fast with Cantata."

## White box tests

For IEC 62304 you need to white box test the critical functions. On one GE Healthcare project, critical functions made up only about 10% of the software platform. So 90% of the dynamic testing could be automated using tests that themselves were automatically generated. Auto-generated tests give the testers a lot more time to give critical areas a higher level of focused testing.

For the white box testing of critical code the auto-generated black box test harnesses, which are generated in C or C++ as appropriate, can easily be adapted through point and click dialogues and a range of templates.

(See video on auto test case generation http://youtube/bcvIGJkLgMU )



The level of coverage can be selected for entry-point, calls, statement and MCDC. Selecting a higher level automatically includes the lower levels. The test results will highlight any decisions and statements not fully exercised, indicating the changes to tests or additional tests needed to exercise the code. These can then be auto-generated.

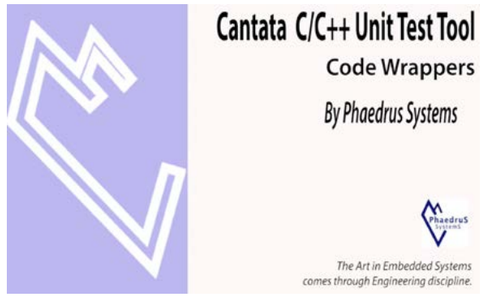
> "Once the first tests had been run, we developed a framework based around Cantata that allowed developers to use templates. Soon, all developers were testing to exactly the same standard. Maintaining each other's test scripts became much easier as a result of this standardization," said Rolf Keller, software engineer on Galileo at Hamilton Medical.

## Stubs to wrappers

In most cases units under test are stubbed. A test harness replaces all the functions called by functions in the unit under test and stubs supply pre-set data to the functions. Of course the unit under test may also contain several lined functions but only the called functions external to the file need be stubbed.

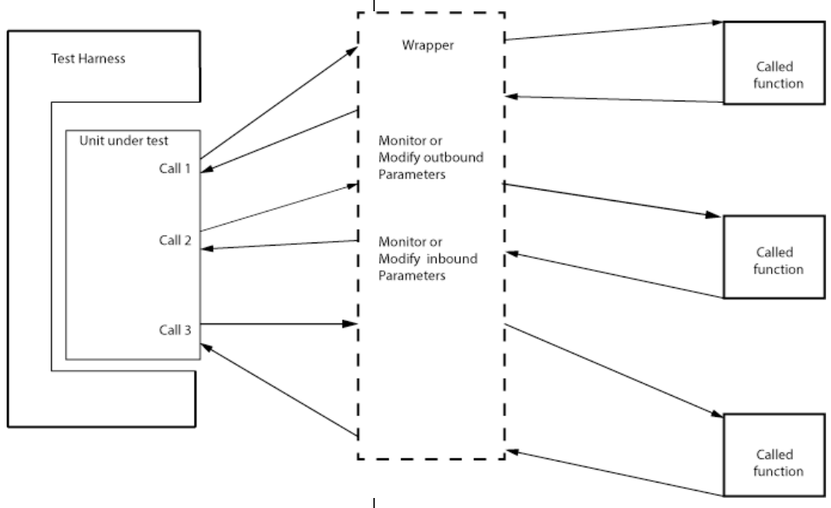
Cantata can, uniquely, use wrappers, which wrap the function under test but still call the functions that would have been called. This means the build can include all the functions and functionality. The overall data and code flow is the same as an operational system.

The wrapper can check or modify any of the inbound or outbound parameters. This lets the testers insert any data to modify the program behaviour and permit the testing of defensive programming and infeasible code paths. This is particularly useful when certain values would be difficult to obtain from actual hardware without damaging it or the system. (See video on using code wrappers http://youtu.be/dTPBIAycS_s )




As well as HTML and XML reports, there is a "Certification Ready" textual report, which greatly speeds up the production of certification artefacts. The Certification Ready report should be put into the requirements management system to close the loop.

The Cantata test and reporting system fills the requirements of IEC 62304:2006. Full details are in the Cantata Standard Briefing for IEC 62304:2006 available without NDA. This primarily it addresses Clause 5: SW development process, Clause 6: SW Maintenance process, Clause 8: SW Configuration Management Process and Clause 9: SW Problem Resolution Process.








## Legacy Code

The ability of Cantata to very quickly auto-generate and run both black and white box tests on the desk top makes the tool easy to use on legacy software. Static analysis tools will also run on legacy code. It can be very economical to use quality tools on new versions of a device. Adding MISRA-C/C++ or HICPP to legacy code can be more problematic and is best done for a module or file at a time as new work is being done.

Meeting the requirements of IEC 62304 is never going to be easy, but a carefully collected set of tools as part of well thought out development process is going to ease the pain and improve significantly the speed of development and quality of the final product. Just generating test specifications at each level of the requirements and design phases before any code is written provides considerable economies. Capers Jones in his book, ***Economics of SW Quality*** provides extensive facts and figures. Our paper ***Requirements are Required*** has a full set of links to further information on this book including a video presentation by the author

## References

http://www.phaedsys.com/library/presentations.html )

Requirements Tool: http://www.phaedsys.com/principals/visure/visure.html

Requirements paper: http://www.phaedsys.com/library/presentations.html

Static Analysis: http://www.phaedsys.com/principals/programmingresearch/index.html

Implementing MISRA-C: http://www.phaedsys.com/library/presentations.html

Dynamic test: http://www.phaedsys.com/principals/programmingresearch/index.html

# Testing Medical Software to meet IEC 62304 and FDA Requirements

## Originally published in Electronic Specifier January 2015

First edition January 2015


## Phaedrus Systems Library

The Phaedrus SystemsLibrary is a collection of useful technical documents on development. This includes project management, integrating tools like PC-lint to IDE's, the use of debuggers, coding tricks and tips. The Library also includes the QuEST series.

Copies of this paper (and subsequent versions) with the associated files, will be available with other members of the Library, at:

**http://library.phaedsys.com**

PhaedruS SystemS

*The Art in Embedded Systems comes through Engineering discipline.*