Verification of Executable Object Code from a Model (Version 1.0)

Steve Morton
FAA Software DER
LDRA Certification Services

www.ldra.com

Introduction

The introduction of RTCA/DO-331 Model based Development and Verification Supplement to DO-178C and DO-278A offers new opportunities to leverage the strengths of model based development under RTCA/DO-178C. The concept of model simulation for Executable Object Code (EOC) verification credit allows for the painstaking work of model verification to be reused to partially achieve EOC verification objectives.

This paper explores the conditions under which model verification can be used to partially satisfy EOC verification objectives and identifies areas which should be closely attended to in order to satisfy the regulatory requirements.

Model Based Development and Verification (RTCA/DO-331)

Under DO-178C and DO-331, some aspects of EOC verification can be satisfied using model simulation. Per DO-331 section MB.6.8.2:

> *Verification of the Executable Object Code is primarily performed by testing. This can be partially assisted by a combination of model simulation and specific analyses as described below. This combination can be used to partially satisfy the following software testing and coverage objectives:*
>
> 1. *Executable Object Code complies with the high-level requirements.*
> 2. *Executable Object Code is robust with the high-level requirements.*
> 3. *Test coverage of high-level requirements is achieved.*
> 4. *Test coverage of software structure to the appropriate coverage criteria is achieved.*
> 5. *Test coverage of software structure, both data coupling and control coupling is achieved.*
>
> *But specific tests should still be performed in the target computer environment, since some errors may be detectable only in this environment.*
>
> *The following software testing and coverage objectives cannot be satisfied by model simulation since simulation cases should be based on the requirements from which the design model is developed:*
>
> 6. *Executable Object Code complies with the low-level requirements.*
> 7. *Executable Object Code is robust with the low-level requirements.*
> 8. *Test coverage of low-level requirements is achieved.*

It bears repeating that the simulation cases and procedures to be used for model verification are required to be developed from the higher-level requirements from which the Design Model itself was developed. It also should be noted that simulation cases and procedures are subject to the same verification objectives as test cases and procedures used in more traditional paradigms to verify the EOC in the target environment.

Required Model Analyses

DO-331 section MB.6.8.2 continues with a list of considerations for showing that the design model is valid for use as an EOC verification tool. These aspects, and their ramifications, are discussed in detail below.

Model Relationship to the EOC Source

From MB.6.8.2:

> a. *When certification credit is sought from model simulation to partially satisfy software testing objectives and test coverage regarding high-level requirements (see DO-178C 6.4.a, 6.4.b, and 6.4.4.a), then these activities include:*
>
> 1. *Ensuring that the same Design Model is used for simulation as is used for the production of the Source Code and/or Executable Object Code.*
>
>   On the surface, this sounds as though simple configuration management practices are enough to answer this concern. However, there are nuances that pertain to showing that the model is truly representative of the EOC.

For instance, most model based developments are hybrid in nature – where the model represents the bulk of the source code, but certain segments are derived from other sources, such as hand coding. The interactions between the model and these other code segments must be accounted for in the model based verification planning. A typical example of this approach is the use of data definitions from either hand coded files or a data dictionary tool that is not tied to the model. If a value embedded in the non-model code is used as a comparison point in the model, then the model may not be wholly representative of the Source Code and resultant EOC.

Another common example of hand-generated source code being blended with model generated code is the use of hand coded assembly language subroutines in specific areas to enhance execution speed.

Specific Tests Performed in the Target Environment

MB.6.8.2.a continues with specifics of what testing of high level software requirements can be performed in a simulated system with the stipulation that any testing relegated to the target system must conform to the testing and structural coverage objectives of DO-178C:

> 2. *Determining what software testing and test coverage objectives and which specific high-level requirements are planned to be satisfied by simulation; all other software testing and test coverage objectives should be satisfied by testing as described in DO-178C section 6.4. Typical errors listed in DO-178C section 6.4.3.a cannot be detected by model simulation since they require the target computer hardware.*
>
>   *Typical errors consistent with DO-178C sections 6.4.3.b and 6.4.3.c that may be revealed by model simulation include:*

- *Inadequate end-to-end numerical resolution.*
- *Incorrect sequencing of events and operations.*
- *Failure of an algorithm to satisfy a software requirement.*
- *Incorrect loop operations.*
- *Incorrect logic decisions.*
- *Failure to process correctly legitimate combinations of input conditions.*
- *Incorrect responses to missing or corrupted input data.*
- *Incorrect computation sequence.*
- *Inadequate algorithm precision, accuracy, or performance.*
- *Incorrect state transitions.*

Model simulation cannot be used to detect errors related to the target computer hardware, for example:

- *Incorrect handling of exceptions, such as arithmetic faults or violations of array limits.*
- *Data corruption, especially global data.*
- *Timing related requirements and performance.*
- *Hardware resource related performance.*
- *Hardware monitoring requirements (for example, built-in test).*

For EOC functionality wholly represented by the model, non-hardware dependent aspects are available for verification through model simulation. Hybrid modeling environments, where the model is surrounded by non-modeled code, can complicate the ability to claim credit for verification in the modeling environment.

Model Environment to Target Environment Comparison

3. *Justifying, in detail, how the simulation activity satisfies the specific software testing and test coverage objectives identified in item 2 above. To this purpose, an analysis should provide compelling evidence that the simulation approach provides equivalent defect detection and removal as testing of the Executable Object Code in compliance with DO-178C section 6.4.3. This analysis should:*

   i. *Verify how representative the model simulator environment is compared to the target computer environment, addressing processor differences, if any. Topics such as floating point precision, integer word sizes, math libraries, set of instructions, etc. should be addressed, depending on the certification credit sought. The analysis should be supported by the execution of a representative set of verification procedures on both environments. In addition, the analysis may rely upon confidence gained from features such as target system's hardware abstraction layer and/or target computer system's operating system capabilities.*

   ii. *Address the equivalence and differences between the simulation executable object code and the target Executable Object Code. These differences can be the result of:*

- *Different compiling and linking process: The differences to be considered are typically compiler options and/or optimizations. These differences should be analyzed with respect to the software testing objectives identified in item 2 above. The analysis should be supported by the execution of a representative set of verification procedures on both environments. Considerations defined in section MB.4.4.2.a should be addressed during this analysis.*

- *Source Code differences: Though the source code used for simulation is expected to be the same Source Code used to produce the target Executable Object Code (for example, the same autocode generator is used for both), any applicable differences to improve testability (for example, source code or model element library instrumentation) should be justified regarding their effects on the produced executable object code and subsequent achievement of the software testing objectives identified in item 2 above.*

Note: The two items above are not mutually exclusive.

Because the model, in its modeling environment, is being used as a representation of the EOC, with execution (on the target hardware) as the subject of the verification, "compelling evidence" must be provided to the certification authority to justify the granting of certification credit. While the definition of "compelling evidence" is left to the governing certification authority, it stands to reason that a detailed, qualitative analysis of the two environments is necessary. The analysis must encompass the whole set of tools that lead from the model to the source code, including the automatic code generator, the compiler(s), and linker(s), and any differences in libraries for the dual platforms.

DO-248C FAQ 75 clarifies that this "compelling evidence" should not be based on statistical sampling:

> *The verification objectives described in DO-178C/DO-278A section 6 should be fully satisfied. Sampling should not be used to fulfill these objectives. This also applies to coding standards referenced in the question above (DO-178C/DO-278A section 6.3.4.d). The only exception to this is that regulatory authorities may accept less than 100% verification of conformance to the code presentation rules as defined in DO-178C/DO-278A section 11.8.b.*

It is also useful to note that the EASA exemption of general purpose microprocessors from the application of DO-254, as noted in EASA CM-SWCEH-001 Issue 01 Rev 01 Development Assurance of Airborne Electronic Hardware section 9.2, shown below, is contingent on the full exercise of the DO-178B (and therefore C) software verification objectives and activities.

> *Software and microprocessors are out of scope of this Section. The development assurance of microprocessors and of the core processing part of the microcontrollers and of highly complex COTS microcontrollers (Core Processing Unit) will be based on the application of ED-12B/DO-178B to the software they host, including testing of the software on the target microprocessor/microcontroller/highly complex COTS microcontroller.*

Structural Coverage Considerations

DO-331 section MB.6.8.2 allows for the granting of credit for the satisfaction of structural coverage objectives appropriate to the level:

> b. *When certification credit is sought from model simulation, simulation results may partially support the test coverage objectives of the software structure (see DO-178C 6.4.4.c and 6.4.4.d) only if the source code used for simulation is identical to the Source Code used to produce the target Executable Object Code.*

Note that the use of the word "identical" here is different from the strictures presented in item a, which merely require the model to have a parental relationship with the Source Code. Further, the "source code" used for the simulation is required to be identical to the "Source Code" (see DO-178C section 11.11) used to generate the EOC, although the method for showing this is not addressed.

DO-331 section MB.6.7 states that "Model coverage analysis is different than structural coverage analysis and therefore model coverage analysis does not eliminate the need to achieve the objectives of structural coverage analysis per DO-178C section 6.4.4.2." This seeming contradiction is addressed in FAQ 11 in Appendix B of DO-331, titled "**May the applicant use the model coverage analysis activity to achieve the structural coverage analysis objectives?**":

> *This FAQ assumes that model coverage analysis has been performed as defined in MB.6.7.*

As noted in MB.6.7, "Model coverage analysis is different than structural coverage analysis and therefore model coverage analysis does not eliminate the need to achieve the objectives of structural coverage analysis per DO-178C section 6.4.4.2." However, model coverage analysis may be used as a means for achieving structural code coverage analysis objectives under appropriate conditions. In that case, the applicant should include the plan for this approach in the Software Verification Plan and capture the actual demonstration/verification as part of the Software Verification Results. These conditions should be evaluated on a case-by-case basis and agreed upon by the certification authorities, in order to take into account specific aspects of the modeling notation, model coverage criteria, and code generation characteristics.
These conditions should include at least the following:

- *Model coverage analysis criteria hold the same properties as the applicable structural code coverage analysis criteria hold for the level of the software being developed, for example, MC/DC coverage for the level A.*

- *Qualification of the code generation tool chain with respect to objectives for which certification credit is sought (in particular Annex MB.A Table MB.A-7 (Annex MB.C Table MB.C-7*

- *Any libraries used by code generated from the Design Model are verified according to DO-178C/ DO-278A section 6, including structural code coverage activities in accordance with the required software level.*

FAQ 11 is supplemental information that is intended to assist the applicant comply with the verification guidance discussed previously in this paper. Specifically, according to DO-331 MB 6.8.2, this FAQ clarifies how the applicant might be able to obtain partial credit for verification of high level software requirements. The MB 6.8.2 guidance with respect to low level requirements and the necessity of verification at the EOC level is not addressed by FAQ 11.

In addition, DO-331 MB.6.8.2 specifies that simulation cases and procedures are based on the requirements from which the design model is developed, meaning that derived low-level requirements represent a special challenge for structural coverage measured against a model. Intrinsically, the existence of a requirement implies the existence of a code structure to implement those requirements. Derived requirements typically do not trace to higher level requirements, and therefore are not represented in the requirements from which the simulation cases and procedures are to be developed. Coverage of software structure from model coverage criteria when there are derived low-level requirements therefore tends to be problematic, and using a single level of requirements under DO-178C is subject to justification.

The previously mentioned issue of data defined in non-model generated Source Code is also a difficulty, especially in the instance of constants defined in the otherwise-generated Source Code. This violates the requirement specified in MB.6.8.2.b.

It is also wise to remember that structural coverage offers one final important verification: That the design, as implemented, does not violate the precepts and conditions envisioned by the designers, such that all sections of the implementation are useful and reachable. If X must be greater than 100 to satisfy a requirement, then X must be capable of exceeding 100 in order for the requirements, as written, to be valid.

Parallelism in Development and Verification

There is another consideration to account for in this discussion, as well. DO-178C, including its supplements like DO-331, envisions a parallel path of development and verification of the software. When requirements are created, the requirements are subjected to analytical verification – reviewed for completeness and correctness – and then are submitted for verification by demonstration – creation and execution of test cases. With model based development, some degree of the analytical verification is performed via simulation – using demonstration of the effect of those requirements (in the modeling environment) meet the intent of the creator of those requirements.

In non-model development, the written requirements are reviewed, and then commonly handed to two teams for further use – the development and verification teams. The verification team is supposed to work in parallel to the development team, but often is unable to match either the pace or the manpower of the developers in order to have test cases and procedures ready for use when the implementation appears on the scene.

With model-based development, where simulation cases and procedures are used in part to validate the model-based requirements, there is a golden opportunity to reuse those validation cases and procedures as verification against the EOC, if they can be directly translated into tests in a non-simulated environment and be subjected to the test coverage activities described in DO-178C, section 6.4. Modern verification tools, specifically designed to enhance reusability of simulation cases and procedures as tests, offer a chance to achieve the holy grail of reusability in the verification arena – the very battleground that sets aerospace software apart from so many other fields.

Conclusion

RTCA/DO-178C and DO-331 introduce the concept of using model verification to partially satisfy certain aspects of Executable Object Code verification objectives. Inherent in this concept is a requirement to demonstrate validity of the model, in its modeling environment, with respect to being a true representation of the aspects of the EOC being verified.

Both functional verification and structural coverage objectives may be partially satisfied through model simulation, subject to specific and detailed criteria for demonstrating model validity for the credit being sought. Any use of model simulation to partially satisfy EOC verification objectives is required to be identified in the software plans, and is subject to certification authority agreement. The applicant is required to supply “compelling evidence” to the certification authority to demonstrate validity of the modeling environment for the credit sought.

Using a hybrid model/non-model development environment is a complicating factor in establishing the requisite validity, as is variance introduced by using different compilers, linkers, or libraries in the modeling and target environments. Derived low-level requirements may be especially problematic for the satisfaction of software structural coverage objectives.

The DO-178C guidance (as opposed to supplemental information) regarding on-target testing of low-level requirements and structural coverage analysis does not substantially differ from DO-178B. Model verification tests, by definition, are created using the requirements from which the model was developed. The traditional model of testing the executable object code on target hardware or otherwise acceptable test environment remains the long-accepted and preferred method, with credit for model-simulation coverage potentially available only if the stringent rules illustrated by DO-178C 6.8.2.a.3.i for correlating the model environment to the target environment are satisfied.

Finally, using state of the art independent verification tools and techniques, the effective reuse of the model simulation cases can be achieved. These simulation cases can be exported to native target test harnesses that re-execute test scenarios with complete fidelity. Moreover, the structural coverage analysis process can be performed as a by-product of the automated test harness operations, aiding the applicant in showing that the DO-178C verification process has been completed to the satisfaction of all applicable objectives.

About Steve Morton

Steve Morton is a member of the LCS team. He has been an FAA Software DER since 2006, and was a member of the SC-205 Editorial Committee and Subgroup 3, the Tool Qualification subgroup. A recovering software tool designer, he has seen both sides of the software tool fence – as both developer and user.

Certificate Number FM 26376

www.ldra.com

LDRA

LDRA UK & Worldwide
Portside, Monks Ferry,
Wirral, CH41 5LH
Tel: +44 (0)151 649 9300
e-mail: info@ldra.com

LDRA Technology Inc.
Lake Amir Office Park, 1250 Bayhill Drive Suite # 360
San Bruno CA 94066 Tel: (650) 583 8880
e-mail: info@ldra.com

LDRA Technology Pvt. Ltd
#2989/1B, 3rd Floor, 12th Main, 80 Feet Road,
HAL II Stage, Bangalore- 560008. Near BSNL Building
Tel: +91 80 4080 8707
e-mail: india@ldra.com