Chapter 19
High-Integrity Model-Based Development

K. Lano
King’s College London, UK

S. Kolahdouz-Rahimi
King’s College London, UK

ABSTRACT
Model-Based Development (MBD) has become increasingly used for critical systems, and it is the subject of the MBDV supplement to the DO-178C standard. In this chapter, the authors review the requirements of DO-178C for model-based development, and they identify ways in which MBD can be combined with formal verification to achieve DO-178C requirements for traceability and verifiability of models. In particular, the authors consider the implications for model transformations, which are a central part of MBD approaches, and they identify how transformations can be verified using formal methods tools.

INTRODUCTION
Model-based development is the engineering of software systems using models, which may be graphical or textual in nature, but which should conform to some precise language definition. MBD is contrasted to traditional software development in which informal natural language text or low-level program code are the only artifacts. The formal nature of modelling languages used in MBD permits the definition of analysis and transformation tools which operate on models. In particular, model transformations are considered to be an essential part of model-based development, and they enable the automation of many operations on models:

- Mapping a model at one level of abstraction to a model at a different level (refinement, abstraction or reverse engineering).
- Comparing two models for differences or for consistency with respect to some relation (eg., different views of the same system).
- Generating code or other text from a model, such as test cases.

DOI: 10.4018/978-1-4666-6359-6.ch019
• Propagating changes from one model to another linked model.
• Migrating models from one version of a modelling language to another version.

Model-based development offers many advantages, including the ability to reuse models, to facilitate communication with non-software experts, and to reduce implementation costs by automating code generation.

For these reasons, MBD has had increasing uptake across many industry sectors, including high-integrity and safety-critical application domains such as avionics. In consequence, standards for these domains, such as RTCA DO-178 (RTCA, 2012), have begun to incorporate guidance on the use of MBD for critical systems.

The undisciplined use of models may actually decrease software reliability, if the models are developed without higher-level requirements (so that required functionality is omitted from the models) or without precise model analysis (so that the meaning of the model, when converted to code, is insufficiently understood). Therefore, DO-178C defines rigorous procedures for the use and validation of models. In this chapter we will consider the requirements of the DO-178C standard in detail, and provide techniques for MBD which aim to satisfy these requirements.

BACKGROUND

DO-178C is the successor to the RTCA/EUROCAE DO-178B standard “Software Considerations in Airborne Systems and Equipment Certification” (1992). DO-178C was published in 2012 and represents a significant extension and update of DO-178B. In particular, four supplements are added, giving detailed guidance on:

• Model-based development and verification.
• Object-oriented technology.

• Formal methods.
• Tool qualification.

Like DO-178B, DO-178C remains primarily a ‘process’ standard, whose intent is to define necessary properties of the software development process which ensure that (i) all required functionality of the software is correctly implemented, and (ii) that no unintended functionality is implemented.

For MBD this implies that all models should be derived from, traced back to, and verified against higher level requirements. Models themselves can be at different levels of abstraction, such as specification models (expressing high-level requirements) and design models (expressing low-level requirements).

The key attribute of a model in DO-178C is that it should be unambiguous. This means that the modelling language itself must have a well-defined syntax, and that a precise semantics exists which gives an unambiguous denotation to every syntactically valid model.

Tracing between models is of critical importance, in particular developers should ensure that automated transformations which derive one model from another (and source code from models) are able to provide detailed tracing information defining how elements in the target model are derived from elements in the source model. Tracing ultimately should relate model elements to some original system requirements, and source code elements to model elements.

DO-178C differentiates Specification Models (high-level requirements models) from Design Models (low-level requirements models): the former are unambiguous descriptions of the externally observable properties of the software system or component, such as its functional, performance and safety characteristics. They should not define internal software design details such as particular algorithms. In contrast, design models may define internal data structures, architectures