Adoption of Model-Based Testing and Abstract Interpretation by a Railway Signalling Manufacturer

Alessio Ferrari, University of Florence, D.S.I., Italy
Gianluca Magnani, General Electric Transportation Systems, Italy
Daniele Grasso, General Electric Transportation Systems, Italy
Alessandro Fantechi, University of Florence, D.S.I., Italy
Matteo Tempestini, General Electric Transportation Systems, Italy

ABSTRACT

Introduction of formal model-based practices into the development process of a product in a company implicates changes in the verification and validation activities. A testing process that focuses only on code is not comprehensive in a framework where the building blocks of development are models, and industry is currently heading toward more effective strategies to cope with this new reality. This paper reports the experience of a railway signalling manufacturer in changing its unit level verification process from code-based testing to a two-step approach comprising model-based testing and abstract interpretation. Empirical results on different projects, on which the overall development process was progressively tuned, show that the change paid back in terms of verification cost reduction (about 70%), bug detection, and correction capability.

Keywords: Abstract Interpretation, Code Generation, Model-Based Testing, Railway Signalling, Safety-Critical Software

INTRODUCTION

The adoption of modelling technologies into the different phases of development of software products is constantly growing within industry (Mohagheghi & Dehlen, 2010; Miller, Whalen, & Cofer, 2010). Designing model abstractions before getting into hand-crafted code helps highlighting concepts that can hardly be focused otherwise, enabling greater control over the system under development. This is particularly true in the case of embedded safety-critical applications such as aerospace, railway, and automotive ones, that, besides dealing with code having increasing size and therefore an even
more crucial role for safety, can often be tested only on the target machine or on ad-hoc expensive simulators. For these reasons, the industry of safety-critical control systems has been the first in line to adopt modelling technologies, such as the SysML™/UML® graphical languages (Object Management Group, 2010a, 2010b, 2010c) for high-level design, and modelling and simulation platforms like Simulink®/Stateflow®1 or the SCADE Suite®2 for lower-level design, to support the development of their applications.

General Electric Transportation Systems (GETS)3 develops embedded platforms for railway signalling systems and, inside a long-term effort for introducing formal methods to enforce product safety, employed modelling first for the development of prototypes (Bacherini, Fantechi, Tempestini, & Zingoni, 2006) and afterwards for requirements formalization and automatic code generation (Ferrari, Fantechi, Bacherini, & Zingoni, 2009). Within the new development context also the verification and validation activities have experienced an evolution toward a more formal approach.

In particular, the code-based unit testing process guided by structural coverage objectives, which was previously used by the company to detect errors in the software before integration, has been completely restructured to address the new model-based paradigm. The process refactoring has been driven by three main reasons:

- The traditional approach based on exercising the code behaviour resulted in being too costly to be applied to a code that saw a size increment of four times for the same project within two years. This fast growth was partly due to the increase of the actual projects size and partly to the code generators that, as known, produce more redundant code than the one that could be produced by manual editing;
- When an automatic tool is used to translate from a model to software, it has to be ensured that the latter is actually compliant with the intended behaviour expressed by the model;
- Unit testing alone, whether model-based or code-based, cannot cover all the possible behaviours of the code in terms of control flow and data flow. Most notably, it lacks in detecting all those runtime errors, such as division by zero and buffer overflow, that might occur only with particular data sets.

The restructured unit level verification process is based on two phases, namely model-based testing and static analysis by means of abstract interpretation. The first phase is used to exercise the functional behaviour of models and code, and, at the same time, to ensure that the synthesized code conforms to the model behaviour. The second phase is used to ensure that the code is free from runtime errors. Unit level verification costs were in the end reduced of about 70%, while notably decreasing the man/hours required for bug detection and correction.

The rest of the paper is structured as follows. In the first section we review the literature on industrial applications of abstract interpretation and model-based testing. The second section describes the unit testing process adopted, introducing the normative context of the railway industry and giving details on the verification phases. The third section analyses the results obtained applying the approach on two case studies, and the last one draws conclusions on the presented experience.

The article is an extended version of a conference contribution (Grasso, Fantechi, Ferrari, & Becheri, 2010) augmented with additional material from a recent case study (Ferrari, Magnani, Grasso, Fantechi, & Tempestini, 2010). Compared with the original work, the current one completes it with the following contents: (a) description of the evolution of the model-based testing phase, where a new tool, called 2M-TVF, has been introduced to counter the objections about the non-trusted code generator; (b) full description, with examples, of the abstract interpretation phase, previously presented only in terms of process tasks; (c) empirical results merged with those of the recent case study presented in (Ferrari, Magnani, Grasso, Fantechi,
On the Impact of Network Dynamics on a Discovery Protocol for Ad-Hoc Networks
www.igi-global.com/article/impact-network-dynamics-discovery-protocol/3883?camid=4v1a

Node Stability Index: A Stability Metric and an Algorithm to Determine Long-Living Connected Dominating Sets for Mobile Ad hoc Networks
www.igi-global.com/article/node-stability-index/64226?camid=4v1a