SmartTestGen+: A Test suite Booster for Enhanced Structural Coverage

S. Raviram[1], P. Peranandam[2], M. Satpathy[2], and S. Ramesh[2]

[1] GM Powertrain – India, GM Tech Center (India), Bangalore 560066
[2] India Science Lab, Global General Motors R&D, Bangalore 560066

Abstract. Our work concerns with test case generation for structural coverage of Simulink/Stateflow (SL/SF) models. We have developed a tool called SmartTestGen which integrates multiple test generation techniques; experiments show that this tool performs better than some commercial tools. In this paper, we discuss a novel experiment. SmartTestGen uses random testing as one of the testing techniques. The random testing component first generates random test cases; the tool then *extends* these test cases to cover the uncovered targets. In our experiment, instead of using the random test cases as the initial seed, we use the test cases of an existing test suite. We have evaluated the impact of this modified testing process by considering 20 industrial strength SL/SF models.

1 Introduction

The Simulink/Stateflow (SL/SF) modeling notation [1] is widely used in industry. We have performed experiments with many test case generation techniques – random testing, model checking, constraint solving, mix of random testing and constraint solving etc – on a number of industrial-strength models. We observed that the coverage achieved by the individual techniques in a broader sense complement each other. With this aim in mind, we have developed SmartTestGen [5], an integrated test generation environment which uses various test generation engines, each engine implementing a different technique. We have observed that SmartTestGen outperforms some of the commercial tools [5].

Even if SmartTestGen performs better than an existing tool, it is unlikely to replace such a tool in an industrial setting where hundreds of engineers use the existing tool for test case generation. Introduction of such a tool could be seen as a `disrupting technology`. In this context, we present in this paper a `supporting technology` involving SmartTestGen which is more likely to be accepted by engineers; this we outline in the following.

SmartTestGen has a random testing component which generates random test sequences; thereafter, other components of the tool extend the traces due to the random test sequences to cover the uncovered targets. We have performed a novel experiment on SmartTestGen. We replace the random test sequences generated by the random testing component of SmartTestGen by the test sequences of an existing test suite – possibly obtained by some other test generator. In other

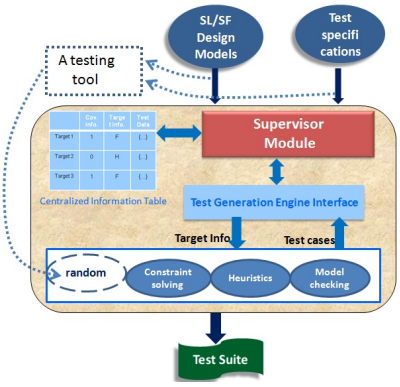

Fig. 1. `SmartTestGen` architecture: dotted lines show the modifications.

words, the traces due to the test sequences of the existing test suite are extended by other components of `SmartTestGen`. We refer to this new use of `SmartTestGen` as `SmartTestGen+`. Our main contributions:

- `SmartTestGen+` tool using an existing test suite as its initial seed: we argue that this extension is a `supporting technology` and hence can be adapted with ease in an industrial testing process.
- Evaluation of `SmartTestGen+` on 20 production quality industrial models: for our experiments, we have considered the test cases produced by a commercial tool, and compared its coverage results with those of `SmartTestGen+`.

2 SmartTestGen+ Architecture

Figure 1 shows the `SmartTestGen` architecture. This tool has three major components a) Centralized Information table (CIT) b) a set of test generation engines, and (c) the Supervisor module.

The CIT contains all the target candidates – based on the coverage criteria – to what extent they have been covered, what are the uncovered targets, and the test cases for the covered targets. In addition, the CIT also stores targets which has been shown to be unreachable.

`SmartTestGen` uses three test generation engines: (a) the `random` test generation engine to generate the initial test cases, (b) the `constraint solving and heuristics` engine which essentially extends the traces of the current test sequences by using a combination of constraint solving and heuristics [3], and (c) a SAL based `model checking engine` which is used to cover some given targets by using model checking. Heuristics are primarily used to cover targets when the constraints are non-linear or when the constraint size becomes too large [3].

The functionality of the `Supervisor module` is as follows: the `Random testing` engine produces the initial test cases. Depending on the nature of the targets, an appropriate test engine is selected for test case generation. This goes on till

all engines are invoked. The constraint solving and heuristics engine extends the earlier test cases (a) by performing constraint solving with respect to an intermediate point of a given test case [5], or (b) by using heuristics. Model checking engine has two tasks: (i) it tries to generate test cases to cover certain targets, and (ii) it also tries to show unreachability of certain targets.
For SmartTestGen+, we deactivate the random testing engine of SmartTestGen. We now use a pre-generated test suite as the initial cases; refer to Figure 1. We can assume that this test suite is produced by an existing testing method. Once we have the initial test cases, the SmartTestGen testing process is used to generate the subsequent test cases and the unreachability proofs. In Figure 1, the portion with the dotted lines illustrate this modification.

3 Experimental results

We consider the test cases already generated by the Reactis tool [2] as our initial test suite. Note that Reactis is a highly successful tool, widely used in industry for test case generation [2]. We then use SmartTestGen+ which enhances the above test suite for additional structural coverage. We have considered twenty SL/SF design models from various domains of automotive engineering such as Active safety (AS), Performance traction control (PTC), Powertrain (PT), Heating ventilation and cooling (HVAC) and Electronic stability control (ESC). The model sizes vary from 37 blocks to 901 SL/SF blocks. These models contain Stateflow blocks, multi-dimensional inputs, legacy code, non-linear blocks like multiplication and division, dynamic lookup tables and hierarchical triggering of blocks. Simulink Verification & Validation (V & V) tool box [4] is our common measuring platform. All the experiments were carried out on a machine with Intel Xeon 3 GHz and 3.5 GB RAM running Windows XP professional. The tool versions used were: Reactis 2009.2 and Matlab R2011.2.

We have compared the results of SmartTestGen+ with those of Reactis when used independently. The graphs in Figure 2 respectively show the comparison results for the decision, condition and MC/DC coverages of all the 20 models. For each model, the first bar shows the coverage by Reactis, and the second shows the coverage of SmartTestGen+. with the SmartTestGen+ achieves (a) better decision coverage than Reactis in 50% cases, (b) better condition coverage than Reactis in 33% of the cases, and (c) better MC/DC coverage than Reactis in 55% of the cases. In the remaining cases the coverage results are equal.

Using the model checking engine, we have verified that the remaining decision targets of *PT1*, *HVAC2* and *PTC3* models are un-reachable. Similar advantages are also observed in case of models *PT1*, *HVAC2* and *PTC3* for condition coverage, and in case of *PT1* and *HVAC2* models for MC/DC coverage. The Reactis tester as of now does not address the issue of unreachability.

4 Summary

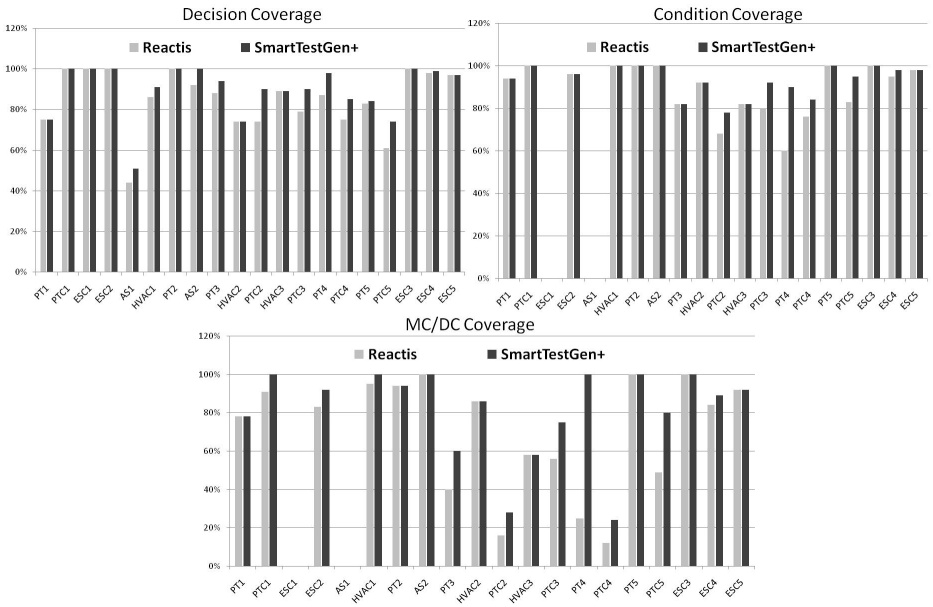

Fig. 2. Comparison of Decision, Condition and MC/DC coverages between Reactis, SmartTestGen and SmartTestGen+

- We claim that introduction of `SmartTestGen+` in an industrial setting is not a `disrupting technology` but a `supporting technology`. Consider a scenario in which the testers use the existing test generation tool to produce test cases. Based on the criticality of the applications, the test cases of this tool could be fed to `SmartTestGen+`, and test cases with higher coverage would possibly be obtained. All the testers need not learn `SmartTestGen+`, only a small percentage can use it. In this sense, the original testing process would be minimally affected.
- In case of automotive or aerospace domains, many applications are safety-critical. Therefore the added coverage we obtain would increase the reliability of the applications.

References

1. The Mathworks. Available at: http://www.mathworks.com.
2. Reactis. Available at: http://www.reactive-systems.com.
3. Satpathy M, Yeolekar A, Ramesh S. 2008. Randomized Directed Testing (REDIRECT) for Simulink/Stateflow Models, ACM EMSOFT'08, Atlanta.
4. The Mathworks, Verification and Validation Tool Box. Available at: http://www.mathworks.com/verification-validation/.
5. Paranandam P *et al.* 2012. An Integrated Test Generation Tool for coverage of Simulink/Stateflow Models, Proc. of IEEE DATE'12.