Debugging Embedded Systems Requirements Before The Design Begins *

* “The beginning is the most important part of the work” - Plato

Fabien Gaucher
ARGOSIM SA
8-10 rue de Mayencin, 38400 St Martin d’Hères, France
fabien.gaucher@argosim.com

Yves Génevaux
ARGOSIM SA
8-10 rue de Mayencin, 38400 St Martin d’Hères, France
yves.genevaux@argosim.com

Abstract

Over the last two decades, there has been a real movement towards requirements engineering [1]. However, the various tools that have been developed with the aim of improving the specifications development are mostly focused on requirements management and traceability, and there was no easy-to-use requirements validation tool available for checking their functional consistency before any design or coding takes place. As a result, 40 to 60% of design bugs are caused by faulty requirements that generate costly iterations of the development process.

Argosim’s ambition is to change this situation by providing STIMULUS [2], the very first requirements simulation tool for real-time embedded systems. STIMULUS combines the ability to write requirements into a formal yet simple textual language, but also and even more importantly, to simulate them by generating the possible behaviours of the specified system in the form of execution plots. The analysis of simulation results allows for the quick and early detection of ambiguous, incorrect, incomplete or even conflicting requirements.

In practice, STIMULUS integrates seamlessly into the agile development processes [3]. It allows for the concurrent development and validation of both the requirements and their test scenarios in a very incremental way.

Once validated at the specification level, both requirements and test scenarios can be reused to validate the design implementation. Test scenarios make it possible to generate many test vectors automatically, while requirements are reused as test observers, or test oracles, in order to automatically check that the system satisfies its requirements.

This paper describes the major innovation offered by STIMULUS in the field of requirements engineering. It will first present its operating principles and then briefly illustrate both the requirements validation and software validation workflows with an example from the automotive domain.

Keywords Requirements Engineering, Textual Specifications, Simulation, Constraint Solving, Debugging, Validation, Automatic Testing, Real-time Embedded Systems.

SIGPLAN’05 June 12–15, 2005, Location, State, Country.

1. STIMULUS operating principles

Following a number of recent research results in the field of synchronous languages and verification methods, notably carried on in the work of the VERIMAG laboratory [4] [5] [6], Argosim released STIMULUS, a highly innovating tool which specifically addresses the challenge of debugging functional real-time requirements by applying the two following basic principles:

- STIMULUS considers requirements as constraints imposed on the specified system behaviour. A constraints solver is therefore able to compute the space of possible behaviours at a given execution cycle, and many execution paths can be explored by using a random generator. STIMULUS then produces classical simulation plots without needing neither a design model nor a computer code.
- In order to make STIMULUS easily available to requirements practitioners who are used to express requirements in natural language, the requirements are formatted in text using sentence templates [7] [8]. Since each sentence template is also formally defined by means of constraints, STIMULUS makes it possible to write requirements that look very similar to their original formulation, while being executable.

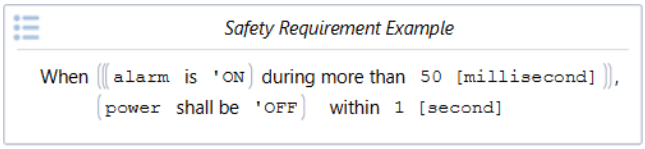

Figure 1. A formal textual requirement written in STIMULUS

The edition of formalized textual requirements is a key asset of the tool. Figure 1 gives an example of a simple requirement written in STIMULUS. It was edited by drag-and-dropping several predefined templates, namely When, DuringMore and Within. The set of predefined templates is not fixed and the user may build his own domain-specific templates by composing existing ones or by using basic constraints, and then by associating it to some natural language sentence with “gaps”.

In addition to the textual language, STIMULUS also provides modelling features like state machines and blocks diagrams. The first allows to describe operational modes of the system, while block diagrams supports either a functional or architectural system decomposition, with clear interfaces presenting the signals exchanged between sub-systems.

The blocks hierarchy eases the refinement of high-level requirements into low-level requirements as well as the allocation of those latter requirements to the various sub-systems.

2. Early validation of system requirements

Figure 2 defines a safety requirement that is part of a simple headlights control system coming from the automotive domain. This requirement interacts with few other requirements in order to specify the behaviour of the automatic mode that shall, among others, prevent flashing whatever the light intensity is.

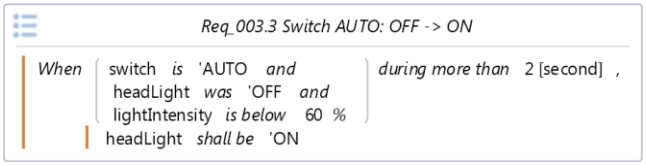

Figure 2. One of the requirements that switch the light 'ON'

Before simulating the requirements, one can specify basic assumptions on the light intensity as well as the switch command or, in other words, describe test scenarios for the requirements. These test scenarios are connected to the set of requirements using a block diagram similar to the one depicted in Figure 3.

Test scenarios are specified using a combination of state machines and textual constraints, allowing to test many variants of the same scenario, including time variability, different limit values, etc. For instance, the Loop75To55 scenario induces a light intensity fluctuation between 75 and 55%, with bounded derivate.

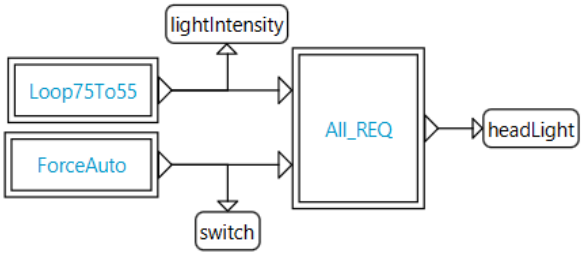

Figure 3. Connect requirements (All_REQ) to test scenarios

When simulated, the block diagram produces the plot of Figure 4. In this particular execution, there is a violation of the high-level requirement (headlights shall not flash) by a low-level requirement. Each time STIMULUS reports a conflict, it also identifies automatically the minimal set of conflicting requirements.

The simulator's debugging features (error messages, highlight of active and inactive requirements, access to the various signal values at any time of the execution, etc.) allows users to thoroughly analyse the conflict, identify its cause in order to modify the faulty requirements or add missing ones.

3. Validation of the system implementation

The various requirements and test scenarios can be reused in order to validate the system implementation, once it is available.

The principle is to:

- Substitute the requirements with the compiled code (wrapped in some STIMULUS external system) in the block diagram so that, during simulation, the code will be stimulated by the same test scenarios as the requirements.
- Check that the compiled code behaves as specified by turning the requirements into observers, or test oracles. STIMULUS will then automatically report any requirement violation occurring during simulation.

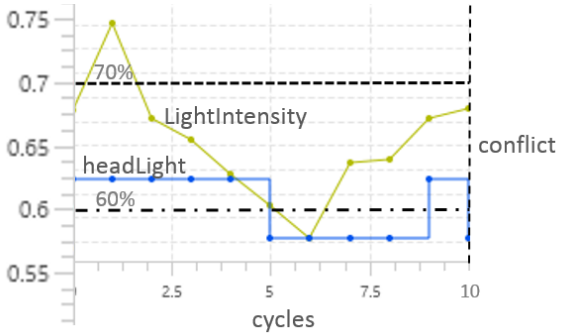

Figure 4. Simulation results that exhibit a conflict

4. Conclusion

In this paper, we have presented STIMULUS, the first effective tool that brings simulation capabilities at the specification level to allow for the early validation of functional real-time systems requirements. This tool offers totally new and easy-to-use features for system architects to make requirements right the first time and validation engineers to check automatically that the system implementation satisfies its requirements.

The ability to formalize textual requirements which are close to the natural language, while being perfectly defined, allows for development teams and stakeholders to share non ambiguous requirements that can fit to the specific application domain as well as to the culture and usages of the company. This textual language is also very useful for describing non deterministic test scenarios that will generate an envelope of test vectors within the possible behaviours.

These features are very complementary to major model-based design tools for embedded systems such as Scade or Simulink, for which STIMULUS provides connections that enable the export of generated test vectors and requirements observers.

References

[1] K. Pohl: Requirements Engineering: Fundamentals, Principles, and Techniques, Springer Publishing Company, Incorporated, 2010.

[2] B. Jeannet, F. Gaucher: Debugging Real-Time Systems Requirements: Simulate The "What" Before The "How", Embedded World Conference, Nürnberg, Germany, 2015.

[3] P. Abrahamsson, J. Warsta, M. T. Siponen and J. Ronkainen: New directions on agile methods: a comparative analysis, Proceedings of the 25th International Conference on Software Engineering, IEEE Computer Society, Washington, USA, 2003.

[4] P. Raymond, Y. Roux and E. Jahier: Lutin: A language for Specifying and Executing Reactive Scenarios, EURASIP Journal on Embedded Systems, 2008.

[5] E. Jahier, N. Halbwachs and P. Raymond: Engineering Functional Requirements of Reactive Systems using Synchronous Languages, International Symposium on Industrial Embedded Systems, IEEE, Porto, Portugal, 2013.

[6] S. P. Miller, A. C. Tribble, M. W. Whalen and M. P. E. Heimdahl:, Proving the shalls: Early validation of requirements through formal methods, International Journal of Software Tools for Technology Transfer, 2006.

[7] J. Davis, L. Wagner, J. Hoffman, K. Gross and A. Fifarek: SpeAR v2.0: Specification and Analysis of Requirements, Safe and Secure Systems and Software Symposium (S5), 2016.

[8] A. Mavin and al.: EARS (Easy Approach to Requirements Syntax), 17th IEEE International Requirements Engineering Conference, Atlanta, Georgia, USA, 2009.