D6.4 Abstract

Author: IB Krates

ESA contract AO/1-8376/NL/15/Cbi

Introduction

The current document contains a concise summary of the work performed by IB Krates and the results obtained within the project “Qualifiable code generation backend for TASTE”, ESA contract AO/1-8376/NL/15/Cbi. Further details can be found in the full report D6.1.

About QGen

QGen (www.adacore.com/qgen) is an open source qualifiable and tunable code generation and model verification toolset that grew out of the European projects Project-P (http://www.open-do.org/projects/p), Hi-MoCo and Gene-Auto. QGen is developed jointly by IB Krates and AdaCore. QGen is intended for use in safety-critical domains. Its development and qualification according to the avionic guidelines DO-178C/ED-12C and DO-330/ED-215 is currently ongoing. This process is generally well aligned with the requirements in the relevant ECSS standards ECSS-E-ST-40C and ECSS-Q-ST-80C.

Code generation from SDL models

As the SDL language is already used in many projects using TASTE, most emphasis was put on the support of SDL code generation. There were two SDL-related case studies. The first one was an academic model of medium size and complexity. Ada and C code generation from the case study models was achieved and functional behaviour was verified in functional tests and in tests comparing the behaviour with that of OpenGEODE generated code. The second case study was the testsuite of the OpenGEODE tool comprising of nearly 70 models focused on testing various features of the SDL and ASN.1 languages. The QGen SDL code generation support implemented during the project reached the state, where 68% of the tests pass with QGen generated Ada code and 65% with QGen generated C code. Most of the remaining tests are related to less commonly used and/or corner-case SDL and ASN.1 features that might be also expressed otherwise.

Supported SDL and ASN.1 features

The QGen SDL code generator supports a subset of SDL features supported by OpenGEODE, which itself supports a subset of SDL 92 with some additional features that were introduced in SDL 2000 or later.

The features supported by QGen include:

- States
- Transitions
- Hierarchical OR / AND state compositions
- Joins and Labels
- Local variable declarations, using natively ASN.1 types
- Inner procedures
- Calls to external procedures with in and in/out parameters
- Sending of asynchronous messages to the environments
- Executing actions in transitions:
    - Base actions (assignment, conditional, for loop)
    - Common binary/unary expressions
- Decision nodes and graphs
- Named and unnamed return and stop nodes
- Timers
- Dash states
- Continuous signals

The ASN.1 features supported by QGen include:

- Basic types: boolean, byte, integer, real
- Sequence of (variable and fixed size) type
- Sequence (structures) type
- Choice type
- Empty type
- Value assignment
- Optional fields and default values
- ASN1SCC generated ‘equal’ functions are used
- Data type definitions are imported from ASN1SCC XML dump

Following the TASTE architectural modelling philosophy it is possible to create heterogeneous models that has components implemented in SDL and other languages that provide a certain C language interface for binding the respective Provided and Required Interfaces of the components. The VDM2SPARK code generator implemented in the current project supports this pattern. To ensure that the data types at the interfaces are compatible they must be defined using a common ASN.1 specification for both models.

Code generation from VDM-SL models

The VDM-SL support implemented in this project remained more experimental. However despite that a relatively large amount of VDM-SL features were implemented in the VDM2SPARK code generator prototype. The technical approach benefited from the Overture toolset for parsing and type checking and from QGen for Ada (SPARK) code generation. However many new features were also added to the QGen code generator to support, e.g. the map and set container types in VDM-SL. The code generation support was verified and validated on two case studies. First, the small design-by-contract case study from the Finnish nuclear industry [1] and secondly, the well-known AlarmSL demo model by

John Fitzgerald and Peter Gorm Larsen [2]. The first case study was supported and verified in a rather straightforward way. The second case study contained a much larger set of VDM-SL features. However, with a few workarounds and adaptation the core logic of the AlarmSL model was also successfully handled by the VDM2SPARK code generator.

Additionally, specific support was added in VDM2SPARK to allow integrating the generated code into an heterogeneous model-based application combined and managed by the TASTE framework. For that purpose the support of ASN.1 data types was added to VDM-SL models and the VDM2SPARK code generator. Hence, it is now possible to e.g. call VDM-SL functions and operations from SDL models.

VDM-SL feature support

- VDM-SL basic types: real, int, nat, nat1, bool.
- Composite types (records)
- Unions of quote types (enumerations)
- ASN.1 type definitions for the same subset
- Token types with literal values
- Flat and multi-module specifications
- Common unary and binary operators
  (See the next section for the supported operators)
- State and value definitions
- Assignments, Let statements
- Operations and explicit function definitions
- Pre- and postconditions
- Map and Set types. Based on Ada Functional Containers
  (See the next section for the supported operators)
- Sequence types based on ASN.1 / ASN1SCC. Limited support

Dedicated support was implemented for defining custom data types using the ASN.1 data specification language, using these in VDM-SL models and generating code that leverages from the dedicated data manipulation (encoding and decoding) support provided by the ASN1SCC (https://github.com/ttsiodras/asn1scc) ASN.1 compiler.

Additionally, it is possible to generate the top-level interface of the code from VDM-SL models in such a way that it can be interfaced with external C code. The particular code generation pattern has been designed to be used with the TASTE (http://taste.tuxfamily.org/) architectural modelling framework.

Integration with modelling tools

Preparations were made for integrating QGen with the OpenGEODE and TASTE tools for code generation and simulation/debugging of SDL models. As a first step QGen can be launched for code generation from SDL models directly from OpenGEODE. Using QGen as a simulation backend from OpenGEODE would require a moderate additional development - the internal chart data structures generated by QGen must be complemented with an additional interface that would allow to map the chart’s state into an OpenGEODE compatible form. This remains for future work.

Support for design-by-contract workflow

Apart from the general code generation technology a broader design-by-contract approach was investigated during the project. In these approach the VDM-SL language was used for specifying the software in a high-level formalism, whereas the detailed algorithm design was performed later in Simulink. Further, the AADL language is used to allocate the designed algorithms into the overall system architecture. A possible application of the methodology was demonstrated on the Finnish nuclear industry use case mentioned above. The use case focused on the VDM-SL and Simulink parts. Adding the AADL layer remains for future work. A prototype implementation was developed that converts a VDM-SL specification to a “template” Simulink model that includes generated observer blocks that implement the semantics of dedicated parts of the VDM-SL specification. Thus, it is possible to implement the lower level design in Simulink and verify its correct behaviour during Simulation and/or formal analysis of the generated code by the Ada SPARK toolset.

Improved mapping between TASTE AADL models and Simulink

An improved mapping of TASTE AADL models to Simulink was proposed that provides a more natural representation of the the architecture of multi-component models in Simulink with improved support for roundtrip development. A potential scheme for implementing this mapping has also been outlined. The implementation and validation of this approach remains a subject for future work.

References

- [1] Pakonen, A; Tahvonen, T; Hartikainen, M; Pihlanko, M, “Practical applications of model checking in the Finnish nuclear industry”, Proceedings of the 10th International Topical Meeting on Nuclear Plant Instrumentation, Control and Human Machine Interface Technologies, NPIC & HMIT 2017, 11 - 15 June, 2017, San Francisco, CA, USA.
- [2] John Fitzgerald and Peter Gorm Larsen, Modelling Systems – Practical Tools and Techniques in Software Development}, Cambridge University Press, 2nd edition, 2009