Evaluation of validity of verification methods

Oskar Ingemarsson, Sebastian Weddmark Olsson

Chalmers University of Technology, Mecel AB
Supervisor: Josef Svenningsson
Supervisor Mecel: Johannes Rehnman

August 18, 2014

Overview

Introduction

Background

- Circa 80 ECUs
- Millions of lines of code
- Vehicles become more and more complex
- More functionality
- Safety critical
- Testing more then half the production cost

Glossary

Functional Safety (according to ISO 26262)

Glossary

Functional Safety (according to ISO 26262)

ASIL (Automotive Safety Integrity Level)

Introduction

Objective

- Automation of tests of modules in AUTOSAR
- Examine if it is possible to reach the requirements for a higher ASIL classification, thus fulfilling more requirements within the concept of functional safety.

Watchdog Manager

AUTOSAR specification

"The Watchdog Manager supervises the execution of a configurable number of so called Supervised Entities. When it detects a violation of the configured temporal and/or logical constraints on program execution, it takes a number of configurable actions to recover from this failure."

Glossary

Functional Safety (according to ISO 26262)

ASIL (Automotive Safety Integrity Level)

Supervised entity (Critical section)

Checkpoint (Point in these critical sections)

Functions

Getters

- GetVersionInfo(VersionInfoType*)
- GetMode(ModeType*)
- GetLocalStatus(SupervisedEntityIdType, LocalStatusType*)
- GetGlobalStatus(GlobalStatusType*)
- GetFirstExpiredSEID(SupervisedEntityIdType*)

Setters (interesting)

- Init (ConfigType*)
- DeInit ()
- PerformReset()
- SetMode(ModeType, uint16)
- MainFunction()
- Checkpointreached(SupervisedEntityIdType, CheckpointIdType)

Supervision mechanisms

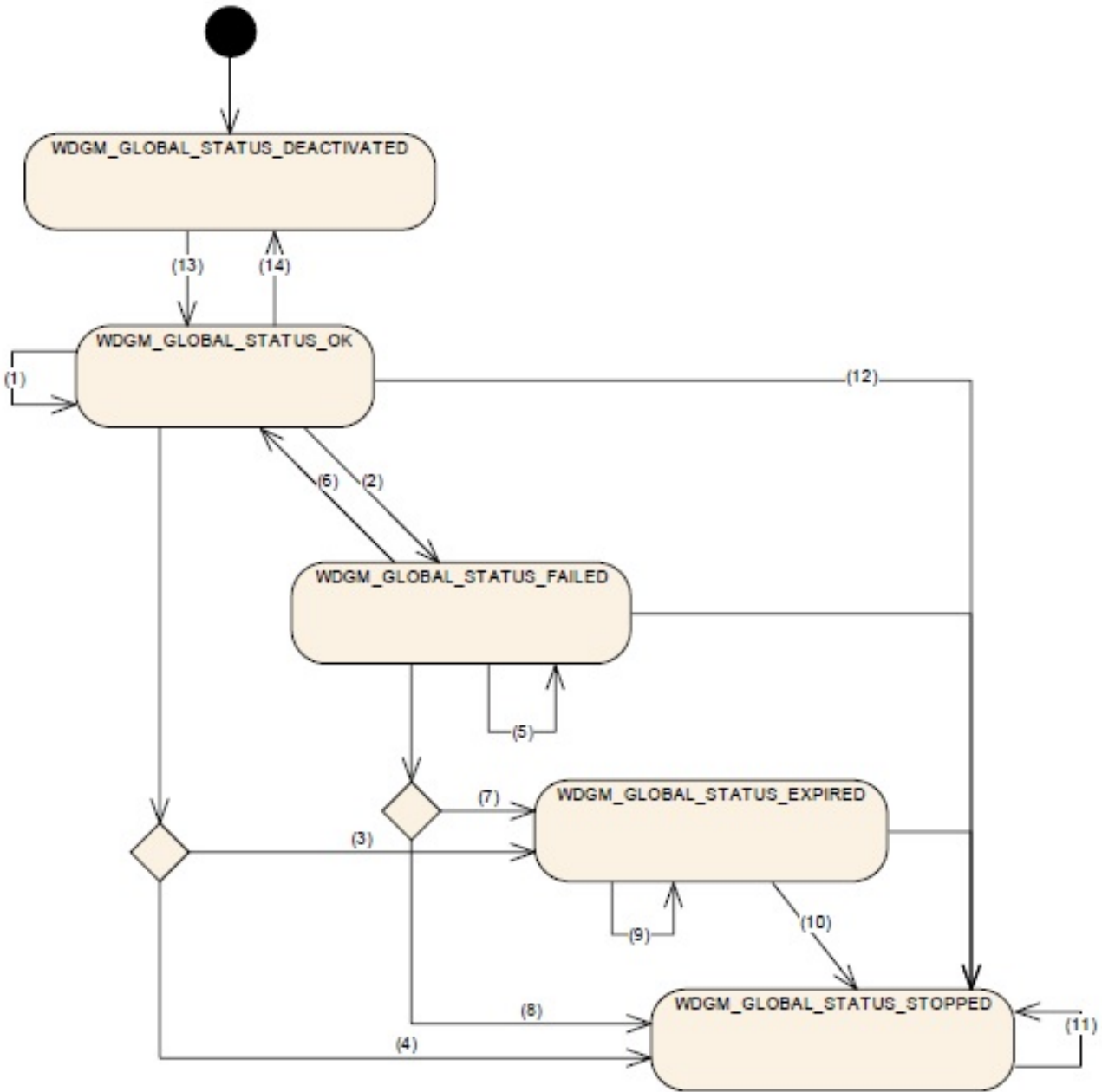
- Alive supervision
- Deadline supervision
- Logical supervision

The state machine

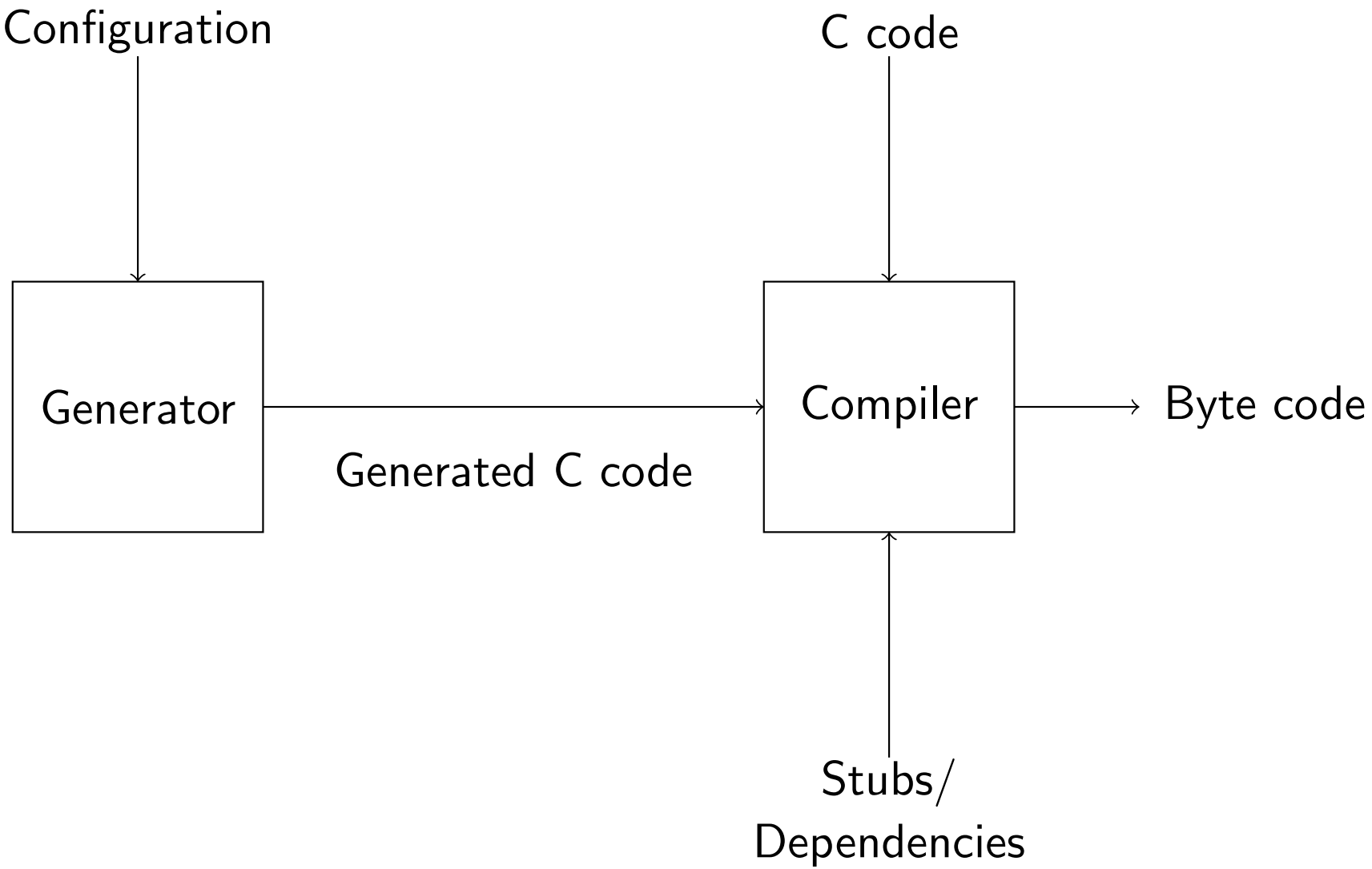
Global status (for the watchdog manager)

Local status (per supervised entity)

The state machine



Generating the C byte code



Glossary

Functional Safety (according to ISO 26262)

ASIL (Automotive Safety Integrity Level)

Supervised entity (Critical section)

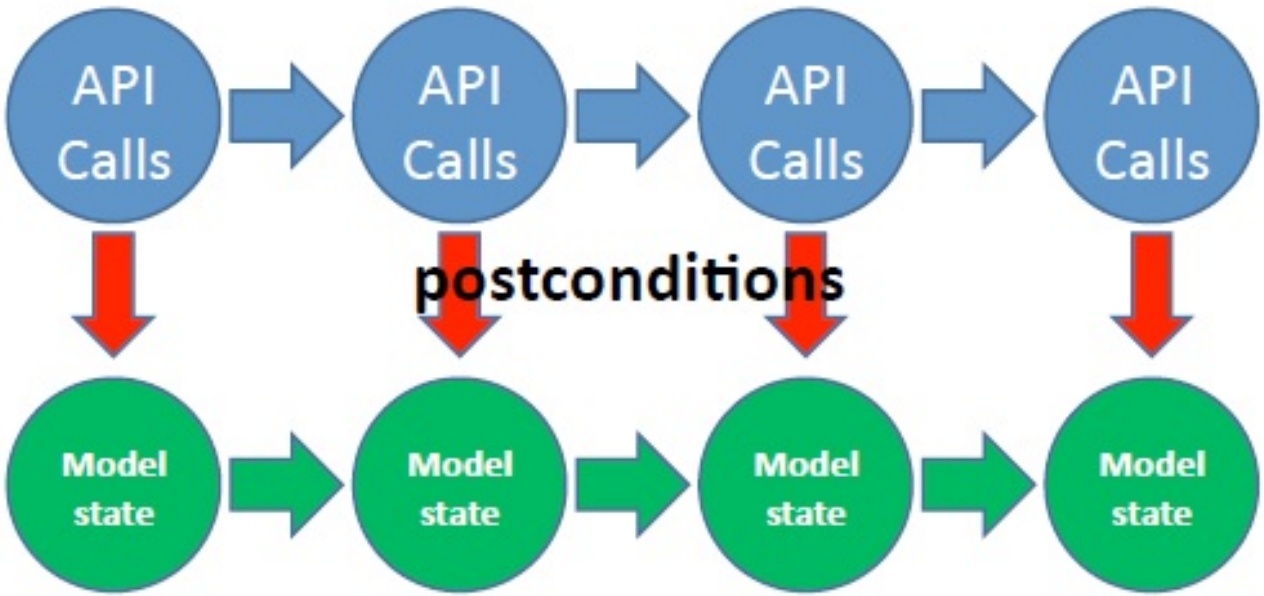
Checkpoint (Point in these critical sections)

QuickCheck (A commercial testing tool)

QuickCheck

- Model based testing tool

How QuickCheck works



QuickCheck Properties

Properties

Quickcheck uses properties that can verify that AUTOSAR requirements are not violated.

Post condition example

```
getmode_post(S, [Is_Null], {Return_Value, Mode}) ->
  DevErrorDetect = S#state.originalCfg#wdgmgeneral.dev_error_detect,
  case Return_Value of
    0 -> Mode == S#state.currentMode; %% [WDGM170]
    1 -> DevErrorDetect andalso (Is_Null orelse %% [WDGM254]
                                 not S#state.initialized) %% [WDGM253]
  end.
```

Generating test cases

```
weight(S, _init) ->
  case S#state.initialized of
    true                          -> 1;
    _                             -> 200
  end;
```

Configurations

Three different configurations

BSI	minimal configuration
Example	complex configuration
Freescale	realistic configuration

Testing

Positive testing

- No absorbing state
- Correct arguments
- Correct command sequences

Negative testing

- Invalid arguments
- Null pointers
- Absorbing state

Bugs

- “Fix” the model
- Fix C-code
- Mocking function with bug

Bugs we found

C code

≈ 10

Generated C code

≈ 2

AUTOSAR complications

- ambiguous meaning
- conflicting requirements
- omitted requirements/explanations
- spelling of configuration parameters
- references

Coverage & Statistics

Coverage

- Line coverage of Erlang code (97.38%)
- Function coverage of the C code (100%)
- Condition/Decision coverage of C code (81%)
- State space coverage (100% of valid transitions, 0% of invalid transitions)
- Requirements coverage ($\approx$ 60%)

Statistics

- Generated command sequences of up to 120 commands (really no limit)
- 95% starts with the command `WdgM_Init`

Development Error Detection

Dangerous configuration parameter.

Our Testing Tool

- Automatically generates relevant test cases
- Combines requirements and tests them together
- Configuration independence
- Designed for integration with other modules

Future work

Integration testing.
More configurations = better test results.

Conclusions

- QuickCheck and functional safety.
- AUTOSAR as a standard.
- Complexity of configurations.
- Measurements of state space, and code coverages.
- Implementation time for finding bugs.

Questions?