Software Safety Architecture
Designed for Testability

Melanie Cossy
STZ Softwaretechnik, Esslingen, Germany
www.stz-softwaretechnik.de
melanie.cossy@stz-softwaretechnik.de

Introduction
In modern vehicles more and more functionality depends on electronic components and software. Examples are systems like the electronic stability program or the electronic throttle control. To guarantee that a vehicle never endangers humans, objects or the environment, it must be ensured that the different electronic systems integrated into a vehicle behave safe – even if failures occur. This implies that the electronic systems must contain safety-related software – in addition to the normal functional software – to detect and handle failures during operation. The correctness of the safety-related software parts is very important for the overall vehicle safety and therefore it is necessary that the car manufacturer can test them thoroughly.

Due to the short development cycles the good testability of the safety-related software is extremely important. Good testability improves the quality of the electronic system and the quality of the vehicle itself. High test coverage values minimize the probability of cost intensive and image harming recall campaigns. To increase the test coverage different teams must be able to test the safety-related software concurrently. Automated test execution must be possible.

In the following it will be shown how the testability of the safety-related software can be greatly enhanced by considering testing requirements already during the design phase.

Traditional testing approach
The traditional approach to test the safety-related behavior of an electronic system is to provoke faults and to observe how the electronic system reacts to handle the fault. In the past these tests have been executed after the integration of the electronic system into the car by comparing the observed behavior with the specified behavior. Nowadays hardware-in-the-loop test benches are used to test the different electronic systems independently. Hardware-in-the-loop test benches allow the automated test execution and reduce the required testing time and costs substantially.

During hardware-in-the-loop tests the electronic system is considered as a black box. But black box testing can become challenging, if the complexity of the electronic system and its safety-related software increases. The safety-related software of a brake-by-wire system for example must be able to cope with more than 500 different safety-related failures. This implies that more than 500 different failure detection algorithms must be tested. In addition it is necessary to specify and test the correct handling of an extremely huge number of failure combinations ($2^{500}$).
To test such a complex electronic system the traditional black box approach cannot be used. Instead the safety-related software must be decomposed into smaller modules that can be tested independently.

**Safety Architecture**

The main idea of the safety architecture is the clear separation between error detection and error handling. Monitors are responsible for error detection. They identify incorrectly working system components and generate fault notifications. The fault notifications are the input information of the safety interface modules that determine the current status of the monitored hardware and software components. Safety control modules use the component status information to select an appropriate reconfiguration strategy that can be applied handle the erroneous system components. To control the reconfiguration process the safety control modules determine the current execution modes of the functional software.

The fault notifications, the component status information and the execution modes are internal data/control flows of the electronic system. By making these data/control flows observable and manipulatable for testing purposes the testability of the safety-related software can be greatly enhanced. The different safety-related software modules can be tested separately and the test responsibilities can be clearly assigned to persons or teams (see Table 1).

<table>
<thead>
<tr>
<th>Team</th>
<th>Responsibility</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component team</td>
<td>Responsible for a component and the detection of its failures.</td>
</tr>
<tr>
<td></td>
<td>To test the detection algorithm and the parameterization of a monitor, a real hardware fault is provoked (with a hardware-in-the-loop test bench) and it is checked whether the monitor under test generates an appropriate fault notification.</td>
</tr>
<tr>
<td>Safety team</td>
<td>Responsible for the safety logic (decomposed into safety interface and safety control modules) that determines the correct reconfiguration strategy. The safety logic does not depend on the underlying hardware. Therefore it is possible to use a software module test environment for the verification of the implemented safety logic. If an executable specification is available, the verification of the safety logic can be done very efficiently through back-to-back tests.</td>
</tr>
<tr>
<td>Function team</td>
<td>Responsible for a module of the functional software and its different execution modes. To verify a functional software module and its different execution modes in various driving situations, the execution modes are enforced with the help of a special test interface that allows software manipulations while driving the car.</td>
</tr>
</tbody>
</table>

Table 1 Typical allocation of tasks and responsibilities

**Conclusion**

The design of the safety architecture enables car manufacturers to completely specify the safety-related software and to efficiently execute test series that ensure that the implementation of the safety-related software fulfills its specification. To achieve this objective the safety architecture decomposes of safety-related software into modules to clearly defined interfaces that enable the tester to observe and manipulate the information flow through the system during testing.
Software Safety Architecture
Designed for Testability

Conference on Testing Embedded Systems
Bilbao, November 24-25, 2003

Dipl.-Ing. (FH) Melanie Cossy, MSc
STZ Softwaretechnik
Esslingen, Germany

www.stz-softwaretechnik.de
melanie.cossy@stz-softwaretechnik.de
Content

- Motivation
- Software Safety Architecture
- Testing the Modules of the Safety Architecture
- Conclusion
Motivation

- In modern vehicles more and more safety-related functions depend on electronic systems controlled by software.

  ⇒ It must be ensured that the electronic systems always behave safe – even if failures occur.

  ⇒ Special safety software is required to detect and handle failures during normal operation.

- To guarantee that a vehicle never endangers humans, objects or the environment, the car manufacturers must ensure that the safety-related software of the electronic systems is correct.
Motivation

Traditional testing approach:
- In the past tests have been executed by integrating the electronic system into a car.
- Nowadays hardware-in-the-loop test benches are used to test the electronic systems independently.
Motivation

Traditional testing approach:
- To test the safety-related software, hardware faults are stimulated and the reaction of the electronic system is observed.
- The electronic system is considered as black box and therefore the observable behavior is limited.
Motivation

Testing challenge:
- Testing the whole electronic system as a black box is challenging, if safety-related software is complex.

- An Electro-Hydraulic Brake System for example must be able
  - to detect more than 500 different failures and
  - to handle $2^{500}$ different failure combinations.

- To test the correct handling of this huge number of failure combinations the traditional black box approach cannot be used.
**Motivation**

- The ensure the correctness of the safety-related software large amounts of tests are necessary.

⇒ It must be possible:
  - to automate test execution
  - to execute tests in parallel
  - to test the different modules of the safety-related software independently

- To achieve this, it is necessary that testing objectives are considered already during the design of the software architecture.
Software Safety Architecture

Requirements:

- The software safety architecture must be able to fulfill the safety and reliability requirements of an electronic system embedded in an automobile.

- The decreasing development times require the applicability of concurrent and highly efficient engineering methods during specification, development and test.

- It must be possible to specify and test the correct handling of all failure combinations.
Software Safety Architecture

To fulfill these requirements the software safety architecture:

- separates the safety-related software parts from the functional software parts.

- decomposes the safety-related software into modules with clearly defined tasks and interfaces.

- provides special test interfaces that allow the observation and manipulation of internal data and control flows for testing purposes.
The safety architecture separates the functional software from the safety-related software and organizes the safety-related software into error detection and error processing modules.
Software Safety Architecture

- **Monitors**
  detect failures of safety-related hardware or software components and emit fault notifications.

- **Safety Interface Modules**
  represent the safety-related HW/SW components and provide status information about these components.

- **Safety Control Modules**
  control the execution of safety measures by determining the execution modes of the execution modules.

- **Execution Modules**
  are mode driven and realize the main functions of the application.
Software Safety Architecture

- To make the internal data and control flows observable, special test interfaces are necessary.
- Such a test interface can be implemented very efficiently by using the CAN calibration protocol, which has been standardized by the Association for Standardisation of Automation- and Measuring Systems.

Motivation

Safety Architecture

Testing

Conclusion
Testing the Monitors

- The correctness of the monitors depends on the underlying hardware.
  - Therefore hardware-in-the-loop test benches are used to test the different monitors.

- To test a monitor a real hardware fault is provoked and it is checked whether or not an appropriate fault notification is generated.

- The advantage of the safety architecture for such tests lies in the possibility to test individual monitors directly.
  - The desired reaction can be specified as the fault notification that must be generated from the monitor under test.
The Safety Interface Modules and Safety Control Modules are specified as state machines using an appropriate graphical tool.

The main advantage of such a tool is that the specification is executable. This facilitates the automated testing of the implementation against the specification.
Testing the Safety Interface and Control Modules

- The safety interface and safety control modules do not depend on the underlying hardware.
  ⇒ It is possible to test them very efficiently in a software module test environment.

- Back-to-back test environment for the safety interface and safety
Testing the Safety Modules

There are various ways how the test input stimuli can be generated:

- manual, interactive generation of fault notifications with the help of a graphical user interface
- extraction of the fault notifications from a real-life measurement in the vehicle
- systematic generation of fault notifications
Testing the Execution Modules

- Most of the software modules can be tested in a software module test environment or with a hardware-in-the-loop test bench that simulates the car, the driver and the environment.

- But to verify the functionality of the execution modules tests in the vehicle are still necessary.

- The different execution modes of an execution module must be verified in all possible driving situations.

- This can be done by using a test interface that allows the manipulation of the execution modes via the CAN Calibration protocol.
**Conclusion**

- The presented safety architecture has been designed to enable car manufacturers
  - to completely specify the safety-related software of the different electronic systems and
  - to efficiently test an implementation to ensure that it fulfills the specification.

- To achieve this objective the safety architecture decomposes the safety-related software into clearly defined modules that can be tested independently.
Conclusion

- By testing the different modules of the safety-related software independently,
  - the complexity of the tests can be reduced and
  - the test responsibilities can be clearly assigned to various persons or teams.

- The proposed modules and interfaces of the safety architecture
  - enable car manufacturers to significantly increase the scope and the coverage of the executed tests
  - and therefore help to improve the quality of electronic systems.