Design Trade-offs for Systematic Tests of Embedded Communication Systems

Eric Armengaud, Andreas Steininger
Vienna University of Technology; Embedded Computing Systems Group E182-2
{armengaud, steininger}@ecs.tuwien.ac.at

Martin Horauer
University of Applied Sciences, Technikum Wien
horauer@technikum-wien.at

Roman Pallierer
Dependable Computer Systems GmbH
pallierer@decomsys.com

1. Introduction

Embedded computer systems are currently revolutionizing the automotive industry [1]. Mechanical solutions are gradually being replaced by microcontrollers, and what has formerly been a local sensor, actuator or control system is now becoming part of a complex distributed system. The power of this approach lies in the interaction of the distributed components: Information of sensors distributed all over the car, for example, can be used to detect skidding and force the braking of one single wheel to aid in steering – a functionality impossible with strictly localized solutions. At the same time the interaction of many distributed components causes a considerable challenge for testing, since the proper function of one single component in isolation is required but not sufficient to guarantee the proper distributed functionality. An exhaustive functional test of the complete distributed system, on the other hand, is ruled out by its complexity. So an appropriate test strategy is urgently needed.

The idea to develop such a test strategy is the central focus of our research project STEACS ("Systematic Test of Embedded Automotive Communication Systems")[1]. With project partners spanning from academia to industry we had some very interesting discussions on how such a test concept should ideally look like. These discussions not only ascertained us that the choice of a project type in the area of conflict between fundamental research and applied research was a very fruitful choice for this topic; we also felt that it would be interesting to summarize the essence of these discussions in the following.

2. Considered Target System

We consider an automotive control application executed on a set of distributed nodes that are connected over a serial broadcast channel (bus, star or mix of both topologies). Due to their favourable properties in safety-critical automotive applications we consider time triggered communication protocols, and in particular the FlexRay protocol in our study. In these protocols a global time base is maintained by a distributed clock synchronization algorithm, and all sensitive activities on the communication channel are time-driven. While it is relatively easy to test the involved nodes in isolation, the challenge lies in the testing of the inter-operation between the nodes.

3. The "Dream"-Solution

In a first step we tried to define what would be a perfect solution from the testing point of view, without considering practical limitations such as overheads, test time, and cost. We felt that the term "systematic" could be best translated into "comprehensive" in this context and that the concept is understood as a generic long-term solution.

Clearly a perfect test concept provides *100% test coverage*. Therefore all protocol features must be tested in detail, ideally in all possible modes of operation. To this end we can choose one of the following strategies:
(1) Observe the cluster during normal operation without having the tester interfere in any way. While this provides very natural test conditions, it will probably not be possible to observe all conceivable modes of operation within a limited observation time. Even in our "dream" solution it appears useless to assume infinite observation time.
(2) Run a specific "stress" application or have the tester interfere with the system such that all relevant modes of operation are actually entered during the observation interval. Since we want to include fault tolerance and error handling features in our test as well, we have to generate error conditions, by means of *fault injection*, for instance.

Compliance with the specification must not only be tested on the functional level, but *parametric tests* need to be performed as well. For example, it must be possible to find out whether the local clock oscillator of a node is notoriously slow even if the value is still within the al-

[1] The STEACS-project (http://embsys.technikum-wien.at/steacs.html) received support from the Austrian "FIT-IT[embedded systems" initiative, funded by BMVIT and managed by the FFF under grant 807146.

lowed margins. At the same time features that are not protocol-related (like low or noisy power supply, e.g.) are not within the scope of our test. Detecting such non-target faults as a side effect is undesired, since we want our test to be *perfectly selective*, i.e. not to cause false alarms.

In addition to detecting faults we want to be able to perform *diagnosis*: If a service is detected to be incorrect or missing we want to know the exact cause. In order to achieve such a high-resolution diagnosis we have to perform extensive monitoring. We have developed a layer model [2] – with abstraction levels reaching from the signal representation on the communication bus to the semantic contents of the message – that allows us to identify the input and output stream of every service in the protocol. In context with this model extensive monitoring means tracing the complete data stream on every abstraction level, such that an (off-line) analysis of the recorded streams allows a clear identification of the initial fault. For this purpose we want to add a monitoring node to the communication bus that receives all bus traffic and transforms it into the respective layer-specific representations. This should occur without the operational system even noticing the existence of this tester node.

4. The Practical Borderlines

From a practical point of view a "systematic" test is a "(cost) effective" one. In contrast to the dream-solution – which concentrated on finding a most elegant generic test concept – the practical borderlines have a higher focus on the *problem*. In this context it is not so important for the test to have high coverage with respect to a theoretical fault model, but the most important question is rather: "Is the test concept capable of solving our current (and potential near-future) problems?" Should the test cover more than these burning problems, it is probably too expensive and could be reduced. On the other hand, it is very welcome, if the test aids in solving relevant non-target problems. The diagnostic resolution is not oriented towards the actual root of a problem; in practical operation it is generally sufficient to identify the node or module that needs to be replaced. This, however, should ideally occur in an automated fashion instead of requiring a sophisticated interpretation of symptoms by an expert. Test duration is, of course, limited, but the actual limits depend on the phase during which the test is performed: In the development phase and the system integration phase, the "stress" approach will be applied, and there will usually be very tight limits on test duration. During the mission an online monitoring tool that does not influence the ongoing operation may be allowed very long observation periods.

Another point that showed up very early was the limitation of resources: The time to develop and implement the concept is limited, and there are constraints with respect to cost, size and power consumption of the solution. Since for these reasons we cannot afford to develop an own specific hardware platform, we have to choose from a limited set of available platforms, which limits the admissible complexity [3]. For example, monitoring the complete flow of information on every abstraction level as suggested in the dream solution is ruled out by this constraint – we have to choose the information we trace very carefully. Moreover, in practice not all required information can be collected just by having a tester node record the traffic on the bus. As already mentioned it will be necessary to actively influence the data on the bus, e.g., by means of fault injection. In addition internal information from the observed nodes will be required in some cases. For this purpose it will be necessary to have some software agent residing on the node under consideration that transmits the required information over the communication bus from where it can finally be received and analysed by the tester node. Unfortunately, the execution of the software agent and the transmission of the collected data are likely to cause an undesired influence on the timing behaviour of the node and on the bus schedule, respectively. Therefore we will try to circumvent this solution whenever possible.

5. Merging Dreams and Borderlines

Considering these practical limitations our dream solution may seem very naive at the first glance. Still we believe it is a good strategy to make clear what can be attained in the ideal case beforehand and decide for every issue whether a reduction makes sense. From this point of view our dream solution represents a 100% performance full-featured tool. If we decide to reduce the diagnostic capability, e.g., we can balance the savings with the performance degradation.

From a formal point of view the task of concept elaboration involves optimizing the performance parameters towards the dream solution while considering the practical borderlines as boundary conditions. This optimization process is extremely complex, and many decisions – especially those related to cost and complexity – depend on implementation details that can be clarified only by a direct comparison of alternatives.

6. References

[1] G. Leen and D. Hefferman. In-Vehicle Networks, Expanding Automotive Electronic Systems. *IEEE Transactions on Computers*, pp. 88–93, January 2002.

[2] E. Armengaud, A. Steininger, M. Horauer, R. Pallierer. A Layer Model for the Systematic Test of Time-Triggered Automotive Communication Systems. 5th *IEEE International Workshop on Factory Communication Systems*, 2004 (submitted).

[3] E. Armengaud, A. Steininger, M. Horauer, R. Pallierer, and H. Friedl. A Monitoring Concept for an Automotive Distributed Network - The FlexRay Example. 7th *IEEE Workshop on Design & Diagnostics of Electronic Circuits & Systems*, 2004.