Design and Validation of Embedded Real-Time Applications

S. Siegl, C. Lauer

University of Erlangen-Nuremberg, Department of Computer Science 7, Erlangen, AUDI AG, Ingolstadt, Germany

Abstract: The design and validation of embedded real-time applications is challenging, especially when legacy sub-systems are involved. To account for the uncertainty in system-development at early design stages we use statistical modelling and discrete event simulation to perform sensitivity analysis. These analysis results provide vital information about the system characteristics and indicate usage scenarios where the behaviour of the system differs significantly from the average case. Based on the simulation results and the initial system requirements a usage model for the application is being set up. The model represents the requirements in an unambiguous and traceably correct manner. For each possible path through the model, considering stimuli and their timing, a unique system reaction is defined. This way the requirements are clarified. The usage model allows the derivation of test cases that can be used in the design phase to validate the model and in the acceptance phase to test the final system. Through the combination of the simulation results and the usage modelling we are able to:

- identify critical system conditions.
- validate the system design w.r.t. the usage model

The proposed methods are currently applied in both the design and validation of safety critical applications.

Keywords: Embedded Real-Time Systems, Design, Validation, Non-functional properties, Safety-Critical

1. Introduction

More and more complex embedded applications are developed to realize efficient and safe automotive systems. Important design decisions have to be made at early design stages, when performance parameters of sub-systems are not verified and requirements are not fully developed. However, it is important to identify critical paths through the system and usage scenarios that imply the fulfilment of hard real time requirements as early as possible. Doing so allows for an early validation of the design w.r.t. the application requirements. Hence, design flaws can be avoided and valuable time and money can be saved.

Simulative approaches to scheduling analysis are a feasible analysis technique at early design stages and are often applied if complex scheduling strategies and communication protocols are deployed. In contrast to formal schedulability analysis, parameters can be statistically influenced and investigations from an average-case perspective are possible. Commercial tools like the ChronSim simulator [1] or academic tools like MAST [2], and Cheddar [3] represent such schedule simulators. However, only commercial tools offer facilities to analyse automotive systems.

In this paper we focus on the analysis of statistically influenced simulation models. Although some of the above mentioned tools support statistical patterns for signal arrival and task execution time latency none of the available tool support proper simulation control mechanisms. In order to perform statistical simulation without output bias we implemented a simulation model and simulation control mechanisms.

Model-based testing makes use either of behavior models of the SUT (system under test) or of usage models that describe the expected usage of the SUT [4]. Due to the fact that in the early design phase no behavior models of the system are available we decided to derive a usage model from the initial requirements and simulation results. This model is independent from the design models and represents the requirements in a unique and correct form. It is used to analyze and clarify the requirements, and to validate the design and simulation in terms of conformance with the requirements and feasibility.

In the later system testing phase the usage model is used to automatically generate test cases from it [5]. Because exhaustive testing of real systems is hardly feasible in practice a set of test cases is derived to meet a given test goal. By applying a usage model already in the design phase, the validation and detection of design flaws happens in an early phase. The knowledge gained in the design phase can be used to generate sets of test cases that expose the system to usage scenarios that are known to give important information for the validation of the final product.

2. Embedded Real-Time Application

ECUs in the automotive domain host multiple applications that may be responsible for safety, chassis, body, or diagnosis. In the safety domain, integrated ECU architectures that perform both active and passive safety applications are getting increasingly popular. Figure 1 depicts a set of example applications that may be hosted by such an integrated architecture.

In this example a remote sensing system (e.g. radar, laser scanner, computer-vision) perceives the

current driving context and transmits object lists to a collision detection algorithm. In this step a metric is evaluated which captures the certainty of an imminent crash.

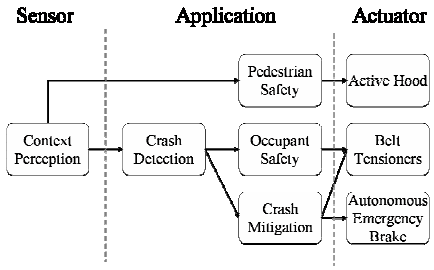

Figure 1: Example applications from the automotive safety domain

Pro-active pedestrian protection mechanisms also take into account the sensor data object lists as the crash detection does not consider vulnerable road users. The occupant safety and crash mitigation application only react on opposing vehicles and may activate reversible belt tensioners and brakes. The actuator for the active pedestrian protection is an active hood that offers an additional deformable zone in case of an accident with pedestrians.
Integrated ECUs can contain multiple micro controllers, ASICs, inertial sensors, and field-busses. The architecture in figure 2 contains an ECU which hosts two controllers connected by an SPI field-bus. Controller 1 is connected to the remote sensing system via a dedicated connection and controller 2 is connected to the intra-car time-triggered bus system. The activation of the actuators is done by controller 2 either by dedicated connections (active hood and belt-tensioners) or via the time-triggered bus (brake). The crash detection is allocated to controller 1. The three safety applications are hosted by controller 2.

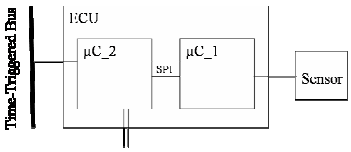

Figure 2: Example system architecture

In the following we will show how the concepts of discrete event simulation and model-driven testing based on usage models can cooperate in order to design such systems.

3. Discrete Event Simulation

In the following we describe the model that simulates the propagation of the application data from the sensor to the respective actuators.

3.1 Simulation Model

The time required for the acquisition of context information is dependent from the number of objects in the scene, their arrangements, and object clutter. However, the sensors are designed to supply a new vector of objects at a certain frequency. Hence, the sensor is modeled to generate data tokens with a constant rate. The transmission of the sensor data is subject to the object list size and takes between 300 and 400 µs. It is evaluated using a uniform distribution. The processing of the object lists in the crash detection task on controller 2 is also dependent from the number of objects and is also subject to preemption. The total response time of the software tasks that run on the micro controllers is simulated using the respective execution times, priorities, activation offsets, and the task scheduling strategy of the operating system. Table I summarizes the task properties.

Table I: Properties of the real-time tasks in the simulation model

TaskID	µC	T_{exec}[ms]	T_{period}[ms]	Prio	T_{offset}[ms]
CollDetect	1	uni(1,4.3)	10	3	0.7
Legacy1000	1	uni(0.18,.2)	1	2	0.4
Legacy250	1	uni(0.08,0.1)	0.25	1	0
CrashMiti	2	uni(1.6,3.9)	10	3	2.5
OccuSafe	2	uni(0.48,0.7)	5	2	1
PedeSafe	2	uni(0.6,1.7)	5	1	0

The intra-ECU SPI communication takes a constant amount of 50 µs whereas the communication of the application decisions via the time-triggered bus system can only take place at a predefined time. Time simulation of the end-to-end delay from sensor data acquisition to result transmission is simulated by tokens that traverse the simulation model from source to sink (e.g. sensor to bus-interface). The tokens are delayed at each model element according to statistical parameters and/or scheduling behavior of the controllers.
The performance of the safety applications is dependent on the end-to-end delay between context perception and actuator activation. Directly connected actuators can be activated with little additional delays. However, time-triggered bus systems can cause addition delay when the transmission time just passed before the results of the safety applications became available. Note, that the global communication schedules in time-triggered architectures are defined before the local node scheduling takes place (see chapter *System Design* in [6]). The deadline in Table II represents the instance when the data has to be available (w.r.t. the sensor data acquisition) in order to ensure a timely transmission via the bus system.

3.2 Output Analysis

As described before we use statistical modeling to account for uncertainty in the system.

Table II: Timing properties of the applications w.r.t. the sensor data acquisition

Application	T_{deadl}[ms]	T_{min}[ms]	T_{max}[ms]	T_{median}[ms]
Crash Mitigation	28.5	13.76	18.23	16.65
Occupant Safety	12.25	6.16	11.95	11.23
Pedestrian Safety	11.5	5.27	6.31	5.7

To remove statistical bias from the results of the output analysis we perform simulation control on multiple experiment replications until a relative error below 10% is achieved. The results of the experiment are presented as box plots in figure 3.
Every application signal run contributes one data point to the box plot. The median of the signal latencies is well under the signal deadline for both the crash mitigation application and the pedestrian protection application. Furthermore, the simulated max-values of the signal delay are lower than the deadlines given in Table II. Although the latencies of the occupant safety application are also under the deadline stated above, the results suggest that about 50% of the signal delays are under 1 ms from their respective deadlines. At early design stages this might be an unwanted risk. Two possible solutions are feasible:

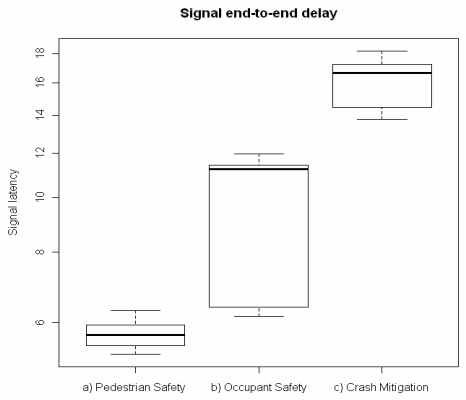

Figure 3: Box plots of the simulation results

1. The task scheduling on controller 2 can be adjusted to allow for a higher priority in the execution of task *CrashMiti*. This, however, would also increase the latencies of the other tasks, hence, resulting in higher end-to-end latencies of the respective applications.
2. The delay caused by the crash detection algorithm is high compared to the rest of the signal. Deadline violations might occur at later design stages when this processing step takes lots of time, i.e. the algorithm has to evaluate a large number of objects. Focusing on this very scenario in ECU testing reduces the need for higher safety margins at early design stages.

4. Worst-Case Response Time Analysis

So far, we focussed on the average-case perspective of the system timing. The average-case is important if the system perfromance is influenced by the average signal delay and the maximum signal jitter. However, during the development of safety or life-critical systems the worst-case timing behavior must not be neglected. Using discrete event simulation to cover the worst-case timing analysis can only be applied to a certain extent. In section 3 we have modeled the responsetime of the tasks using a uniform distribution of execution times. For the worst-case analysis we now have to use the worst-case execution times of each task and derive the exact response-time without any statistical influence. This requires the knowledge of the full task set and all respective task properties.

A. Simulating the Worst-Case

In a purely time-triggered system without clock drift the worst-case delay can be simulated by either setting up the worst-case scenario and simulating one single message transmission, or by simulating over the hyper-period of time and tracking the worst-case delay. In complex systems, however, the worst-case scenario is not necessarily known a-priori and simulating the hyper-period of the system is more feasible. The hyper-period P denotes the least common multiple of all cyclic time-domains D_i in the system (equation 1).

$$\mathrm{P} = lcm(D_0,...,D_{i-1}) \qquad (1)$$

In the example system consisting of two controllers and one time-triggered communication system, this hyper-period is the lcm of the bus period, and the two schedule periods. Systems consisting of multiple clock references inherently incorporate clock drifts and drifts between the periods $t_0,...,t_{i-1}$, hence, P has to be calculated taking into account the clock drifts $\tau_0,...,\tau_{i-1}$:

$$\mathrm{P} = lcm(D_0 + \tau_0,...,D_{i-1} + \tau_{i-1}) \qquad (2)$$

Unfortunately, $\tau_i << t_i$ therefore P increases rapidly even for small numbers of different clock references. The simulation of the hyper-period may become computationally infeasible, therefore, formal methods are often used to analyze the worst-case timing of systems.

B. Asynchronous Task Scheduling Analysis

A software task that does not meet its deadline, e.g. the transmission time of a time-triggered bus system, may decrease the system performance and even result in an unsafe system state. However, using response-time analysis to validate correct timing behavior is difficult if complex scheduling strategies are in use. Consider the following asynchronous task set. A task set is called asynchronous if the first activation of an arbitrary task may be delayed by an offset value:

Table III: Legacy airbag system task set

Task	Prio	WCET [µs]	Period [µs]	Offset [µs]
Signal Processing	0	8	125	23
Internal Sensors	1	37	250	50
External Sensors	2	440	2500	0
Belt Executive	3	5	1000	0
OS Services	5	8	250	100
Communication	6	200	5000	0
Controller_Com	7	130	5000	1100
PedeSafe	8	330	5000	0
OccuSafe	9	200	5000	1000
CrashMiti	10	1950	10000	6000
Diagnosis	11	430	10000	4000

Table III summarizes the detailled view on controller 2 in section 3. Since statistical modeling is not feasible for worst-case analyses, we need the exact task properties of all tasks in the system. Whereas in section 3 we used a uniform distribution of execution times to account for increased latency by task pre-emption we are now able to calcuate upper bounds of the worst-case response times for the tasks *CrashMiti*, *PedeSafe*, and *OccuSafe*. To calculate these bounds we use the response-time analysis proposed in [7] using equation 3.

$$w_{n+1} = Ci + \sum_{t \in trans} I(i,t,W) \qquad (3)$$

With *I(i, t, W)* being the interference of transaction t with activation W on task *i*:

$$I(i,t,W) = \sum_{k \in (tr(t) \cap hp(i))} \left(\left\lceil \frac{w_n + W_t - \hat{O}_k}{T_j} \right\rceil_0 - \left\lfloor W_t - \hat{O}_k \right\rfloor_0^1 \right) C_k \qquad (4)$$

Equation 3 calculates the size of a busy window w_n, i.e. the time the controller evaluates tasks without any idle moments, using a fix-point iteration. The length of the busy window depends on the offset *O* of each task w.r.t. the start of the busy window and an assumption about the tasks that have been activated at the instant the busy window starts (defined by a vector *W* of activation times for each ransaction; A transaction of a given activation period contains all tasks with the respective activation period.). Hence, for a worst-case analysis, every possible instant when tasks may be activated have to be evaluated in order to find the critical instalt that leads to the actual worst-case response time. The length of the busy window and the activation information of the analysis task are evaluated to the response time of the task using equation 3. For an in-depth introduction to the schedulability analysis please refer to [7].

$$r_i(W_i) = w - (\hat{O}_k - W_i) \qquad (5)$$

In our case the resulting response-time boundaries calculate to the ones summarized in table IV: Note, that the times include a certain amount of pessimism particularly if some of the tasks are not triggered by external events but from the same clock reference. Moreover, the computational complexity of the presented technique is infeasible for larger task systems with many different activation periods. Various approximations to exact solutions have been proposed in the literature which can be found in [8,9,10]. The response-time analysis of the task set presented above is calculated with the values is table IV.

Table IV: Results of the WCRT Analysis

Task	WCRT [µs]
Signal Processing	8.0
Internal Sensors	45.0
External Sensors	591.0
Belt Executive	596.0
OS Services	567.0
Communication	889.0
Controller_Com	811.0
PedeSafe	1737.0
OccuSafe	1077.0
CrashMiti	4935.0
Diagnosis	7229.0

As can be seen from the results of the analysis, the upper bound of the response-time of task *OccuSafe* is significantly larger than the assumptions made in section 3. Validating a response-time of under 700 µs requires a priority between 7 and 8. Changing the priorities this way only affects the tasks *OccuSafe* and *PedeSafe*, see table V:

Table V: Refined WCRT properties with different priorities

Task	WCRT [µs]
PedeSafe	1998.0
OccuSafe	298.0

Now, the latency of the Occupant Safety application caused by controller 2 can be bounded to 298 µs while the latency of the Pedestrian Safety application will be increased. In order to validate the rest of the timing chain starting from the sensor device and ending at the respective actuator device we will now focus on using the results from the simulation study for test case generation and ECU timing validation.

5. Testing with Timed Usage Models

In the following we focus on how we can use the results from the simulation study for test case generation and ECU timing validation.

5.1 Timed Usage Models

Nowadays usage models are employed to describe the possible usage of the SUT and to derive test cases. Markov chain usage models (MCUM) are an established way for doing so. The usage model is basically a collection of states and transitions and it represents the possible usage of the system [5].
It can be understood as a formal definition of all possible actions, defined by stimulations, called the "Inputs" and owned by the transitions. Because it serves as a test model, transitions own also reactions, i.e. outputs of the SUT that can be defined as "Expected Results".
The problem was that classic MCUMs do not provide a way to integrate time and timing systematically in the usage model. We developed Timed Usage Models (TUM) to overcome these drawbacks [11]. TUMs allow the integration of non-exponential timing, either on states or on transitions, in order to be able to describe the usage in a realistic way. The usage model serves as test model and hence it must be able to consider and describe real time aspects in an appropriate manner. Usage states and transitions can be assigned a probability density function (pdf) over time. These pdfs are used to sample the sojourn time in states and the execution duration of stimuli. The variability in timing of usage and different users can be described with usage profiles that store the probabilities of stimuli and their timing.

A formal definition of a Timed Usage Model shall be given: A Timed Usage Model (TUM) consists of:

- A set of states $S = \{s_1, \ldots, s_n\}$, that represent possible states of usage.
- A set of arcs A, representing state transitions. An arc from state s_i to state s_j is denoted by a_{ij}, multiple arcs between s_i and s_j are not allowed.
- A set of stimuli Y on the SUT. A stimulus y_j is assigned to each arc.
- The transition probability from state i to state j, denoted by p_{ij} for an existing arc a_{ij}. Otherwise the transition probability $p_{ij} = 0$. The transition probabilities obey the conditions $0 \le p_{ij} \le 1$ and

$$\sum_{j=1}^{n} p_{ij} = 1 \qquad \forall i = 1, \ldots, n \quad (1)$$

  states that the probabilities of all outgoing arcs from a certain state s_i must sum up to one.
- A probability density function (pdf) t_i to reflect the sojourn time is assigned to each state s_i.
- A pdf of the stimulus time t_{ij} is assigned to each arc a_{ij}. This pdf describes the duration of the execution of a stimulus on the SUT. The concept provides the possibility to characterize a stimulus by its typical variation in time, that can be fix or variable and vary from very small to large values.

The elements of a Timed Usage Model are presented in Figure 4 as a graph example.

Two states have special characteristics, that are:

- State s_1 is the sole initial state (also: start state).
- State s_n is the final state (also: end state).

The transition probabilities p_{ij} from state s_i to state s_j as well as the values of the timing attributes t_i and t_{ij} can be stored and exchanged by means of a matrix P. This way different user types can be distinguished w.r.t. the appliance of stimuli and the timing of and between stimuli.
All paths from the start to the final state are valid test cases. The statistical sampling of test cases can be guided by different usage profiles that represent different users or usage conditions of a system.

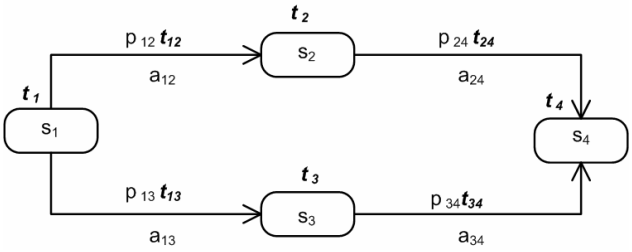

Figure 4: Attributes of Timed Usage Model

5.2 Extended Automation Method (EXAM)

The test method used by the AUDI AG and within the Volkswagen AG to perform tests at component and system level is called EXAM, which is the abbreviation for Extended Automation Method [12]. The test automation EXAM is available as Freeware and can be downloaded from www.exam-ta.de .

Moreover, EXAM as it is used within the Volkswagen AG defines a process, the roles, and the tools used to:

- model test cases graphically and platform independently in UML. Sequence diagrams are used for this task and build the formal basis for test case specifications.
- generate platform dependent test scripts automatically from the formal description in UML. In this way a separation between the test case description and its concrete implementation is achieved.
- to use sharable test automation functionalities from a structured database. Thus, test cases can be developed in independent test teams and test know-how is accumulated enterprise-wide.

EXAM provides the means for the formal specification of platform independent test cases. Yet in EXAM itself each test case must be invented and created manually.
We introduced usage modeling and test case generation from usage models in order to generate test cases into EXAM.

5.3 Process

In this section the process for integration and system testing is described. The process described in this section does not expect the creation of the model in a previous development phase. So if no model is at hand, it can be created in this development phase.
The process is as follows: The process initializes with a test request and the specification documents, e.g. in natural language. In a next step the test designer creates a usage model that formalizes the requirements. Based on this model that represents the requirements in a platform independent manner test cases in the UML are generated. In Fig. 4 this is the pictogram with "'Step 1, Step 2"'. EXAM generates automatically platform specific python code from the test case specifications. The test cases are executed on HIL simulators and results saved in test reports.

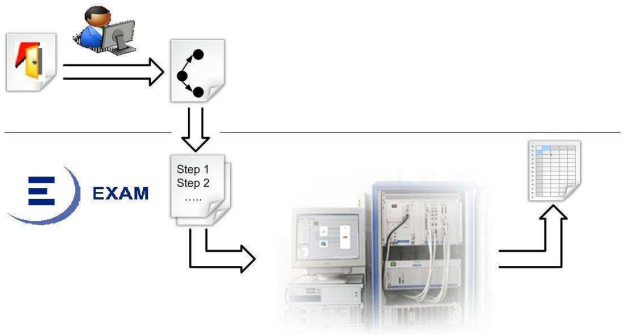

Figure 5: Process with EXAM for HIL testing

5.4 Toolsuite

We chose All4tec MaTeLo (www.all4tec.net) for usage modelling and test case generation. The MaTeLo tool suite was enhanced so that the EXAM library can be accessed from MaTeLo. Functionality for test automation from EXAM can be associated with usage models. This makes it possible to generate platform independent test cases out of MaTeLo. In a next step these test cases can be executed on a test-bench. We executed test cases on HIL simulators.
MaTeLo owns a "Requirements library" allowing associations of requirements to the model objects. The aim is to link the test model with the requirements and to insure the requirements validation during the test campaign. During the creation of the model it is possible to associate at each step from which requirement it is derived. This way the analysis of requirements is supported by the tool.

5.5 Validation of the Requirements and Simulation Results

The creation of the TUM is a very important task, as the model is used as a basis to assess the chosen design and to derive test cases. Already during the creation of the model the requirements are analyzed and brought into a unique and traceably correct representation. To form a valid representation of the requirements, the TUM is constructed using principles of sequence-based software specification. It consists of the following steps:

1) Identify the system boundary
2) Enumerate all sequences of stimuli and their responses across the system boundary

By this procedure, a complete and consistent usage model is created. An additional feature is the easy traceability of requirements. Beside the annotation of the response to each stimuli sequence the corresponding requirement is assigned. This ensures the correctness of the model and the detection of incomplete and inconsistent requirements.
If no requirement or desired system response is found for a sequence this has to be documented and a requirement has to be derived. So this procedure describes a technique to analyze the requirements. The results of the simulation, the discovered anomalous and critical paths, are compared with the desired reaction implied by the requirements and the usage model. Furthermore, the TUM can be used to review the simulation. The TUM provides the basis for a systematic identification of scenarios that are critical from usage point of view. It can be a result of the simulation that from a system point of view these scenarios are not critical.

6. Case Study

6.1 Timed Usage Model of Safety Application

In Figure 4 the top level of the usage model is presented. In the beginning initial conditions to set the applications active are set. Next all inputs that affect the application under development are structured by transitions. Each transition represents one interface of the application. Equivalence classes of input data are defined. Each interface acts at a predefined abstraction level, e.g. one that stimulates the object lists. This way it is possible to systematically define the desired system reaction for each combination of classes. This is done in the macrostate *Check Subsystems*. In this state the desired reaction of all subapplications, such as collision detection and reversible belt pretensioners are specified. From this state it is possible to finish the scenario or to modify the input parameters. As maximum delays for the system reaction the values from the simulation were taken.

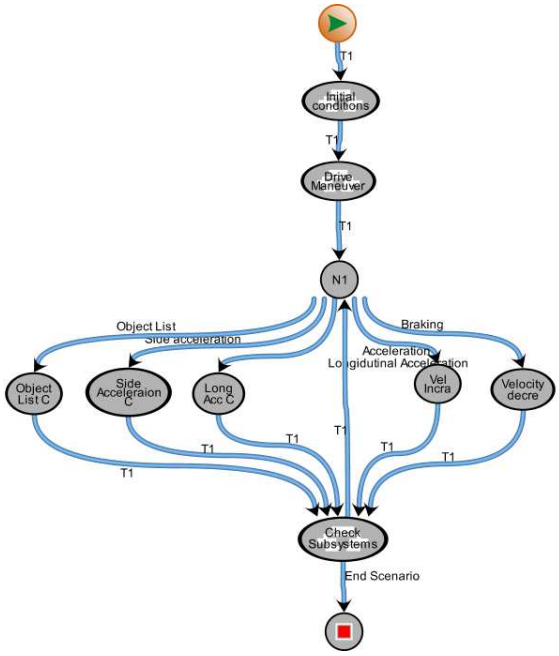

Figure 6: Top level of Usage Model

Usage scenarios that were identified critical by the simulation are generated from the usage model. The specified system reaction from the usage model is compared with the results of the simulation. In the case of a complex application the validation of the simulation is not a trivial task. Compliance test cases can be generated from the usage model to assess the design of the system.

6.2 Test Case Generation

HIL simulators are an established test bench in industry and allow the testing of embedded applications under real conditions. The main focus is functional testing to detect design flaws or erroneous implementations. However, the high acquisition and working costs of HIL systems require many departments to share a single HIL for validation and testing of different applications. Therefore, testing time is scarce and should be used as efficiently as possible.

Using MaTeLo Testor it is possible to control the test generation using various parameters and selecting the appropriate test strategy. Currently two families of algorithms are available and can be configured for test case generation:

- Usage oriented: These algorithms use the probabilities of the chosen test profile. The generation is controlled via the model architecture and the probabilities given by the profile. Test cases that are sampled from test profiles that represent usage can be used to estimate the reliability of the SUT. This way a stopping criterion for testing can be defined. This algorithmic family provides

also the possibility to generate test cases for boundary testing of equivalence classes.

- Coverage: This algorithm can be used to generate a set of test cases in which each transition of the model is covered at least once. This algorithm allows the validation of the SUT against recurrent failures. Results of test cases from the coverage algorithm can be used to have a basic indicator about the proper functioning of the SUT.

Currently All4tec Testor is enhanced to provide an API for test case generation strategies. This API makes it possible within Volkswagen AG to develop and use proprietary test case generation strategies that can e.g. generate an optimized set of test cases to meet a certain test purpose. Test generation strategies are currently being developed by the author and base on the concepts presented in [11]. Indicators and test management information are integrated in the model that are used by the test generation strategies.

6.3 Results of case study

With our approach design flaws are detected in early phases of the development. Knowledge about usage scenarios that are of high significance to assess the correct behavior of the application is accumulated during the design phase. These are e.g. scenarios with a large number of objects in the vehicle's environment. This knowledge can be used to generate a set of test cases for efficient HIL-testing. This way information from the design phase is passed to the system- and acceptance testing phase and helps to improve the final validation steps.

7. Conclusions

In order to investigate the performance metrics of an embedded real-time application at an early design stage, we conduct discrete event simulation with statistical models. The simulation model comprises of automotive specific communication and scheduling protocols and statistical bias is avoided by proper simulation control. The results of the output analysis are then used in the usage model-based test method. Combining the two approaches enables us to:

- identify critical usage scenarios that may represent an unwanted risk in the product development.
- validate the correct system behavior w.r.t. the usage model, hence, reducing the risk of a costly redesign at later design stages.

Currently Timed Usage Models are created for safety applications, energy management, and air conditioning of vehicles. Systematically derived test cases give valuable information about scenarios that should be covered by test cases. Knowledge from the design phase supports this process.
Applying model-based analysis and validation techniques at the system design phase supports a more systematic engineering process and helps to build products of higher quality.

6. References

[1] T. Kramer and R. Münzenberger: "New Functions, New Sensors, New Architectures – How to Cope with the Real-Time Requirements", Advanced Microsystems for Automotive Applications Smart Systems for Safety, Sustainability, and Comfort, page 435, 2009

[2] M.G. Harbour, J.J. Gutierrez, J.C. Palencia, and J.M. Drake: "*Mast: Modeling and Analysis Suite for Real-Time Applications*", Proceedings of the Euromicro Conference on Real-Time Systems, Delft, The Netherlands, 2001.

[3] F. Singhoff, J. Legrand, L. Nana, and L. Marcé.: "*Cheddar: a flexible real time scheduling framework*", ACM SIGAda Ada Letters, 24(4):1–8, 2004.

[4] S. Rosaria and H. Robinson: "*Applying models in your test-ing process*", Information and Software Technology, 42:815–824, 2000.

[5] S. Siegl, V. Entin, R. German, G. Kiffe: "*Model Driven Testing with Time Augmented Markov Chain Usage Models - Computations and Test Case Generation Algorithms for Time Augmented Markov Chain Usage Models*", ICSOFT (1) 2009: 202-207

[6] H. Kopetz: "*Real-time Systems: Design Principles for Distributed Embedded Applications",* Springer, 1997.

[7] K.-W. Tindell. Using *offset information to analyse static priority preemptively scheduled task sets*. University of York, Department of Computer Science, 1992

[8] J. Mäki-Turja and M. Nolin. Efficient implementation of tight responsetimes for tasks with offsets. *Real-Time Systems*, 40(1):77–116, 2008.

[9] J.C. Palencia and MG Harbour. Offset-based response time analysis of distributed systems scheduled under EDF. In *Proceedings of the 15th Euromicro Conference on Real-Time Systems*, pages 3–12, 2003.

[10] R. Pellizzoni and G. Lipari. A New Sufficient Feasibility Test for Asynchronous Real-Time Periodic Task Sets. In *Proceedings of the 16th Euromicro Conference on Real-Time Systems*, pages 204–211, 2004.

[11] S. Siegl and R. German, "Model Driven Testing with Timed Usage Models in the Automotive Domain," in ISSRE 2009. IEEE Computer Society

[12] G. Kiffe, EXtended Automation Method (EXAM) Konzeptpapier Version 3.0, Audi AG, Ingolstadt, Mai 2009. [Online]. Available: http://www.exam-ta.de .