EQUITAS: A tool-chain for functional safety and reliability improvement in automotive systems

Reda Nouacer, Manel Djemal, Smail Niar, Gilles Mouchard, Nicolas Rapin, Jean-Pierre Gallois, Philippe Fiani, François Chastrette, Arnault Lapitre, Toni Adriano, et al.

Microprocessors and Microsystems 47 (2016) 252–261

EQUITAS: A tool-chain for functional safety and reliability improvement in automotive systems

Réda Nouacer [a,*], Manel Djemal [b], Smail Niar [b], Gilles Mouchard [a], Nicolas Rapin [a], Jean-Pierre Gallois [a], Philippe Fiani [c], François Chastrette [d], Arnault Lapitre [a], Toni Adriano [d], Bryan Mac-Eachen [e]

[a] CEA, LIST, Software Reliability and Security Laboratory, P.C. 174, Gif-sur-Yvette, 91191, France
[b] LAMIH, University of Valenciennes and Hainaut-Cambrésis, 59300 Valenciennes, France
[c] Sherpa Engineering, 12 avenue de Verdun, F-92250 La Garenne-Colombes, France
[d] ALL4TEC - Immeuble Odyssée - Bât E - 2-12 rue du Chemin des Femmes, 91300 MASSY, France
[e] Continental Automotive France SAS, Division I B&S, PG3 CFT, 1 Avenue Paul Ourliac, 31036 Toulouse, France

Keywords:

Testing
Verification
Embedded systems
Automotive
Virtual plateform
Fault injection

a b s t r a c t

To support advanced features such as hybrid engine control, intelligent energy management, and advanced driver assistance systems, automotive embedded systems must use advanced technologies. As a result, systems are becoming distributed and include dozens of Electronic Control Units (ECU). On the one hand, this tendency raises the issue of robustness and reliability, due to the increase in the error ratio with the integration level and the clock frequency. On the other hand, due to a lack of automation, software Validation and Verification (V&V) tends to swallow up 40% to 50% of the total development cost. The "Enhanced Quality Using Intensive Test Analysis on Simulators" (EQUITAS[1]) project aims (1) to improve reliability and functional safety and (2) to limit the impact of software V&V on embedded systems costs and time-to-market. These two achievements are obtained by (1) developing a continuous tool-chain to automate the V&V process, (2) improving the relevance of the test campaigns by detecting redundant tests using equivalence classes, (3) providing assistance for hardware failure effect analysis (FMEA) and finally (4) assessing the tool-chain under the ISO 26262 requirements.

1. Introduction

In the past, safety and security were mainly critical in a few industrial fields, such as military, nuclear, health, avionics domains. However, as embedded systems are now present in a large number of devices, there is an increasing demand for safety and security. Recently, ISO 26262 [16] introduced stricter safety requirements in the automotive field.

It is largely accepted that the architecture of embedded systems is becoming more and more complex, both at the hardware and at the software level. Thanks to steady progress in the field of microelectronics (Fig. 1), embedded system engineers are now able to integrate more system functions on powerful System-on-Chips (SoCs). The automotive industry also benefits from these advances in microelectronics and engineers are now able to integrate advanced vehicle functions on high performance ECUs. Due to the increase in the error rate with the degree of integration, the clock frequency and the functioning conditions (temperature, magnetic fields, etc.), the issues of robustness and reliability become crucial in the design phase.

In conventional design tools, the hardware (HW) and software (SW) of automotive embedded systems is developed in parallel and the integration of the two parts is performed very late in the design process. In this phase, many errors can be detected, such as misunderstanding of the specifications (API, data formats…), missing real-time constraints, and bad resource sharing such as bad sizing, bad scheduling, de-synchronization, etc. Thus, the final integration of HW and SW, which consists in incorporating the application, the operating system and the device drivers, is a tremendous task requiring a time-consuming and complex debugging process.

[1] This work was financially supported by Bpifrance AAP FUI16 project EQUITAS and the General Counsel of Essonne (Conseil Général de l'Essonne-France). It is supported by competitiveness clusters System@tic, iTrans and ID4CAR

DOI : 10.1016/j.micpro.2016.07.020

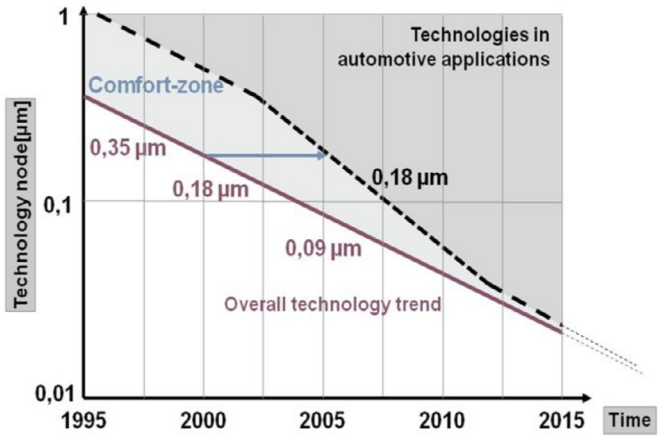

Fig. 1. CMOS technology trend for automotive applications [1].

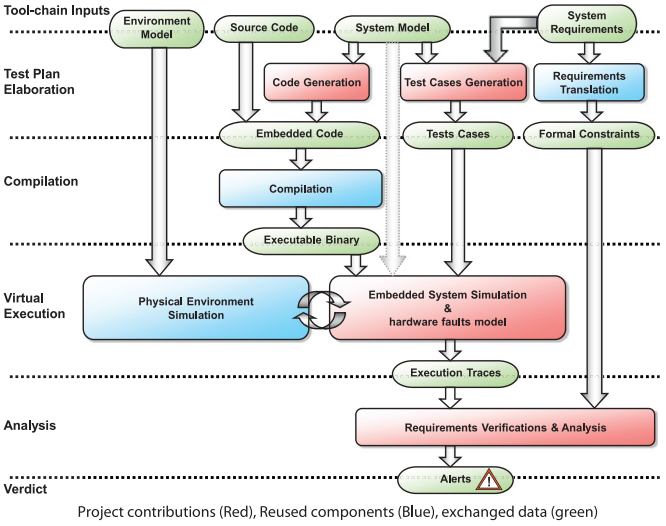

Fig. 2. The EQUITAS work flow.

Indeed, fixing system anomalies at this point of the design induces high costs and delays.

Automotive embedded SW results from the assembly of heterogeneous software components that are not necessarily available simultaneously. These components are designed in different design environments such as Simulink, XML, manual coding, StateMate using various tools from different providers. This makes the validation of the system complicated. In the last few years, code generators have been more and more used in embedded system design. But still many components of the software stack are hand coded based on specifications. This is mostly the case for low-level software such as operating system routines, low-level drivers and optimized code. Moreover, the testing process is prone to human errors. Application developers not only write the test sets, but also execute them, analyze the results and write the test report.

The quality and efficiency of the development tools directly impact software development productivity. For critical applications such as automotive systems, this software development productivity remains relatively low. Only 0.5 to 5 LoC (Line-of-Code) are produced per hour. The main reasons for this low productivity are the complexity and the limited automation of the V&V tasks, making the design cost higher and higher. Due to the complex HW/SW integration and the new certification rules [2], V&V takes about 40% to 50% of the total development effort.

For many years, it has been accepted that most software bugs are discovered in the final phase of the design cycle due to bad choices in the first phases of the project. Many defects discovered during the verification and validation phases, can be attributed to inadequate specification or to design choices which do not conform to the specification. Such defects could have been detected earlier in the design phase, if appropriate models and tools were used. It is essential to reinforce the usual development process with a solid systemic approach enabling early validation of soundness, adequacy and consistency of the specified elements at the system level. This approach cannot be considered unless the verification/validation tools have been used upstream.

The remainder of the paper is organized as follows. Sections 2 and 3 describe the project objectives and the technological bricks used to build the EQUITAS toolchain and the related work in the respective fields. Sections 4 and 5 describe the technical challenges and the EQUITAS tool chain. Sections 6, 7 and 8 focus on the technical achievements of the project. Finally, Sections 9 and 10 describe the case study and conclusion.

2. Project objectives and expected results

A practical solution to the problems mentioned above requires automating not only the simulation runs, but also the test case generation. This would enable better test coverage, simulation results analysis and diagnosis, and exchanges between all these processes. Moreover, a fully representative virtual validation environment is needed. Virtual platforms, including both ECUs and physical environment models, are required. These platforms must include mechanisms for easy input injection, namely nominal inputs, and faults, as well as extended means for systematic observation. Indeed, not being dependent on physical hardware simplifies the deployment of the verification/validation environment. This independence produces fewer logistical problems and eliminates physical/electrical constraints.

The EQUITAS (stands for ***E**nhanced **Q**uality **U**sing **I**ntensive **T**est **A**nalysis on **S**imulators)* project aims to limit the impact of software V&V on the cost and time-to-market of embedded systems while improving reliability and functional safety.

The EQUITAS project includes the following activities:

- Development of a continuous tool-chain (cf. Fig. 2) to automate the verification and validation process of whole embedded software stacks in the context of automotive electronic systems: test generators, simulation scheduler, automotive simulators, trace analyzer, analysis of compliance with the requirements.
- Improvement of the relevance of the test campaigns by detecting the redundant tests, using equivalence classes.
- Providing assistance for the hardware failure effect analysis (FMEA) by introducing a hardware fault model, during simulation.
- Assessment of the tool chain using real automotive use cases to extract a comprehensive validation methodology using virtual platforms.
- Assessment of the tool-chain under the ISO 26262 requirements.

3. State of the art

3.1. Methodology

One of the major strengths of the EQUITAS tool chain is that it uses the same tests and validation technologies throughout the

different phases of design flow. Another advantage is that EQUITAS enables re-use of validation artifacts, generated for the model level, to validate the next phases until the final embedded system implementation. This represents a real break with tools such as those offered by Mathworks around Simulink that treat only the monitoring and control software, without taking into account the constraints of the target execution platform.

3.2. Automatic test generation

Regarding critical software V&V techniques, there are two main approaches:

- The first approach, based on evidence, is limited in the case of very complex systems (explosion of proof algorithms) or has an inappropriate formal expression as it uses, for instance, low level layers;
- The second approach, based on simulation, aims to limit the number of test cases to be generated and to cover all possible cases.

Currently, there are attempts to solve these problems by trying to create tool chains based on software bricks such as Matlab/Simulink and/or StateMate and/or SCADE + DesignVerifier and/or Prover and/or MaTeLo and/or Teststand, etc. However, currently there is no integrated solution that addresses all the issues raised by EQUITAS, which offers a solution based on the coupling of the DIVERSITY [9] and MaTeLo [18] tools.

DIVERSITY is a model analysis tool based primarily on symbolic execution and originally intended for calculating symbolic execution paths (which are equivalence classes of test cases) of the analyzed models [10], in particular to detect inconsistencies in the models but also to generate test cases [11]. It is used in the project to detect redundancies in sets of numerical tests produced by MaTeLo.

3.3. Analysis and verification of compliance with the requirements

Simulation, either numerical or symbolic, is the most commonly used way to get a quick feedback on a model or a code, to ensure that it produces what is expected. However, when systems and their associated requirements grow in complexity, it becomes difficult to determine whether a simulation or execution satisfied a requirement. This is especially true when the requirements take the form of patterns with timing properties. Conventional investigative tools, debuggers and simulators, are insufficient for the analysis of complex properties including timed ones. Thus, simulation cannot be a panacea unless it is completed by automatic ways to analyze such properties. ARTiMon is a technology that meets this need. It offers a textual language for the expression of functional requirements with many operators involving time. It then can compile these requirements to synthesize automatic observers, who have the ability to automatically analyze executions/simulations, including on-line, i.e. while they are being executed.

Academically, ARTiMon belongs to the field of synthesis of observers from temporal logic expressions. The closest work is that of Nickovic and Maler from Verimag. In their publications [12,13], it appears that in some aspects the logic treated (offline) by their tool, AMT, is less expressive than that proposed by ARTiMon. Furthermore, ARTiMon guarantees that the generated observers maintain a bounded memory, which enables the analysis of arbitrarily long simulations. The technology of [12,13] is ambiguous on this point, the publications mention only memory saving principles, without ensuring that memory growth is limited.

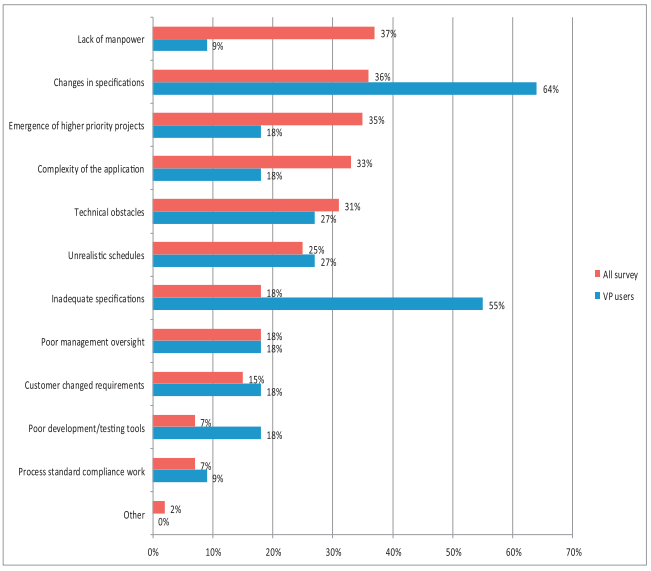

Fig. 3. Reasons for delays in project schedule.

3.4. Virtual platform

As illustrated in Fig. 3, the use of virtual platforms (VP) in both hardware and software development [6–8] helps to master the complexity of applications and meet production deadlines. The use of VPs allows parallelization of HW-SW development, anticipation of integration and, particularly, the detection and continuous identification of inconsistencies and specification errors.

Semiconductor manufacturers, such as Freescale, Intel, Texas Instruments and ARM, provide tools, such as CodeWarrior, WindRiver, DS5, etc., to exploit their system-on-chips or boards. Usually, a chain of tools includes development tools, for simulation and debugging that enables the development targeting their hardware components. However, these proprietary tools offer little interfacing capabilities with external tools. For example, most of the time, the instrumentation of embedded software requires a modification of the application in order to inject test cases or hardware faults and to observe the behavior and to verify requirements. The manufacturers' simulators often include too few, if any, simulated devices. The extension of these simulators, such as adding specific devices, is almost impossible and this impacts the representativeness of the simulation. In fact, there is no alternative to the development on the physical board. As a result, embedded systems integrators have difficulty integrating these multiple tools into their own design flow.

The third-party solutions, such as those proposed by Synopsys VDK and WindRiver have more interfaces. However, the costs of licensing and maintenance are high. Moreover, their parts catalogs are limited and the cost of on- request development is high.

UNISIM-VP [5] is an open-source (BSD licensing) simulation environment that is positioned in the field of hardware/software co-design and test/analysis of embedded systems. UNISIM-VP provides full system structural computer architecture simulators of electronic boards and System-on-Chip (SoC) using a processor instruction set interpreter. The whole software stack, consisting of the user programs, the operating system and its hardware drivers, is executed directly on the simulator.

The virtual platforms are modular because they are component-based software. Hardware components, written in the SystemC language [3], model the real target hardware components, such as CPU, memories, Input/Output, busses and specialized hard-

ware blocks. Hardware components communicate with each other through SystemC TLM-2 [4] sockets that act like the pins of the real hardware. The service components are not directly related to pure computer architecture simulation. They allow initializing and driving of simulation. Services range from debuggers, loaders, monitors and host hardware abstraction layer to make the simulator source code cross-platform.

3.5. Hardware faults and VP

Much work has been done on embedded system reliability in the last few years. Nevertheless, the embedded system designer needs tools that allow, on the one hand, to simulate the operation of the system in different operating conditions in order to correctly configure its architecture and, on the other hand, to study the impact of hardware faults on the behavior of the applications. In this project, we focus on transient faults, also called, Single Event Upset (SEU). According to several studies, these faults are more difficult to predict than permanent faults, which are detectable during the production phase [19]. Transient faults can appear in all units of the system and have several origins: the system operating environment, such as temperature or humidity, level of the supply voltage, vibration and electromagnetic waves.

In most of the existing solutions [15,14], a voting mechanism and redundant resources are used to deal with this kind of faults. Other solutions use error correcting codes (ECC). Experiments have shown that the cost in additional circuitry or additional execution time and energy consumption, of such solutions, can be very high. This has the effect of increasing the price and/or the electrical power consumption of the system. Moreover, the solutions proposed so far do not take into account the impact of the fault on the application behavior. In EQUITAS, the purpose is to study the impact of SEUs on the application behavior and their relationship with the V&V process.

3.6. Embedded software platform

The Automotive ECUs are provided with generic embedded software platforms that are based on the AUTOSAR standard [17]. AUTOSAR (AUTomotive Open System ARchitecture) is the automotive open software architecture standard. The first implementations of the AUTOSAR standard have shown the need to significantly improve the integration phases of the architectural components, by a tooled support of design, integration, development and validation phases. ISO 26262 [16] is an emerging standard for safety systems in road vehicles. ISO 26262:2011 defines a framework and an application mode. The activities, the methods to be used and the expected output data are also defined. The implementation of this standard will ensure the functional safety of electrical/electronic systems in motor vehicles. ISO 26262 is an adaptation of standard IEC 61508, taking into account the specificities of the automotive industry [16].

4. Technical challenges

The EQUITAS project presents two challenges. On the one hand, it incorporates test case generation tools, namely MaTeLo and DIVERSITY, which are based on two different theoretical approaches: stochastic and symbolic execution. On the other hand, EQUITAS aims to enhance the simulation environment, namely UNISIM-VP with fault injection capabilities in hardware components, and to interface it to tools for the automatic generation of test cases (DIVERSITY and MaTeLo) and the compliance analysis tool (ARTiMon). The implementation of a continuous tool chain from existing software components is a complicated and delicate task in general and particularly when the components are not designed from the beginning to work together.

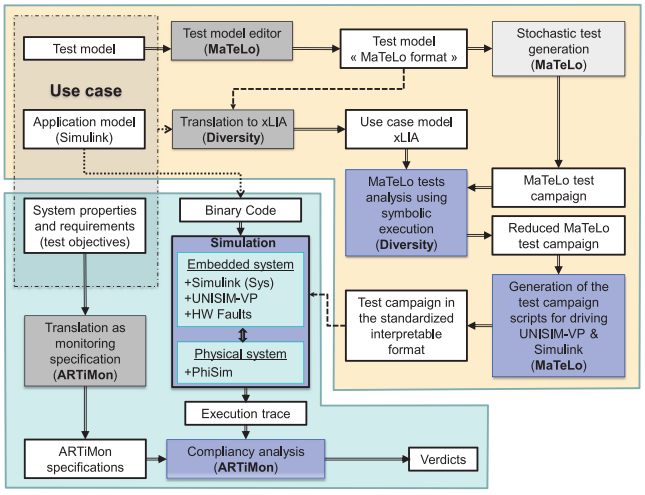

Fig. 4. EQUITAS tool chain.

Thus, the task of integrating the EQUITAS continuous tool chain requires:

- Design and development of bridges between tools
- Adaptation of tools for data exchange and synchronization between communicating tools
- Development of additional software components to achieve the expected features of the tool chain.

The inclusion of hardware faults in the early stages of an embedded system design process aims to increase significantly system robustness and reliability. This objective is confronted with several challenges, the main ones are:

- Identify relevant hardware faults and their associated representative models,
- Be able to trigger hardware faults during operation of the embedded system and representative test cases,
- Extend the UNISIM-VP simulation environment for modeling hardware faults in different units (processors, SRAM, bus, I/O interface, etc.) of the simulator, despite their heterogeneity.

In the field of automatic test generation, there is no integrated solution that provides the flexibility of stochastic tests (with very low computation time), and the accuracy of formal tests (but time and memory consuming). The implementation of this integrated solution is the main challenge here, i.e. to use DIVERSITY to analyze tests generated by MaTeLo, with acceptable performance.

5. EQUITAS tool chain

The implementation of the EQUITAS tool chain, presented in Fig. 4, requires many technical achievements (adapters and extensions):

- Achievement of a test-generation tool which merges techniques used in DIVERSITY and MaTeLo tools. MaTeLo is based on the test campaign model and automatically generates the most likely test cases for use over a long duration. These tests are analyzed by DIVERSITY, which uses symbolic execution applied to a formal model of the system, to remove any duplicates generated by MaTeLo. Duplicates are tests that belong to the same symbolic execution path.

- Interfacing ARTiMon and PhiSim which are respectively the monitoring tool and the physical environment simulator [20]. This link allows validation of the embedded system model at MIL level ("Model In the Loop"). The results of this step are used as an oracle for the following phases of the design flow, i.e. SIL ("Software In the Loop"), PIL ("Processor In the Loop"), and HIL ("Hardware In the Loop"). ARTiMon is wrapped into a 'MATLAB/Simulink® S-Function' and connected to the model. In order to feed the ARTiMon S-function, we added a multiplexer as an input of the ARTiMon S-function. The inputs of this multiplexer are links from variables playing a role in the monitored properties. A clock is added as an input such that the whole has the structure of a state vector with a time stamp. Thus, during the simulation, ARTiMon is fed with a flow of time-stamped states that build a trace which is analyzed on-the-fly.
- Extending the UNISIM-VP simulation environment so that it can be used to study the embedded system reliability. This extension focuses on the modeling of hardware faults (transient and permanent) in different simulated units (processors, SRAM, bus, I/O interface, etc.).
- Interfacing PhiSim to UNISIM-VP (hardware target simulator, i.e. ECU). This interface enables simulation of the automatic control loop (closed-loop).
- Interfacing the ARTiMon tool and UNISIM-VP. This interface enables the automatic analysis of test results to verify the non-functional properties (compliance analysis). This enables the automation and parallel execution of several test campaigns and analysis on the fly.
- Automating the execution of the test set on the UNISIM-VP simulator. This extension of the simulation environment allows the execution of test cases involving specific observable points. In addition, it enables parallel execution of several test cases on a distributed system.

6. Automatic test generation

The MaTeLo model is similar to a Markov chain, where the transition from one state to another is dependent on the current state and the probability associated with a transition. The generation of test cases in MaTeLo is performed through the exploration of the model using the probability of transition. The paths resulting from this exploration, are test cases.

A large number of tests, that is needed to reach a high level of reliability, can be generated very quickly by MaTeLo (Monte Carlo method). A greater part of these tests may be redundant if the model uses realistic probabilities (easily up 90% of redundancies).

The DIVERSITY tool generates a tree whose paths corresponds to the sequences of actions of the model (corresponding to the behavior of the system), by symbolic execution techniques. This tool has two main functions:

1 Model debugging. By analyzing the symbolic execution tree, the tool can detect over or under-specification, as well as problems such as deadlocks [9,10].
2 Automatic test generation based on the coverage of paths which exhibit all the behaviors of the system [11].

The path-coverage criteria realized with symbolic execution in DIVERSITY are used for the detection of duplicates across tests generated by the stochastic approach of MaTeLo. Indeed, when performing symbolic execution on a path, the result is a symbolic path, that is to say a succession of states labeled by symbolic variables. The variables being defined by parametric expressions (e.g. a variable V can be expressed with two parameters x and y, which gives $V = x + y$) and a path condition is the conjunction of the guards of the actions that were performed in the current path

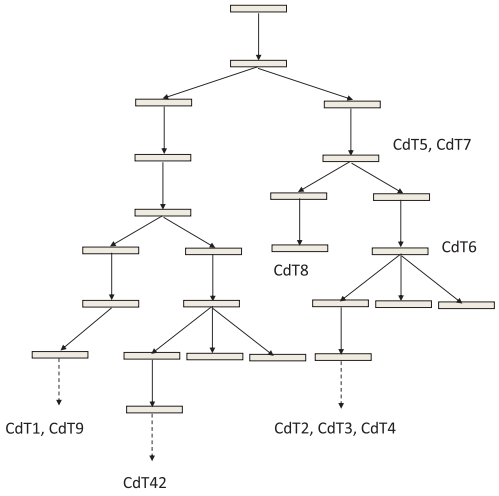

Fig. 5. Test case symbolic paths.

(in the example it can be $x > 0$ and $y >= 1$). They therefore define a set of potential numerical values (in our example we can deduce that V is defined in the area] $1, +\infty$ [). The different numerical values of the system input parameters that calculate the same symbolic path can be considered duplicates, since they allow to exhibit the same action sequences (in the example: all values of x such that $x > 0$, and y such that $y >= 1$).

DIVERSITY and MaTeLo models, viewed as black boxes, are equivalent. But as the models used by the two tools are different, it is necessary to have two corresponding models. For that, a model of the system which is a high-level description of the system is constructed from the specifications: this model must be executed by DIVERSITY and therefore it must be written or translated in the input-language of DIVERSITY. Then the input values generated by MaTeLo corresponding to the paths calculated during its stochastic process are applied on the DIVERSITY model and DIVERSITY classifies these paths with the symbolic equivalence criteria described above. Finally, we keep one representative per equivalence class.

This process is performed by the following steps:

(1) Initial step:
The first symbolic path is computed to cover the first test case
(2) Inductive step:
For a new test case to cover, there are 3 possibilities
(a) this test is included in an existing symbolic path
(b) this test case is partially covered: DIVERSITY computes a new symbolic path as an extension of the partial path which covered a part of this test case
(c) this test case is not covered at all, DIVERSITY computes a new symbolic path for this entire coverage

This process is illustrated in Fig. 5, where the redundant test cases would be: 9, 3, 4, 5, 7, and 6.

The project industrial partners (i.e. Continental and Sherpa-Engineering) consider that two test cases which activate the same functions are equivalent. This induces a natural order relation which means for example that CdT6 < CdT2 shown in Fig. 5. All the generated tests without duplicates have the same coverage during the test campaign as would be obtained if duplicates were present. This test strategy reduces the duration of the test campaign to guarantee at least equal operational reliability of the software.

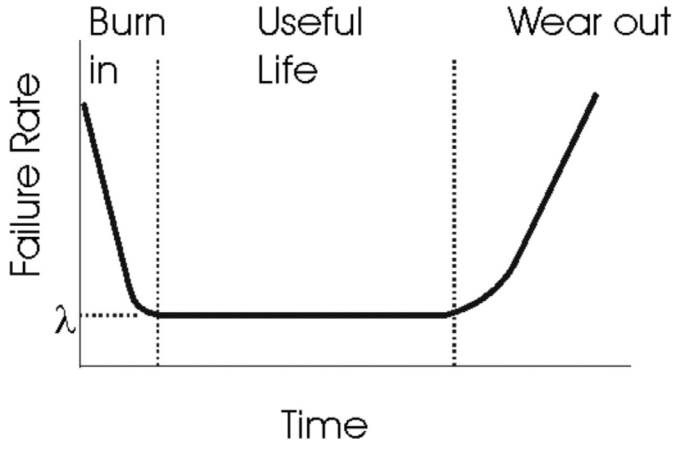

Fig. 6. Component life-time reliability.

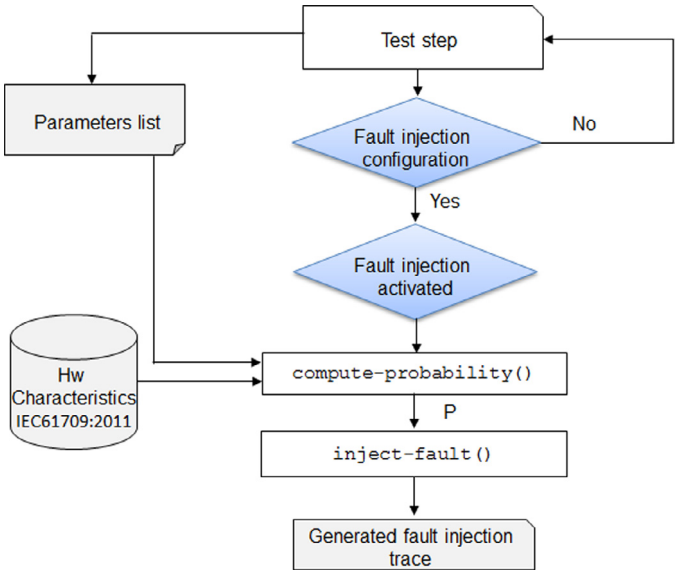

Fig. 7. Fault injection strategy.

7. Reliability and robustness analysis

In this section, we will describe a methodological approach to study the reliability and robustness of a complete embedded system by merging functional testing on a virtual platform and hardware fault injection. The use of virtual platforms aims to offer efficient instrumentation capacities that are not possible on a real system. In addition, it allows observation of system behavior in the presence of faults by the injection of hardware faults at different steps of the system engineering. The extension of UNISIM-VP focuses on the modeling of hardware faults (transient and permanent) in different simulated units (processors, SRAM, bus, I/O interface, etc.). New services and interfaces for hardware fault injection (Memory Fault Injection, CAN Fault Injection, etc.) have been developed. Test cases generated by the MaTeLo tool [18] represent a large number of real-life situations and their execution allows the user to obtain an experimental measurement of the system operational reliability.

7.1. Fault modeling

Fault modeling is a crucial step towards proposing fault injection techniques. Hence, building an accurate fault model is an imperative to represent correctly how faults occur in reality, which is not the case for random methods. The proposed fault model is mainly based on the Failure Modes and Effects Analysis (FMEA) technique. This technique provides a combination between failures and their impacts on the system. Quantitatively, the reliability of a device is expressed by the reliability function *R(t)* which is the probability that the device will operate correctly from time *zero* to time *t*. An alternative way of expressing device reliability is by its failure rate $\lambda(t)$, which represents the rate of failures per unit of time.

The bathtub curve shown in (Fig. 6) gives the evolution over time of the failure rate. It is a manufacturer's responsibility to ensure that product in the *'infant mortality period'* does not get to the customer. This leaves a product with a *'useful life period'* during which failures occur randomly, i.e. λ is constant, and finally a *'wear out period'*, usually beyond the product useful life, where λ is increasing.

During the *'useful life period'* assuming a constant failure rate, MTBF is the inverse of the failure rate and we can use the terms interchangeably, i.e.

$\lambda = 1/\text{MTBF}$

where MTBF is the Mean Time Between Failures (for reparable systems).

When the failure rate λ is constant, the reliability is statistically expressed by:

$$R(t) = 1 - \mathrm{P}(failure\ \ time < t) = exp(-\lambda t)$$

where *P(t)* represents the probability that the system will not conform to its specification throughout a duration *t*.

To study HW/SW system reliability and robustness, we need to generate a series of values λ_i to cover the entire lifetime and to take into account the functioning conditions. In the proposed framework we intend to use the IEC61709:2011 standard [21] for hardware fault quantification.

7.2. Failure rate prediction and robustness

Fault injection (Fig. 7) is driven by the test cases and fault scenarios. The list of these scenarios is defined according to the target application requirements and the test objectives. For instance, this list contains the different target memory regions (base address, size) where faults will be injected to avoid unused memory regions and to focus only on allocated regions.

The fault injection strategy is used to conduct two studies:

- System failure rate prediction
- System robustness measurement

To measure system failure rate, fault injection is driven by the probability *P* of occurrence of a fault for each scenario. The *'compute-probability'* function evaluates the probability *P* by using the reliability model depending on the target hardware component characteristics. Based on this probability, the fault injector takes the decision to inject a fault or not.

The HW/SW system is simulated several times and, for each test, the following set of data is collected:

- The test interval length and the number of failures observed in this interval
- The target area tested during the test interval
- The time at which each fault is injected and the application mode used

In contrast to the system failure rate prediction, to study system robustness, we want a deterministic injection strategy. In this case the simulation is only driven by the test case execution. The test

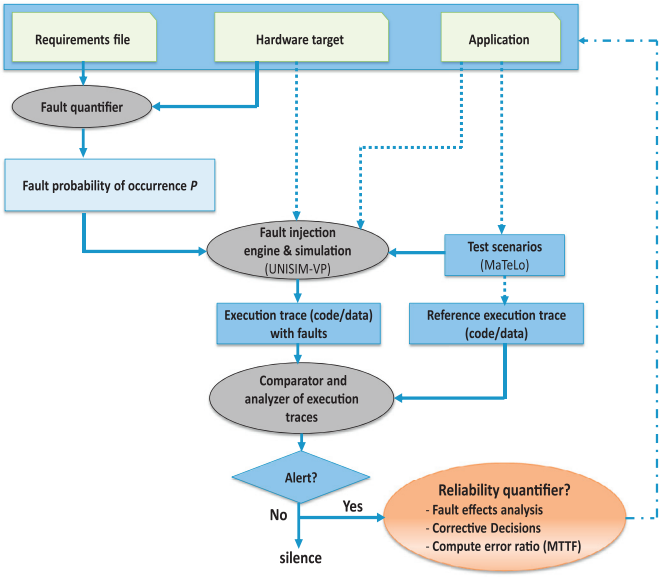

Fig. 8. Hardware fault injection in design and test flow.

case defines the relevant points of fault injection and activates the fault injector module at the appropriate time. The computed probability doesn't condition the fault injection but is used to evaluate the criticality of the injected fault. The simulation traces contain the fault probability information, as an annotation, to be used as an input for the reliability and robustness quantification task.

7.3. Fault injection approach

The proposed approach (Fig. 8), described hereafter, takes as inputs the application, the target hardware and a set of system requirements. The fault injection model takes into account the target hardware characteristics, using the IEC61709:2011 standard [21] as a reference. The following services and corresponding interfaces have been developed:

(1) **Fault quantification**. We define a physical model of the fault by studying the architectural features of the hardware target and a set of parameters/phenomena associated with it. Parameters such as: fine engraving, semiconductor, protection of the integrated circuit, age, temperature, frequency, etc. are taken into account. This model calculates the probability of occurrence of a fault.
(2) **Fault injection.** Once the fault model is defined, the test scenarios base is instrumented in order to fix the relevant injection points for the current use case. These injection points will serve as triggers during the scenario execution to inject hardware faults. The fault injector also takes as input a list of fault scenarios (i.e. the parameters list in Fig. 7).
(3) **Traces comparator**. Three time-stamped traces are generated as a result of the simulation: the instructions trace, monitored data trace, and injected faults trace. All these traces are compared with functional traces (without fault injection). The result is used during the reliability quantification.
(4) **Reliability quantifier**. Having the possibility to inject faults at the different HW components of the embedded system, we need to detect and analyze their impacts. This corresponds to the degree of propagation of the injected faults and the probability that the fault induces failure. Once a failure is detected, the embedded system structure and behavior is then analyzed. The purpose here is to compute the robustness level by analyzing the effects of the injected fault and its probability of occurrence in order to take corrective decisions, or to compute error/failure ratio when a system failure happens as a consequence of injected fault.

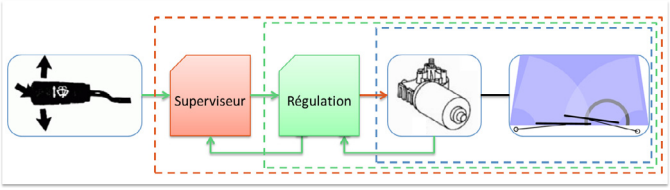

Fig. 9. Functional architecture of the windshield wiper control.

8. Formal dynamic analysis

The validation of the program behavior can cover several aspects. This may be to verify that the program does not reach an undesirable state. It may also mean that the program does not produce a succession of states containing an unwanted sequence. For example, the times between events are not respected or that events do not occur in the correct order.

The extension of UNISIM-VP by the ARTiMon plugin allows formal dynamic analysis of the behavior of an embedded system on virtual platforms. This coupling allows, firstly, automatically taking into account the requirements (transformed into properties to be verified) and on the other hand, the analysis at runtime, continuously and non-intrusively of the behavior of an embedded system (HW/SW).

This approach reduces the time and resources (human and material) required for validation. The task - incumbent on the validation teams - of interpreting the requirements of operation and turning them into automated or manual verification procedures is reduced to the extent that the ARTiMon plugin is able to automatically compile the formal requirements as automatic observers. This avoids misinterpretations, the ad-hoc encoding of costly verification procedures, that are not maintainable, or tedious manual verifications with the risk of significant errors. The automatic observers, generated by ARTiMon, work online: this enables the earliest detection of abnormalities (no need to wait until the end of the simulation to analyze and get a result) that accelerates the simulation-correction iterations. These observers work in main memory without traces being saved to disk: this avoids any access to the storage drive that could slow down the simulation and also saves storage resources (knowing that simulation traces usually reach large sizes).

9. Case study: windshield wiper

To evaluate the EQUITAS tool chain, CONTINENTAL has provided a real full BCM (40 functions) which was a challenge when it came to building the co-simulator. The project consortium has chosen to focus on the wiper function, illustrated by Fig. 9, as a case study for its representativeness of a typical control loop (18 inputs, 21 outputs, 15 parameters). The case study includes all components of the wiper function: from the action of the driver on the stalk switch until the real wiping of the windshield.

The wiper control function consists of two parts:

- The supervisor (ECU) interprets the driver requests through the stalk switch and defines the expected operation of the wipers, primarily in terms of mode: auto, manual, intermittent... and wiping frequency.
- The regulator controls the electric motor to follow the instructions indicated by the supervisor.

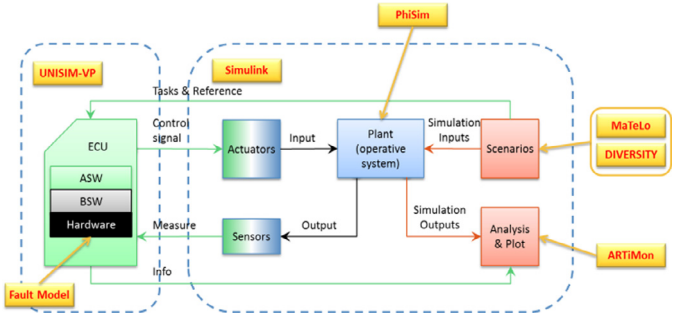

Fig. 10. Block diagram of the wipers control loop.

The block diagram in Fig. 10 details the architecture and some variables of the control loop. The figure gives details of the acquisition chain and the drive chain. It also shows the separation between the physical part and the control part.

The control part, called Body Control Module (or BCM) in the automotive terminology, will be integrated in the vehicle. The target architecture is a dual-core SoC MC9S12XEP100 from Freescale. The BCM is simulated by UNISIM-VP, and the physical part is stimulated by Phisim under Simulink.

In this project, a realistic MaTeLo model of the full wiping system was created. The driver can activate the windscreen washer at any time, as is the case in a real vehicle. This action can be immediate (pull and release) or have a certain duration (pull, maintain and release).

The MaTeLo model was calibrated so that the average duration of a test case is about 15 min. To do so, the creation of the model was based on several factors:

- The probability to loop compared with the probability to exit;
- The duration of the various "wait" interspersed in the model.

DIVERSITY tool was performed on the 5000 test cases submitted by MaTeLo on the basis of the specification studied in EQUITAS (wiping function). In this test suite, DIVERSITY detects approximately 3000 cases of unique test, that is to say, it detects 2000 duplicates. This reduction is encouraging, and demonstrates that the MaTeLo / DIVERSITY coupling works well, even though it is still not very significant from the industrial point of view. The rate of 40% should be significantly improved on larger test sets in the future.

ARTiMon checks some requirements during simulation. For example, it is required that, in intermittent mode (Mode == INTERMIT), between the arrival of the wiper in park position (**rise of** Wip_in_Park_Pos) and a request to move it again (**rise of** WipMvRequest) there exists a delay of value T.

The ARTiMon formalized property is:

when (Mode == INTERMIT) then
when ((**rise of** Wip_in_Park_Pos) then
(**not** WipMvRequest) **Until**$_{[T,T]}$ (**rise of** WipMvRequest)

10. Conclusion

In this paper, we presented the EQUITAS project. The project aims to propose solutions at two levels in the design of complex embedded systems for automotive systems. First, as currently no integrated solution addresses all of the issues raised by test case generation, EQUITAS offers stochastic and symbolic execution approaches by coupling the DIVERSITY and MaTeLo tools. Second, EQUITAS enhances the simulation environment, namely the time-accurate UNISIM-VP by fault injection capabilities in hardware components. In EQUITAS, we also deal with the interfacing of UNISIM-VP to tools for automatic generation of test cases (DIVERSITY and MaTeLo) and the compliance analysis tool (ARTiMon).

For the first contribution, by coupling DIVERSITY and MaTeLo, all duplicates (40% of the test campaign) have been removed from the generated tests without impacting reliability. This strategy can ensure the relevance of the remaining tests, which should increase the operational reliability of the software.

For the second contribution, new generic services and interfaces for hardware fault injection have been developed and added to a virtual platform. Contrary to existing tools, fault injection and analysis in EQUITAS take into account the physical model of the system and automatically generate the traces for performance estimation. The developed approach (Fig. 8) has been used to develop a runtime adaptive architecture [22].

Two components (UNISIM-VP and DIVERSITY) of the tool chain, provided by CEA, are open-source. For ARTiMon, Phisim and MaTeLo, respectively provided by CEA, Sherpa-Engineering and ALL4TEC, a licensing for research purposes is possible.

The next steps of the project will be dedicated to:

- The definition of a V&V methodology based on the virtual platform and to investigate the compliance of the EQUITAS tool chain with the ISO26262 standard.
- Quantitative and qualitative assessment of the tool chain by the project's industrial partners.

References

[1] U. Abelein, Dhn. H.Lochner, S. Straube, "Complexity, quality and robustness - the challenges of tomorrow's automotive electronics", DATE 2012.
[2] D. Potier, "Briques génériques du logiciel embarqué", 7 octobre 2010. [Online]. Available: http://www.entreprises.gouv.fr/etudes-et-statistiques/rapport-dominique-potier-sur-briques-generiques-logiciel-embarque-2010.
[3] IEEE 1666™ Open SystemC Language Reference Manual. [Online]. Available: http://standards.ieee.org/getieee/1666/index.html.
[4] TLM-2.0 Reference Manual, . [Online]. Available: http://accellera.org/images/downloads/standards/systemc/TLM_2_0_LRM.pdf.
[5] UNISIM-VP, [Online]. Available: http://www.unisim-vp.org.
[6] R. Leupers, F. Schirrmeister, G. Martin, T. Kogel, R. Plyaskin, A. Herkersdorf, Martin Vaupel (2012). "Virtual platforms: Breaking new grounds". Design, Automation & Test in Europe Conference & Exhibition (DATE).
[7] IBS, GSA ROI Calculator, Synopsys, Inc.
[8] T.D. Schutter, B. Software, Faster! Best Practice in Virtual Prototyping, Synopsys, 2014 Inc., September.
[9] J.-P. Gallois, C. Gaston et A. Lapitre : "AGATHA, un outil de simulation symbolique", (in French) AFADL 2004.
[10] N. Rapin, C. Gaston, A. Lapitre, J.-P. Gallois, Behavioural unfolding of formal specifications based on communicating extended automata, ATVA (2003).
[11] D. Bahrami, A. Faivre, A. Lapitre. DIVERSITY-TG, "automatic test case generation from Matlab/Simulink models". 6th European Congress on Embedded Real Time Software and Systems (ERTS2 2012), Toulouse, France, February 2012.
[12] O. Maler, D. Nickovic, Monitoring temporal properties of continuous signals, Proc. FORMATS-FTRTFT 3253 (2004) of LNCS.
[13] O. Maler, "Checking timed and hybrid properties", Phd Thesis, Verimag, 2008.
[14] I. Alouani, S. Niar, F. Kurdahi, M. Abid, Parity-based mono-copy cache for low power consumption and high reliability, IEEE Rapid System prototyping conference, 2012 (RSP'2012).
[15] A. Chakraborty, H. Homayoun, A. Khajeh, N. Dutt, A. Eltawil, F. Kurdahi, E < MC2: Less energy through multi-copy cache, CASES'10, october 24–29, ACM, Scottsdale, Arizona, USA, 2010.
[16] ISO 26262:2011. Road vehicles - Functional safety. International Organization for Standardization (ISO), TC 22/SC 3.
[17] [Online]. Available: http://www.autosar.org/.
[18] [Online]. Available: http://www.all4tec.net/MaTeLo/homematelo.html.
[19] S. Borkar, Designing reliable systems from unreliable components: the challenges of transistor variability and degradation, IEEE Micro 25 (6) (2005).
[20] Phisim, [Online]. Available: http://www.sherpa-eng.com/index.php?lang=fr&Itemid=785
[21] IEC 61709: 2011 Electric Components-Reliability-Reference Conditions for Failure Rates and Stress Models for Conversion (2nd ed.). Geneva: International Electrotechnical Commission.
[22] M. Hussein, R. Nouacer, A. Radermacher, A model-driven approach for validating safe adaptive behaviors, digital system design 2016 (DSD'2016).