Virtual Prototyping for validation of functional architectures
A Kaske, G François, M Maier, Alexander Burst

To cite this version:
A Kaske, G François, M Maier, Alexander Burst. Virtual Prototyping for validation of functional architectures. Conference ERTS'06, Jan 2006, Toulouse, France. hal-02270348

HAL Id: hal-02270348
https://hal.science/hal-02270348
Submitted on 24 Aug 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Virtual Prototyping for validation of functional architectures

A. Kaske¹, G. François¹, M. Maier², Alexander Burst²

¹: ETAS SAS, 1, place des Etats-Unis, 94588 Rungis Cedex France
²: ETAS GmbH, Borsigstrasse 14, 70499 Stuttgart Germany

Abstract: This paper will present a new approach how to use virtual prototyping to validate functional architectures. The approach points out how functional architecture can be set up and tested in order to get highly validated data before making final architectural/design decisions.

The introduction will speak about the current facts, the constrains and the tendencies to create a common and understandable picture in which the proposed solution may be implemented.

Then we will give a description and definition of MiLMiL / SiLSiL / HiL, rapid prototyping and virtual prototyping to avoid confusion.

The main part will discuss functional and logical architecture, the mapping of both as well a the variants that have to be considered. It will be shown how validation by virtual prototyping can help to define and proof architectural decision.

Finally we will give a short outlook on how to migrate from virtual prototyping systems via rapid prototyping via fullpass and bypass to HiL applications.

The conclusion summarizes the advantages compared to the current situation.

Keywords: Architecture – Languages, Process, Methods and Tools – Safety, Security – Standardization, AUTOSAR

1. Introduction

The current situation in the automotive industry is marked by a strong movement to new development processes that will guarantee higher quality of (especially) the electronics part (hard and software) of the vehicles. From a very general point of view two major problems are identified, diagnostics (badly linked with functional development and - due to long time missing standardization - incoherent usage of data etc.) and incomplete functional specification and validation. Diagnostics aspects are not part of this paper.

Functional development, independent whether it is in Powertrain/Drivetrain, Chassis or Body Electronics suffers today from late validation of functional architecture decisions. Sometimes these decisions are first tested with or after integration of all subsystems in the vehicle (this is especially true for rare variants of a vehicle). Errors discovered in this late development stage are either very costly to fix or even unsolvable Two tendencies enforce these problems:

- First, the upcoming interdomain functions, e.g., for driving assistance such as Lane Departure Warning (Chassis, Powertrain and Body Electronics ECUs are involved) and
- Second, the tendency to reduce the overall number of ECUs creates the force to integrate functionality from different ECUs (even from different ECU suppliers) into one ECU which creates additional problems like IP protection, responsibility and common software architecture.

Several initiatives are currently on the road where AUTOSAR is certainly the most important with the highest impact on the whole development cycle. We will show how our approach will deal with these different constrains and how it will provide a cost effective method to get higher quality, less errors and well tested functional specifications.

2. Definitions

This paragraph will explain and define what we understand with the mentioned terms.

2.1 MiL – Model-In-the-Loop

Within MiL all functional components (maybe even implemented software components) are plugged directly to a virtual function bus together with all necessary environment models. No other basic software (BSW) is involved. A real time OS (RTOS) may insure that the entire simulation runs in real time but that is not necessary. In a first step everything runs on the same PC. For more enhanced applications it might be necessary to split the functional models from the environment models and to execute each on different PCs. This is relative easily possible if standardized network communication is used for the Virtual Function Bus like COM/DCOM, .NET or CORBA etc.
2.2 SiL – Software-In-the-Loop

Different to MiL, SiL will use all necessary BSW in order to simulate the whole ECU software. Together with the BSW also real-time behavior/execution is introduced. This might lead to additional restrictions especially for bus communication like CAN (bandwidth problems, priority problems, scheduling problems). Nevertheless a virtual signal bus is still used. In this case it is a virtual signal bus. As for MiL in a first step everything is executed on the same PC but as already mentioned for MiL it is possible to separate software and environment into several PCs.

With the integration platform that we propose it is even possible to introduce a real network like CAN\(^1\) and to use the VSB only for sensor and actuator signals. Such an approach maybe called semi-physical SiL application.

2.3 HiL – Hardware-In-the-Loop

Different to SiL, for HiL applications all sensor and actuator signals will be created physically through the environment model(s). Everything happens in real-time.

---

\(^1\) Later even FlexRay

---

The software might be executed on RP systems or on ECUs.

A HiL system may integrate real mechanical or electrical subsystems, e.g., piezo-injectors.

2.4 Virtual Prototyping

VP means more or less everything is executed on the PC as in MiL and SiL applications (in real time). Pure VP has no physical signal or bus communication but as mentioned above its border is fuzzy (e.g. SiL with real CAN bus – semi-physical SiL).

2.5 Rapid Prototyping

RP means generation of physical ECU signals and communication in real time either on dedicated RP hardware or even on a PC. RP applications are able to deal with real sensors and actuators and are often located directly in the vehicle.

RP can be split into 3 different types of RP, bypass, fullpass and semi-fullpass which will be detailed in the following paragraphs.

2.5.1 Rapid Prototyping - Fullpass

Full-pass RP systems are able to generate and measure all I/O signals of an ECU application in real-time. RP systems are also able to execute the functional specification in real-time. It is a complete replacement of an ECU with regard to the specific application. Normally no optimized or implemented software is needed and therefore all computations are done in floating point. For the proposed integration platform it is even possible to execute implementation-ready ECU ANSI C-code. This makes lot new and interesting applications possible e.g. comparative computing where a functional specification is executed in parallel to its implemented software in order to identify easily differences. Finally it is of course possible to reuse implemented software also in the other RP use cases like bypass or semi-fullpass.

2.5.2 Rapid Prototyping - Bypass

Bypass RP systems are a combination of an ECU with an additional RP simulation target. The ECU executes its standard functionality. The RP system is connected via a dedicated communication link to the ECU and executes synchronously the new specified functionality if necessary with additional IO.

This is the most common RP system due to the fact that most new functionalities extend existing systems (ECUs).
2.5.3 Rapid Prototyping – Semi-Fullpass

Semi-fullpass RP systems are somehow very special bypass systems since all applicative software is executed on the RP system and the ECU is just used for signal conditioning and low level I/O management.

3. Electronic Vehicle Architecture

This paragraph is dedicated to give a brief introduction of the 3 major steps of the development of electronic vehicle architectures.

3.1 Functional Architecture

During functional development the OEM has to take several architectural decisions. First the OEM sets up the functions to be available in the vehicle. Additionally the OEM starts to define the variants that will exist for the vehicle (variants can be specific to regional markets, to functional components such as engine/gearbox, to coachwork etc.).

In parallel to the functional architecture an ECU network the logical or electronic architecture has to be designed. This network has to deal with constrains like:

- number of variants
- scalability
- redundancy – security
- cabling efforts
- reuse of existing (sub-) systems
- etc.

3.2 Electronic Architecture

Finally the functional architecture must be mapped to the electronic architecture, i.e. the ECU network(s).

Most problematic is here the number of variants that may exist for a vehicle family (up to 2000 software / hardware combinations = variants). Due to the fact that only a very limited number can be tested with current methods it becomes clear that at lot of errors are discovered first when a variant is build in a real vehicle for an end customer which is too late and very expensive if errors are detected.

---

2 Test and validation is mostly done via the test vehicle fleets with about 100-400 vehicles. That means a maximum of about 400 variants could be theoretically tested.
There may occur a number of problems when applying different mappings like:

- Bus overload due to communications between software modules, that are no more on the same ECU.
- General or sporadic ECU overload like scheduling problems (task activation missed, deadline not reached etc.).
- Bus communication may introduce a delay that may cause some algorithms to become unstable which may end up in a bad reactivity of a component, e.g. electro-mechanical door openers.
- Additional bus communications may block communications with other ECUs due to priority problems which show suddenly an erroneous behavior (without any changes on this particular ECU).

### 4. Validation by Virtual Prototyping

Virtual Prototyping can give people the chance to simulate very early any functional architecture with few effort (hardware costs and time). Virtual prototyping makes this possible especially when it is combined with standardized software architecture such as AUTOSAR proposes it. Using a standardized software architecture is a key to distribute easily software between the virtual ECUs (simulated by a standard PC) as well as on other hardware execution platforms (rapid prototyping systems or ECUs). These virtual ECUs have already a real time behavior that is relatively close to the final ECU (at least when comparing it to current simulation methods) and even more important, they are using the same/similar basic software to insure that all applications are dealing with the same mechanisms independent whether they are executed in a VP environment or on a RP system or on a real ECU.

#### 4.1 Virtual Prototyping Systems

The very first step in prototyping a functional system is to integrate all functionality together with the environment model. This gives information about general aspects of the functionality

- does in principle the function do what the developer expects,
- does the co-working of functions work as expected,
- etc.

This first step does not necessarily answer questions about the look & feel of the function for the end customer because mostly at this point nothing happens in real-time – due to the fact that it is just a MiL application.

A next step could be to introduce “real-time” in the sense that all functions and environment models are executed in real-time. This helps to identify problems of, e.g., look & feel for an end customer – pressing a button should lead to an immediate reaction of the system (vehicle). Immediate is unfortunately a relative term because the perception of human beings of “immediate” depends whether it is an acoustic, a vision or a mechanical feedback. In general we can say that a feedback should be given with less than 250ms in order to be perceived as immediate.

This step can be cut into 2 steps, a first one just creating a real-time MiL application and a second step converting this to a SiL application. For large projects this may introduce a certain additional effort but it helps enormously to identify to root cause of an error because each step modifies just one parameter.

- MiL to real-time Mil – modified parameter is the introduction of real-time
- Real-time MiL to SiL – modified parameter is the introduction of basic software

Next important step is to distribute the modules/functions to a dedicated (simulated) ECU and to create by this a real (pseudo) ECU-network.

- SiL to distributed SiL – modified parameter is the introduction of bus communications and its associated delays etc.

At that point all modules/functions mapped to the same ECU have to be integrated together with the BSW to get the executable code for a pseudo or real ECU.

---

3 Most used CAN drivers do not deal correctly with priority handling of CAN messages which leads to an unpredictable network behavior.
It is important to understand that the proposed virtual prototyping environment is highly scalable. That means it is possible to simulate more than one ECU on only one PC as well as one ECU on one PC which makes it easy to setup variants automatically as well as to test them automatically. To do this it is necessary that the distribution and integration platform provides an automation interface like COM API for example. For the test environment it is not important to know on which ECU a function is executed. The test script just needs to know that a function together with its measurements and calibrations exists. Therefore it is completely transparent to the test environment and no modification is necessary to execute the test scripts on different functional architecture variants.

By this means it is possible to test all variants in an early development phase (over night, during the week end) and to discover the major part of errors related to architectural decisions quite easily

4.2 Migration of VP systems to RP systems

Another source of problems is the non-continuous use of tools and methods during the development process. Normally multiple conversion/migration of data and description formats is necessary because the MIL/SIL tool environment is quite different and incompatible to the RP environment and even worse, different to the one of ECU software development environment.

The proposed approach is able to overcome most of the mentioned insufficiencies because it can assure that the same software architecture as well as the same software components (at least the hardware independent ones) are the same. A migration from a SIL system (independent whether it is distributed or not) to a RP system makes for the proposed integration platform no other difference than compiling all sources (C-code) for a different target which is in this case a RP system.

5. Generic Distribution and Integration Platform

The main assumption (as already mentioned) for a generic distribution and integration platform is that it should be possible to convert a functional specification into C-code (pure ANSI C). This is true for almost all currently used BMT tools.

An additional assumption (as already mentioned) is that it should be possible to use a standardized software architecture independent of the execution platform (execution platform can be the PC, a RP system or an ECU). This seems to be true with the upcoming AUTOSAR standard and the software architecture imposed by the standard.

Finally such an integration platform has to use as much as possible well established standards to make reuse of existing tools and data and to be therefore investment save. This is also necessary to integrate the platform easily into existing tool chains and processes.

With the INTECRIOS Product Family, ETAS offers a prototyping environment which already supports the integration of functions modelled in either Matlab/Simulink or ASCET or written in ANSI C-Code. INTECRIOS may serve as a starting point for the proposed integration platform.

For the integration of a module/function 3 different files are necessary:

- **C-code (ANSI)**
  - the executable representation of the functional specification or even its real implementation.
- **A2L (ASAM MCD-2MC)**
  - the description of internal variables of the C-code (functional specification or implemented software) that can be measured and/or calibrated (e.g. via standard MC tools).
- **SCOOP-IX (XML – ETAS)**
  - the description of messages send and/or received by the component/function as well as its internal behavior, the scheduling information.

When using these 3 files for each component then the proposed integration platform is able to configure complex software systems just by connecting message ports between different components. This defines the data/message flow.

---

4 If the executed test scripts push the systems to its limits.

5 INCODIO by Systecs – available mid 2006 – current planning.

6 Optional, in order to provide better know-how protection it will also integration already pre-compiled binary code- then integration becomes target specific.

7 Additionally AUTOSAR software component descriptions will be supported in the future.
It is important to understand that the proposed approach does not mean that all modules/functions need to be available in the same format (e.g. as Simulink model). In contrary - this approach is able to integrate simultaneously application software components generated by Simulink with those generated by ASCET with hand-written C-code or C-code from any other C-code generator. Due to the fact that the approach integrates on C-code level it is easily possible that the integration platform is independent of the tools and version used to generate this C-code.

The possibility to integrate models/functions (respectively their C-code) from several e.g. Matlab/Simulink version in parallel makes the cooperation between several partners of a development much easier then today, where they have to agree on a specific tool version.

The possibility to integrate models/functions with existing C-code avoids the necessity to create e.g. Simulink or Statemate models for legacy code which might be available even as real ECU code. This is an important feature to enable the introduction of the proposed integration platform because the necessary pre-requirements are pretty low. Mostly the integration of a MIL/SIL approach as described above into broader company processes is blocked by the effort to be spend to develop models for all existing (electronic) functionality of a vehicle, which is normally about several decades of man years.

Another important step is the definition of the scheduling behavior. In the proposed integration platform this is possible via a graphical representation of the underlying OSEK OS.

The big advantage to use right from the beginning an OSEK OS is (even for PC based simulation in MIL/SIL applications) that only those OSEK OS features are used which will be also available on the final target (ECU). No break between MIL/SIL configuration and RP system/ECU configuration. If a different OS/scheduler would be used, special modeling and configuration rules need to be applied in order to guarantee that the MIL/SIL simulation can be transferred easily to the RP system/ECU which would make the process more fault-prone.

The integration platform finally provides all necessary hardware configuration and generates the glue code to create a totally standardized software architecture (based on AUTOSAR when AUTOSAR has finished its work).

Additionally the proposed integration platform offers some other interesting features like the dynamic reconfiguration of the message/data flow. For the ECU a similar feature is known as internal bypass but the integration platform is much more flexible for such a reconfiguration.

6. Migration to HiL systems

This integration platform supports RP systems as well as VP system. From a top level view it makes no difference whether the system is a virtual prototyping system or a real distributed rapid prototyping system or even a mixture between both.

---

8 E.g. TargetLink or Statemate/Rhapsody in Micro C etc.
9 Estimation: 100 - 200 electronic functions per vehicle – development of a tested/validated model for an existing functionality is in average between \( \frac{1}{2}\) to 1 man year.
10 Glue code is a small code fragment to adapt API calls etc..
This makes it easy to go even one step further and to migrate a (distributed) SiL system into a HiL system. The HiL system may be setup with RP systems running as fullpass systems as shown below. Any other setup with ECUs or with bypass systems is also possible. This makes it easily possible to adapt each sub-system to its degree of maturity – ECUs where ECU are already available, bypass systems where most of an ECU is stable and only some new functionality needs to be integrated and fullpass systems which are still in a very early development state.

This possibility to migrate individually each sub-system from one simulation level to the next is another dimension of scalability that this integration platform will propose.

7. Conclusion

The virtual prototyping approach combined with a distribution and integration platform enables very early validation of functional architecture decisions. This permits to identify weaknesses and errors of the architectural decision, to fix them or to modify the architecture when it is still possible and especially much less expensive than with the current approaches.

The proposed integration platform offers a multi-dimensional scalability which makes an easy integration into existing company processes possible.

Thanks to the integration on C-code level the integration platform becomes independent of BMTs and their versions. It is open to any other manually or automatically generated C-code as well.

Finally the independence from and transparency of the hardware execution platform gives the customers a high flexibility to chose adapted means for their problem.

Overall the virtual prototyping approach will help to increase the overall quality of the electronics systems in vehicle what is currently the main concern of most OEMs.

8. Acknowledgement

The authors acknowledge the contribution of their colleagues of the whole ETAS Group to this work as well as the related customers confident to work with us on these sensible topics.

10. Glossary

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BSW</td>
<td>Basic Software</td>
</tr>
<tr>
<td>ERTS</td>
<td>Embedded Real Time Software</td>
</tr>
<tr>
<td>HiL</td>
<td>Hardware-In-the-Loop</td>
</tr>
<tr>
<td>MiL</td>
<td>Model-In-the-Loop</td>
</tr>
<tr>
<td>OS</td>
<td>Operation System</td>
</tr>
<tr>
<td>RP</td>
<td>Rapid Prototyping</td>
</tr>
<tr>
<td>RTOS</td>
<td>Real Time OS</td>
</tr>
<tr>
<td>SiL</td>
<td>Software-In-the-Loop</td>
</tr>
<tr>
<td>VFB</td>
<td>Virtual Function Bus</td>
</tr>
<tr>
<td>VP</td>
<td>Virtual Prototyping</td>
</tr>
<tr>
<td>VSB</td>
<td>Virtual Signal Bus</td>
</tr>
</tbody>
</table>