AUTOSAR BSW in real life: A summary of the last years starting and putting projects into production
Matthieu Courrier, Helen Clergeau, Alain Calvy, Philippe Favrais, Michel Lecat

To cite this version:
Matthieu Courrier, Helen Clergeau, Alain Calvy, Philippe Favrais, Michel Lecat. AUTOSAR BSW in real life: A summary of the last years starting and putting projects into production. Embedded Real Time Software and Systems (ERTS2012), Feb 2012, Toulouse, France. hal-02189890
AUTOSAR BSW in real life: A summary of the last years starting and putting projects into production

Matthieu Courrier, Helen Clergeau, Alain Calvy, Philippe Favrais, Michel Lecat
Body and Security, Continental Automotive France SAS, Avenue Paul Ourliac, 31036 Toulouse

ABSTRACT: This paper will present the major challenges we had to overcome during the AUTOSAR Basic Software Roll-Out within Continental Body and Security: high complexity in terms of architecture, integration and configuration, multiple suppliers, as well as AUTOSAR specification maturity and diversity.

KEYWORDS: AUTOSAR Basic Software, BSW, Software Integration, Basic Software configuration, Test tooling

Introduction

The introduction of the AUTOSAR standard in the Automotive industry is one of the major steps forward of the last decade. AUTOSAR came as a common and concerted solution to a lot of recurrent problems in our business field. AUTOSAR helps to improve the reusability and the portability of the software components. This is a major requirement in our business, where the hardware platforms are changing very quickly.

AUTOSAR also specified the architecture of the Basic Software (Figure 1). Each module with their interfaces and interaction has been standardized, from the higher to the lower layer.

Continental has been an active participant in AUTOSAR since the beginning. Today, in the Body and Security business unit we have several projects in production with AUTOSAR software, and most of the new projects start with an AUTOSAR Architecture.

During the five years of ramp-up we had to arise to several challenges:

- rethink our way of working in the field of Software development and integration
- introduce software supplier management
- simplify the software configuration
- continue to ensure consistent quality and functional behavior
- build and maintain our technical expertise

Each chapter will explain what we did to succeed.

AUTOSAR Basic Software: The revolution

For us, as providers of Basic Software solutions to projects, the migration to AUTOSAR has been a big change.

Instead of developing complete BSW solutions, we now put together SW components from several suppliers, and perform the global configuration.

The Microcontroller Abstraction Layer (MCAL) is provided by the silicon manufacturers in order to benefit from their microcontroller expertise. Some
Basic Software modules are taken as off-the-shelf components from internal and external providers (Figure 4).

Finally, the only specific development that belongs to each product family is the creation of the ECU Abstraction Layer. That is to say, how the main controller activates remote input-output (on external multiplexers) or communicates with external smart devices (Custom ASICs, Watchdogs...).

This was a really challenging revolution: the first Body and Security products based on this architecture are already in production and an increasing number will be sent to production in the next years (Figure 2).

Integration Cycles are more complex due to the high number of modules and the increased number of providers. Therefore the integration in each customer project takes more effort.

We have introduced an organization by Stack: Memory, IO, System, and Communication. Each stack includes the modules from top to bottom, so that we have decreased the number from 67 modules to 4 pre-integrated stacks. A stack is a coherent set of code and documents covering all layers. This means that each integration on the Stack level is done only once and not in each customer project. This also gives a coherence of the modules within a Stack and acts as a reference for the validation (Figure 3).

We have also created a test application including all the Stacks pre-configured to be used as a reference in project integration. This organization enables us to deliver to projects quicker and with increased quality.

The second key topic is the Supplier Management

Releases of supplier modules come on a basis of 4-6 months which doesn’t correspond to the development cycles and customer expectations which are typically every month or even faster

Less development, more integration

Providing Basic Software for AUTOSAR is an activity mainly based on the integration of components from different suppliers, and therefore our daily work has changed from performing the development of components to essentially doing the integration of those components.

If we put to one side the configuration aspect that will be discussed in a subsequent chapter, there are some key topics to discuss here.

The first one is the integration process.

Figure 2: AUTOSAR-based software in production

Figure 3: Organization of the AUTOSAR BSW in Stacks

Figure 4: Diversity of the providers of the BSW code
The MCAL development is synchronized to the introduction of new microcontrollers within Body and Security. It is started at the same time as a new platform, therefore getting mature MCAL in this context is a challenge for our supplier.

Due to this we need to be able to apply ‘hotfixes’ which are temporary solutions delivered quickly without full quality. It is also necessary to have a well defined change management process in order to keep the full traceability of all updates.

A close relationship and defined process with the providers of the external modules has proven to be mandatory.

Configuration: Dealing with complexity

Since the beginning, the AUTOSAR model is continually evolving. The AUTOSAR consortium is developing a major release each year and intermediate releases when blocking issues arise. Today, no less than 5 usable releases are available (2.1/3.0/3.1/3.2/4.0).

One subject of major interest, uplifted by the need of being in conformity with the AUTOSAR standard, is the configuration of AUTOSAR software. This problem is exacerbated by the large number of cohabiting AUTOSAR releases and the complexity of the specification issued by the AUTOSAR consortium. Our first AUTOSAR experience, namely the development of a full ECU, showed the configuration as a critical point in the development process, as effort for configuring the Basic Software grew to a critical amount.

To give a comparison, the configuration effort of the AUTOSAR Basic Software is three times higher than the effort required to configure the previous-generation (non-AUTOSAR) Basic Software, with a higher error rate, which in some cases turns into Software bugs.

Today, the emerging market of AUTOSAR tooling does not propose any solution to apprehend adequately this issue. Our solution was to develop the Continental Body and Security SPARC tool (SPARC for “Software Platform AUTOSAR Configuration”).

The goals for this tool were:

- simplify the configuration in order to keep configuration effort for the AUTOSAR Basic Software comparable to the effort required on the previous generation;
- eliminate configuration related errors by early detection during the configuration phase, removing as much as possible opportunities to introduce errors, and provide on-the-fly consistency checking;
- abstract the AUTOSAR version and as far as possible vendor specific parameters focusing on the need for configuration and not on its variation in a particular version of the implementation;
- organize and display the configuration settings in a global and logical way. This presentation has eliminated redundant parameters and parameters whose value can be calculated or preconfigured.
- abstract the microcontroller specifics from the end user.

The Body And Security SPARC tool is based on a global abstract model representing a whole AUTOSAR Stack configuration. Again, the
organization in Stacks (IO, Memory, COM and System) is used to focus on the functionalities.

From this abstract configuration, the tool populates all the AUTOSAR configuration files with scripts that read from a data file (based on the global abstract model) and write into the AUTOSAR configuration files (Figure 5, Figure 6).

The tool additionally allows a capitalization of our experience with configuration errors. This is done through the “Consistency Checkers”, which is the name for some additional static tests that we add to the abstract model. They check that a parameter is correct in the context of the other module configuration.

With these consistency checkers the errors are detected in the early stages of configuration. The process is in constant improvement. New checks are permanently added to the abstract model, based on our experience.

The reduction of the number of configuration parameters has up to now decreased the configuration time by at least three.

Stack Tests: A must to insure the quality of the delivered SW

The test activity is of prime importance. It helps to detect the configuration and integration problems. Putting together components that implement perfectly the specification may still result in software malfunctions. Therefore intensive testing of the Basic Software is mandatory.

This activity must be done quickly, and often, because the time from receiving modules to delivering them to customer projects is short. This activity must cover all possible use cases. It also must cover in priority the uses-cases that are relevant for our Body and Security applications.

On top of this, due to the fact that Basic Software is deployed in a lot of projects with different settings, we must cover as many configurations as possible.

The test activity benefits from the work we have done on the stack organization and on the configuration tool. First, the Basic Software is tested on a Stack level, which perfectly fits with the way we have organized the AUTOSAR modules. Secondly, our abstract configuration model allows us to generate some input files for our test tool.
The build process (Figure 7) generates:

- an AUTOSAR application, containing the configured Basic Software that we need to test
- a Test Configuration script, that will be used by our test tool

For each stack, the software experts define a set of functional high level tests. They are based on the AUTOSAR specification (requirement coverage) and on our basic software know-how (specific test cases coverage). The generated test configuration is used to adapt the global test execution to the compiled application in order to select which tests are relevant in the given configuration.

This is based on an in-house tool developed to fit to these requirements (Figure 8).

These tests are always performed in a Hardware In the Loop (HIL) environment. When it makes sense however, a Software In the Loop (SIL) environment is available and is used for quick regression checks, debugging and fast coverage analysis.

![Figure 8: Test tool environment](image)

The advantages of this methodology are that we can re-run all tests in a very short time when we receive a new module update, allowing full regression tests at each delivery. For example for the Memory Stack now takes one day from receiving modules to delivering the Stack.

Technical expertise: Still needed?

In the past our team was responsible for the basic software from development to customer integration. We were therefore technical experts for the low layer drivers and completely mastered the microcontroller functionality as well as the software functionality needed for the customer projects.

AUTOSAR specifications have been written by technical experts, who have each their own experience of the functionalities. They made it possible to deploy the AUTOSAR concept on a device that controls an Engine as well as on a device that switches on and off some bulbs. This versatility of the concept has been reached, but this has a cost. The code is more complex and contains more functions. There are sometimes several ways to do the same thing. Each project has its specificities (a gateway must be fast, an engine control must be precise, a body control module must reach very low consumption in low power, an ABS must reset very quickly...).

For each kind of ECU there is a way to deploy and configure the AUTOSAR BSW in order to fulfill the requirements.

With the introduction of AUTOSAR we no longer develop all the modules, however our experience has shown that it still remains necessary to keep:

- the knowledge of the overall functionality in order to support the projects in using AUTOSAR in the most optimized way for their needs, to detect any missing features in the AUTOSAR specification and to support for configuration and integration.
- the detailed microcontroller knowledge in order to support our projects to use the controller to its fullest, either linked to MCAL functionality or for our in-house IO HW Abstraction and Complex Device Drivers (CDD) development which are not part of the AUTOSAR specification yet.

We have become responsible of the whole software delivered to our internal customers. With this we have built-up our AUTOSAR know-how and must also continue to be a team of technical experts on the microcontroller topics.
AUTOSAR: A good revolution?

The standardization of the APIs has proven to be the key of strong commercial models, the latest example being Android. This is even better when it is done in a collaborative way like Linux.

In this scope the AUTOSAR methodology is a real breakthrough for the automotive market. It is the result of collaborative work and it defines a standard. With this, new business models can be envisaged bringing increased flexibility in our product development.

However, there are still some topics that are worth being noted here.

First, the AUTOSAR software architecture multiplies the layers and logically increases the resources needed by the system to work. Let’s take a simple example with the Memory stack. In an old school implementation, you would need one periodic function call, some blocks descriptors in Ram and in Rom, and probably an internal buffer to store the data that are being written. With the AUTOSAR model, the stack is divided in two main independent layers (NvM on the top and Fee/Ea in the middle) that both need to have their main function, their block configuration (Rom and Ram) and their internal buffer. Practically, we observe that the code size is multiplied by 4, whereas the Ram and Rom used by the configuration data is multiplied by 8 (Figure 9).

![Figure 9: Comparison of an AUTOSAR with a non-AUTOSAR platform](image)

It is, on the other hand, a really good thing that the standardization efforts are not wasted in the maintenance of too many releases. In the near future, all the AUTOSAR development will be based on the same specification, which is probably the one being prepared now.

Figure 10: Evolutions of the standard over the last 8 years

an AUTOSAR architecture shall not be considered without a significant improvement of the hardware capabilities.

However the trend from the silicium manufacturer is to always propose better devices for a similar price.

Secondly, the evolution of the standard, and its ability to improve through the feedback of the users, may show some limitations at this step of the development.

From the moment we entered the phase III of the AUTOSAR projects, the support of the oldest releases (2.1, 3.0 and 3.1) has been restricted to the diffusion of “List of Known Issues” (Figure 10). This allows focusing the effort on the main releases (4.0 and 3.2). However, we still have a lot of active projects based on these older releases. Concretely, it means that the feedback we give to the AUTOSAR community contributes to the evolution of the newer releases, which is not exactly what we need when we have to solve quickly some issues we have with an old release.
Conclusion

AUTOSAR has been a real challenge for us, but also a good opportunity of skill development within our teams.

As Basic Software providers, our main concern has always been to guarantee the quality of our deliveries and its relevance regarding the requirements of our customers.

We would not have managed successfully our transition to AUTOSAR without the work on the Stack organization, on the development and integration methodology, on the tooling and on the tests that are described in this paper.

The technical expertise we have capitalized up to now has been preserved and is needed to provide an efficient support to our customers. Our company specific know-how related to the hardware is more than ever alive and is used for the development of the Basic software IoHwAb and Complex Devices Drivers.

Today, we master completely the technology we deliver to our customers and are able to contribute to the AUTOSAR Consortium, without suffering from the initial inconveniences we found in the earlier developments.

References and sources

Figures 1, 10: www.autosar.org
Figures 2, 3, 4, 5, 6, 7, 8, 9: Continental Body and Security