**ABSTRACT**

The use of electronics in automobiles has increased over time. Today, the vehicle is a complex network of systems made by electronic control units, with different purpose. To manage this complex Software Systems Engineering task, Fiat Group Automobiles (FGA) formed a Software Factory Team within the Electronics Architecture Group. The team goal is to design, develop and test Software Component Module following AUTOSAR consortium guidelines. To manage all phases, SPICE standard level 2 is implemented.

The team is formed by software engineers, which design C-CODE and test engineers which develop test cases. There are two principal activities of testing: design, execute test case and analyze the results. If the results are negative, deviation must be reported and followed up.

All activities are Model Based Design centered: this allows to predict general behaviour of the system, accelerate design cycle, reuse of test cases and it permits greater flexibility. This technique also helps to guarantee a good interaction between software design and software integration, in order to find out errors and bugs due to the interaction between software components.

**INTRODUCTION**

The amount of software in vehicles grows exponentially and this increase brings various challenges: key competencies, processes, methods, tools, models, division of work, maintenance, and long term strategies.

Today premium cars include about 70 ECUs and up to 5 different communication buses. Up to 40 percent of the production costs of a car are due to electronics and software [1]. For hybrids, where the amount of software needed for engine control is greater than others, the cost of electronics, as a percent of vehicle costs, is closer to 45 percent. Within 10 years, some experts predict that the percentages relating to the cost of electronics are expected to rise to 50 percent for conventional vehicles and 80 percent for hybrids [2].

Also, safety requirements, high emissions reduction mandated by government regulations and an increase of comfort in a car will probably lead to an enhancement in terms of electronics contents. In this scenario carmakers are forced to re-defined all the lifecycle production code, from the definition of the requirements to the verification and integration in a ECU: Fiat Group Automobiles (FGA in the following) decides to change thinking and basic idea of software design following AUTOSAR - AUTOmotive Open System ARchitecture -standard that improves complexity management of integrated E/E architectures through increased reuse and exchangeability of SW modules.

Software Factory team is part of Electronic Architectures in FGA E&D department and it aims to design, develop and test Software Component Module (SWC in the following) observing AUTOSAR standard. In the article, there is a short description of general state of art in software production. Then an explanation of testing strategy in FGA - Software Factory is given. Finally a list of current FGA projects which uses this technology is showed.

**BACKGROUND**

Traditionally the car industry is vertically organized. In software engineering it is very different, more modular. The mechanical engineering made the various sub-systems in cars quite independent. As a result, suppliers could take over a considerable part of the engineering, the development, and also the production by a consequent outsourcing. Ideally, the parts of cars were produced by a chain of suppliers and more or less only assembled by the car manufacturer (called OEM
in the following). This could lead to outsourcing of a large amount of the engineering and production that optimized the cost and risk distribution. A car was considered as a kit of subparts that were assembled by the OEM [1].

With software, as a new innovating force driving, the situation has been changed radically. The car changes from an assembled device into an integrated system and unintentional feature interaction become issues. The costs of cars start to depend more and more by costs of software development. The behavior of cars becomes much more programmable: some properties, such as comfort or sportive handling are no longer determined only by the mechanics but also by the software/electronics.

This situation has led to systems with a huge size and lots of embedded software/hardware structure in cars. In current premium cars more than ten million lines of code are present and this number will probably increase in the next generation. This implies new competencies, especially in the development processes. In fact it is very important to be able to increase, as fast as possible, expertise in the software engineering domain in an Organization. All the processes are influenced by software engineering and this will be more and more common. As a result the complexity and spectrum of requirements for on board software is enormous [1].

An example is the new innovative electro-hydraulic system integrated in the FGA engines called Multiair. Processes and models of software engineering have influenced a lot of mechanical engineering design: the whole mechanical valve control is driven by a software/electronic strategy AEC - Adaptive Electronic Control.

**FUTURE DOMAIN**

The future development is driven by the following trends:

- new innovative or improved functionality
- quickly interchanging platforms and system infrastructures
- rapid increase in development cost
- request of higher quality and reliability
- shorter time-to-market

In this context AUTOSAR consortium plays a fundamental role.

**AN OVERVIEW ABOUT AUTOSAR**

The AUTomotive Open System ARchitecture (AUTOSAR) was founded as a development partnership in 2003. The growing complexity of software and electronics makes integration increasingly difficult, with negative impact on reliability. AUTOSAR consortium makes true a common objective between OEMs and Tier 1 suppliers: to create a basis for industry collaboration providing a platform which continues to encourage competition on innovative functions (see Fig. 1). In particular:

- Implementation and standardization of basic system functions as an OEM wide “Standard Core” solution
- Scalability to different vehicle and platform variants
- Transferability of functions in the network
- Integration of functional modules from multiple suppliers
- Consideration of availability and safety requirements
- Maintainability throughout the whole “Product Life Cycle”
- Software updates and upgrades over vehicle lifetime
- Confirmation of legal aspects (environmental aspects and safety requirements).
- Enforcement in comfort and entertainment functional domains needs
- Driver assistance and dynamic drive aspects that include detection and suppression of critical dynamic vehicle states and navigation in high density traffic surroundings

All these aspects can be summarize in three main groups:

- Architecture - Integration of all platforms for hardware independent software application
- Methodology - To achieve seamless configuration, seamless description formats and templates. Also to share standard
process life cycle for software stack and integration of application software level in ECU.

- Application interfaces - Definition of interfaces related to automotive application in order to create a standard for software application level.

The AUTOSAR standard will serve as a common platform where vehicle applications will be implemented and will also serve to minimize the current barriers between functional domains. It aims to enable mapping of functions and functional networks to different control nodes in the system, almost independently from the associated hardware (see Fig. 2).

Software organization is formed by:

- Basic Software, that contains modules such as the operating system services and communication,
- RTE layer, abstracts the SWC layer from any implementation details of the basic software and is responsible for SWC communication,
- Application layer contains the application function code organized in atomic independent SWCs. This permits to develop them independently of each others [3] [4].

In the application layer there are SWCs which implement standard or specific core function of every OEMs (see Fig. 3).

AUTOSAR concept supports Model Based Design technique that allows significant advantages in terms of better cost design and a significant reduction of development
Model-Based Design (MBD shortened) is a software development technique which uses models that can be simulated. In MBD a designer models functionalities, and tests them on a PC with real-time simulation. The real-time simulation enables to verify and validate the algorithm, by using codes generated from the model. Furthermore, auto code generation products enables to generate C/C++ program for embedded controller (microprocessor, DSP, etc.).

With Model-Based Design, engineers improve efficiency for the following reasons:

- Hardware prototype reduction and fail-safe verification by real-time simulation
- Efficient test by model verification
- Using a common design environment across among project teams
- Linking designs directly to requirements
- Integrating testing with design to continuously identify and correct errors
- Automatically generating embedded software code with coding time and error reduction
- Developing and reusing test suites
- Automatically generating documentation
- Achieving best time to market

In recent years model based design methodology has become the preferred method for designing, modeling and simulating complex dynamic systems [5] [6].

**SPICE**

In order to develop software component modules, Software Factory team has defined and implemented a process to manage all phases of work. This process has been implemented based on the reference process model called SPICE and it references to international standard ISO/IEC 15504.

SPICE (Software Process Improvement and Capability dEtermination) aims to evaluate organization process in order to measure the capability and the improvement attitude of its internal processes. Processes are evaluated in terms of organization, technique and quality [7].

For each of them, ISO/IEC 15504 defines a capability level (see Fig. 4) on the following scale:

<table>
<thead>
<tr>
<th>Level</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>Optimizing process</td>
</tr>
<tr>
<td>4</td>
<td>Predictable process</td>
</tr>
<tr>
<td>3</td>
<td>Established process</td>
</tr>
<tr>
<td>2</td>
<td>Managed process</td>
</tr>
<tr>
<td>1</td>
<td>Performed process</td>
</tr>
<tr>
<td>0</td>
<td>Incomplete process</td>
</tr>
</tbody>
</table>

![Figure 4. The process capability levels in SPICE](image)

Capability levels are estimate using the process attributes. The international standard has defined nine process attributes:

- 1.1 Process Performance
- 2.1 Performance Management
- 2.2 Work Product Management
- 3.1 Process Definition
- 3.1 Process Deployment
- 4.1 Process Measurement
- 4.2 Process Control
- 5.1 Process Innovation
- 5.2 Process Optimization.

Each process attribute is assessed on a four-point (N-P-L-F) rating scale:

- Not achieved (0 - 15%)
- Partially achieved (>15% - 50%)
- Largely achieved (>50% - 85%)
- Fully achieved (>85% - 100%).
The rating is based on evidence collected against the practice indicators, which demonstrate fulfillment of the process attribute.

All the activities in Software Factory team, fulfilled by Project Planning Configuration Management and SW Quality Assurance, have led the team to level 2 certification reached at the end of 2008. This evaluation had involved Engineering, Supporting and Management processes.

LIFECYCLE IN SOFTWARE FACTORY

Software Factory has structured its jobs following reference SPICE methodology (see Fig. 5):

• Requirements

The first important activity in software lifecycle is the analysis of functional specifications. It is owned by Software Engineer (SE in the following) and by Test Engineer (TE in the following): specifications are composed by functional documents defined from FGA and architectural constrains of software defined by Electronic Control Unit (ECU) suppliers, that aims to integrate the SWCs in the physical hardware. Requirements analysis is necessary in order to solve possible problems in terms of bad interpretation and also to fix issues of requirement lack/inconsistency.

• Architectures

When the requirements are exhaustive, that means understandable and complete, SE starts to plan the architecture of model; in AUTOSAR, application software is organized in independent units, called software components - SWC. SWC is an atomic piece of software that implements respective application function. At this level SE shared the SWC/RTE interfaces with ECU supplier writing the Software Component Configuration document (SWCC). After that, two parallel activities will start: the design and development of model by SE and the design of test cases by TE.

• Model Base Development

The following step is to develop functionalities according with the rules of Model-Based Design. This also enables users to visualize system model and subsystem design in a graphical and interactive environment. All models are developed in Matlab/Simulink/Stateflow. The models are hierarchical and the system is composed into functional units. The graphical environment enables to understand the design and the interactions of the subsystems more easily than text-based models.
Model is composed of two parts (see Fig. 6): a block named Logic, with all the functionality logics and a block named Boundary, that contains input stimulation.

- Model based testing

At this level the tested component is specified as a model. This executable model is used to check if the specification complies with the requirements. When the architecture is defined also model testing is performed. Model-in-the-loop (MIL) technique is used to verify the SWC: once Simulink model is available, TE can validate it using test case planned.

- Auto coding

After the validation, code can be generated from the model for implementation purposes. The code is automatically generated by TargetLink/dSPACE environment, which produces highly optimized C code from a Simulink model.

- Code Validation

The generated C code is tested and modified in the software-in-the-loop simulation system (SIL) to guarantee robustness of the software functions. In SIL the generated embedded code is tested in closed loop simulation using the same model. The differences between the functional Simulink models and the generated code are analyzed in order to evaluate the efficiency of the generated code.

MODEL BASED TESTING

Developing tests and the ability to maintain them are placed under severe pressure due to the short release cycles expected in software market and for the increased competition. These reasons have encouraged the involved teams to search for techniques which improve the traditional approach of handmade individual test cases.

Efforts have focused on developing methods and techniques to support model-based design and this goal has influenced activities in many ways.

Along with the growing functionality and the introduction of model-based development processes, the demands on quality assurance have also increased: in this view, a model based testing has been developed to validate model.

Testing activity is based on a block in Matlab/Simulink called signal builder (see Fig. 7) which permits to create a groups of signal sources and stimulate them into a model.

FGA has developed the idea of automatic test generation, designing a tool called EA_SWTest (Electronic Architecture Software Test) using Matlab Programming Language. The source code of EA_SWTest consists of 4 GUI (Graphical User Interface) that contains 116 classes with approximately 7000 lines of code. The lines are divided between different modules as follows:
• GUI (see Fig. 8), the main user interface;
• tool bars and menus;
• the main function, which starts the program execution.

Figure 8. EA_SWTest tool used for model based testing

Starting from SWCC, shared interfaces definition between Application Layer and RTE environment, TE designs sequences of test cases in an electronic sheet template.

In the following the principal activities to verify software component module are described (see Fig. 9).

• Definition of Test Case: first step is to classify requirements of software module. There are three different types of requirements:
  1. Nominal - Nominal represents the normal/expected behavior requested; test cases shall verified that SWC can cover this requirements in order to implement the functionality.
  2. Fault - Requirement is necessary to implement recovery strategy if a fault is detected.
  3. Misuse - Misuse verifies some particular conditions that can happens with low probability.

Starting from requirements, a chronological sequence of test case will be defined. Test case is a finite structure of inputs and expected outputs, written in a Excel sheet template. They will also include additional information that contains descriptions of execution conditions and applicable configurations. Every test cases are linked to functional requirements implemented (see chapter TEST COVERAGE).

Test engineers can use the Excel spreadsheet tool to write test sequence easily, with the help of macro writing in Visual Basic language. Some important features are:
  1. import RTE interfaces from SWCC document
  2. add new output in an existing test case
  3. automatically numbering of test case
  4. automatically compiling of the test report.

• Generation of signal builder from test case (see Fig. 10): when test case is defined, EA_SWTest generates two signal builders, one contains the timing sequence of the inputs applied to the model and the other one includes the expected output signals.
• Application of test case (signal builder): EA_SWTest stimulates the model with signal builder of inputs; at the end of the simulation, the real output, coming from model are registered into a new signal builder.

• Output verification and report of validation: EA_SWTest compares the expected results with the outputs coming from model, and registers the differences (see Fig. 11). When output from test case and output from model have been compared, EA_SWTest creates a signal builder with the differences.

If there are differences TE informs SE filling bugs report. After that, SE will provides the solution of the malfunctions (see Fig. 12). When differences are fixed, model-based testing process is complete and the model is ready for auto coding.

If a new requirement shall be implemented, the validation process includes also the unchanged requirements. In this case the unmodified subsystems are called no regression subsystems.

TEST COVERAGE

Time and money are commonly used criteria to determine whether to end the testing of a product or not. Unfortunately these criteria do not set any quality standard on the product. If parts of the system are not exercised at all, then the system is probably not tested enough.

Analysis of the coverage reports allows the verification team to modify the test sequence in order to improve specific tasks in the design [8]. The analysis of coverage reports guides and enhances the implementation of the test plan, so considerable effort is invested in finding ways to close the loop of coverage analysis and test generation. Our approach is able to develop this process using a technique to automate the coverage feedback analyzing test results.

EA_SWTest creates two products for every typologies of test case written (nominal, fault or misuse): excel sheet report and a graphical plot. The first document contains two tables. The
first one collects all test cases with the corresponding requirements implemented, and the second one collects the number of requirements with the relative percentage of usage (see Fig. 13). Starting from these data, statistical index and other information about test coverage such as the number of requirements which had been not verified and test case that covers all requirements;

Testing is potentially endless. It is impossible to test until all the defects are removed. A statistical index that gives information of how a certain SWC has been tested is the stress index: with this concept it is possible to establish the requirements which have not verified, ensuring high software quality.

The second document is a plot (see Fig. 14) that gives a graphical idea of coverage for every single requirement. The number of requirements exercised can be compared with the total number of requirements, and the percentage can be calculated, as shown in the following pie chart.

RESULTS

The new approach of model based testing offers considerable promise in cost reduction. Within validation process, the increasing of tests effectiveness will short the testing lifecycle. It also can be especially effective for systems that are changed frequently, because testers can quickly update test case and then rapidly regenerate updated signal builder.
Model based test automation discovers defects better than manual methods.

Before the introduction of automatic testing tool the same test took 20 minutes instead 1 minute to run: this means a reduction in term of validation time and software implementation cost (see Table 1).

**Table 1. manual testing among model based testing**

<table>
<thead>
<tr>
<th>Module</th>
<th>Features</th>
<th>Manual testing</th>
<th>Model based testing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Management of alternator</td>
<td>N° of test cases written</td>
<td>13</td>
<td>110</td>
</tr>
<tr>
<td></td>
<td>Time of execution of single test</td>
<td>15</td>
<td>1</td>
</tr>
<tr>
<td>Defrosting</td>
<td>N° of test cases written</td>
<td>21</td>
<td>79</td>
</tr>
<tr>
<td></td>
<td>Time of execution of single test</td>
<td>30</td>
<td>1'13&quot;</td>
</tr>
</tbody>
</table>

Software Factory has developed and verified the following SWCs:

- Defrosting
- Management of alternator
- Managing of external temperature
- Internal Lights
- External Lights
- Alfa DNA (Dynamic Normal All Weather)
- Start & Stop
- Wiping and Washing
- Radiofrequency Management
- Power door lock
- Management of Fuel level

Integrated in actual projects (vehicle in production and future cars):

- New 500
- New Fiorino
- Alfa MiTo
- Giulietta
- New Doblò
- Grande Punto (restyling)
- New Lancia Y
- New Panda

The introduction of model based testing automation began in 2007 and it has improved with time and job experiences. The following histogram (see Fig. 15) shows the growth of module standardized in vehicles from 2008 to 2010.
SUMMARY/CONCLUSIONS

The successful application of model based testing to different case studies proves the feasibility of the methodology. With automated tests, using MDB technique, it is possible to find bugs in a simulation environment on a PC. This ensures a lower defect rate than manual testing when code is integrated. Also it goes forward in testing process regarding quantity and quality.

With suitable test case tool, it is easy to generate test sequences that cover all the requirement specifications. One of the best results of MBD testing is the high degree of test cases reuse thanks to standardization and scalability concepts. In fact many testing problems have generic elements that easily could be reused in the creation of test cases. MBD technique allows test engineers to focus more time on unique aspects that are specific to an application, without waste effort on definition of generic test sequence. Test engineers should be able to reuse the full and rich set of tests that previously have been identified for similar situations. Improvement about writing test case and model development will increase. The results of these studies will be reported in future works.

REFERENCES


2. Charette, Robert N., This car runs on code, IEEE Spectrum magazine, February 2009

3. Simon Fürst AUTOSAR - An open standardized software architecture for the automotive industry. 1st AUTOSAR Open Conference & 8th AUTOSAR Premium Member Conference October 23rd, 2008, Cobo Center, Detroit, MI, USA

4. Official web www.dspaceinc.com


7. Official web site http://www.isospice.com

Figure 14. Coverage of each requirements regarding Start&Stop functionality

CONTACT INFORMATION

Maria Stella Cavallaro  
Engineering & Design - E/E Architecture & Integration - EA Software Factory  
Fiat Group Automobiles  
Corso Settembrini, 40 - First floor C5, 10135 Torino  
tel. +39.011.0038858  
fax. +39.011.0033609  
mariastella.cavallaro@fiat.com

Antonio Marino  
Engineering & Design - E/E Architecture & Integration - EA Software Factory  
Fiat Group Automobiles  
Corso Settembrini, 40 - First floor C5, 10135 Torino  
tel. +39.011.0037403  
fax. +39.011.0033609  
antonio.marino1@fiat.com

Figure 15. Functionalities developed and standardized from 2008 to 2010