The MeMVaTEx methodology: from requirements to models in automotive application design

A. Albinet, S. Begoc, J.-L. Boulanger, O. Casse, I. Dal, H. Dubois, F. Lakhal, D. Louar, M.-A. Peraldi-Frati, Y. Sorel, et al.

The MeMVaTEx methodology: from requirements to models in automotive application design

A. Albinet[1], S. Begoc[2], J.-L. Boulanger[3], O. Casse[2], I. Dal[2], H. Dubois[4], F. Lakhal[5], D. Louar[5], M.-A. Peraldi-Frati[6], Y.Sorel[5], Q.-D. Van[3]

1: Siemens VDO, BP 1149, 31036 Toulouse, France.
2: Monditech, 1 Place C. De Gaulle, 78180 Montigny-Le-Bretonneux, France.
3: UTC/HEUDIASYC UMR 6599, 60205 Compiègne, France.
4: CEA LIST, Boîte 94, 91191 Gif-sur-Yvette Cedex, France.
5: INRIA, Rocquencourt, B.P. 105, 78153 Le Chesnay Cedex, France.
6: I3S, UNSA, CNRS/INRIA, B.P. 121, 06903 Sophia-Antipolis, France.

Abstract: This paper presents a model-based methodology for requirements expression, traceability and verification. The methodology relies on the EAST-ADL2 framework and two of the UML2 profiles: MARTE for real-time embedded systems and SysML for system requirements modelling. The methodology defines the different models used at each abstraction level of the process. The results are a requirement model and a solution model which is related to the requirements. Verification and validation models and techniques are connected to these models. An automotive case study, namely a knock controller, illustrates the proposed methodology. The tools used in the process are also presented.

Keywords: requirements, methodology, automotive, model-driven engineering.

1. Introduction

This paper presents current results of a work achieved within the framework of the MeMVaTEx project[1] [1]. This project is intended to provide a methodology for requirements traceability using a model driven engineering (MDE) approach in order to design automotive embedded systems. Sound methodologies are necessary to tackle the complexity and the quality concerns. As the MeMVaTEx thematic is complementary to others French or European projects, such as ATESST [2] and TIMMO[2], we have collaborations and we share experiences on real-time design models approach with their members.

Requirements expression and management is a very important challenge in a MDE approach. The MeMVaTEx project focuses on two main objectives. The first one consists in enriching the requirement expression in order to take into account multiform and multi-users requirements. The second important objective is to reduce the gap between the specifications and the solution model, and for this purpose we propose mechanisms to improve traceability. The first step is the validation of the consistency between requirements model and solution models.

This paper presents a model-based methodology for expressing requirements and traceability mechanisms during the modelling process, before finally considering verification & validation (V&V). Our approach relies on different standards. Firstly, the EAST-ADL2 [2] (*Electronic Architecture & Software Tools – Architecture Description Language*), which is defined for vehicle embedded electronic systems development. Moreover, two of the UML2 profiles are considered: MARTE (*Modelling and Analysis of Real-Time Embedded systems*) mainly for timing properties expression [12], and SysML (*System Modelling Language*) for requirements modelling and traceability [9].

From the EAST-ADL2 framework, we adopt a decomposition of the design process into abstraction levels. For each level, we built separately requirement models and solution models. The relationships between elements of those models are expressed by using traceability mechanisms of SysML. The real-time aspects and non functional constraints are modelled within the UML MARTE profile. V&V techniques can then be connected to these models to express the satisfaction of the requirements by the proposed solution.

All these aspects will be developed and illustrated in the paper on the knock control application. This example is a good illustration of electronic embedded systems: multiform requirements, data flow and control flow behaviours, real-time aspects (temporal constraints, deadlines, limited resources).

[1] This work has been performed in the context of the MeMVaTEx project (*http://www.memvatex.org*) of the System@tic Paris Region Cluster. This project is partially funded by the French Research Agency (ANR) in the "Réseau National des Technologies Logicielles" support.

[2] See the TIMMO project web page: *https://www.timmo.org*

This paper is organized as follows: first, we give an overview of the methodology in the second section. The section 3 presents the different model elements for requirement expression (MeMVaTExRequirements profile). The section 4 is dedicated to the solution models with a special focus on temporal behaviour expression hardware and allocation aspects. The verification and validation means are presented in the section 5. The section 6 focuses on all traceability concerns. After presenting the concepts of the methodology, we demonstrate it on the case study. Finally, since MeMVaTEx aims to provide a tooled methodology, we present the tool architecture that we use in order to express and trace the requirements, as well as to create the solution models.

2. The triptych for an EAST-ADL2 based methodology

EAST-ADL was developed in the context of the EAST-EEA[3] European project [5] and the EAST-ADL2 version proposed by the ATESST project is now under finalization and validation [3]. It provides a unified notation for all the actors of a car development (car-maker, suppliers…). EAST-ADL2 allows the decomposition and the modelling of an electronics system through five abstraction levels (*Feature*, *Analysis*, *Design*, *Implementation,* and *Operational*). These levels and the corresponding model elements provide a separation of concerns that is the basis of the structure for the different models in the MeMVaTEx methodology.

Our objective is to help the designer in managing requirements during the system development. In order to deal with requirements, we first need to express them and then, to trace them all along the modelling process. Usually, two ways of considering requirements are followed:

- Either the requirements are managed via a requirement tool, independently (considering the used formalisms) of the modelling design. These approaches are based on requirements tools such as Reqtify[4] or its open source version TopCased-TRAMWAY[5]. Links between requirements and models are some kind of annotations that help in following the requirements in the models. The advantage of this approach lies in the fact that requirement management can be done on them (which ones are validated, which ones are decomposed, etc.).
- Or requirements are directly attached to the solution models but without real specific management (requirements are considered as informal comments). The advantage of this approach lies in the fact that we can keep the same modelling formalism. This simplifies the traceability management of the requirements between them and the proposed solutions.

In order to avoid the management of the requirements, we have chosen another approach, based on models, that takes advantages of both previous ones. This approach directly includes the requirements in the modelling process, so that requirements can directly be put in the models, and connected to the developed solution for an easier traceability and management.

In order to strongly isolate requirements management from solution management, we decided to have a specific structure that clearly separates the different concerns: the requirements on the one hand, and the solution on the other. This issue is crucial because similar requirements can lead to different solutions. Both are based on the same modelling formalism and thus, traceability is strongly facilitated. These three types of concerns constitute what we call a triptych composed by:

- The requirement models: a repository for requirements. We also consider the links between requirements for two succeeding levels, the links to solution model elements that satisfy the requirements, and the links to the V&V.
- The solution models: the developed models that should answer to the related requirements. We can here express the functional and non functional modelling with a special focus on real-time constraints modelling.
- The V&V means used to validate solution models with respect to the related requirements.

These three aspects are developed in the next sections.

3. Requirement models

In this section, we present the structure of a requirement definition.

As presented in the MeMVaTEx glossary [8], and as defined in the EIA 632 norm [6], a requirement is «*Something that governs what, how well, and under what conditions a product will achieve a given purpose*». It is also defined as followed in the IEEE 1233a [7] standard:

> (A) A condition or capability needed by a user to solve a problem or achieve an objective.
> B) A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed document.
> (C) A documented representation of a condition or capability as in definition (A) or (B).

[3] See EAST-EEA web page: *http:// www.east-eea.net*
[4] See Reqtify web page: *http://www.geensys.com*
[5] See TRAMWAY web page: *http://gforge.enseeiht.fr*

(D) The necessity that a system has a particular feature.

In order to define requirements in the requirement models, we base our approach on the SysML profile that defines how to express requirements with specific Requirement Diagrams. The SysML profile allows the designer to consider requirements as first class concepts in UML for system level design, and to deal with traceability concerns since relations between requirements, and requirements or model elements, are also defined in SysML. We only consider in this section the requirement part of SyML. Figure 1 shows that a `Requirement` in SysML is composed by two tagged values: an identifier (`Id`) and the description of the requirement (`Text`). This stereotype is useful for requirement annotation of the diagram.

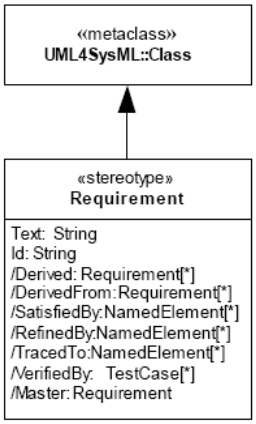

Figure 1 - The `Requirement` stereotype in SysML

But this definition is not enough for requirement engineering since requirements are not sufficiently detailed to support the analyst when he validates requirements with respect to the developed solution. Indeed, this profile does not offer the possibilities to follow requirements (if they are fulfilled), and to relate them to the verification process. These reasons have led us to define our specific profile for the satisfaction of ours needs. The definition of the `MeMVaTEx Requirement` stereotype is presented in Figure 3.

This stereotype is associated to the SysML requirement stereotype in order to get the relations defined in SysML for requirements to model elements. The SysML `Requirement` stereotype is thus not re-usable since the related requirements are referenced as attributes. This profile replaces the `Requirement` defined in SysML with the following tagged values:

- `Title`: this information is a unique identifier of the requirement. It is structured in order to avoid any kind of identifier as proposed in SysML. In our case, a `Title` is composed as follows:

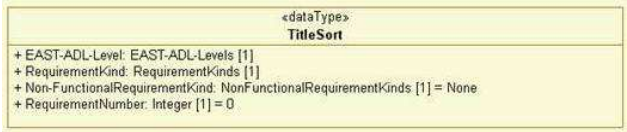

Figure 2 - Title of a `MeMVaTExRequirement`

where the `EAST-ADL-Level` is specified, as well as the `RequirementKind` (functional or non functional), the `Non-FunctionalRequirement Kind` (such as Safety, Maintainability, Variability, Performance and so on…), and the `RequirementNumber` to uniquely distinguish requirements of the same category.

- `Description`: to give the full description of the requirement if this is a textual description.
- `Verifiable`: a boolean indication that specifies if the requirement should be verifiable as such, or if a refinement of this requirement should be considered.
- `Priority`: used to differentiate a mandatory requirement from an optional one.
- `VerificationType`: to memorize or precise the verification technique that is used to check the requirement: `Test`, `FormalProof`, etc.
- `Author`: the author of the requirement.
- `SourceReference`: the name of the initial document from which the requirement is taken.
- `Status`: to specified if the requirement is `Analysed`, `Rejected`, `ToBeAnalysed`.
- `Justification`: the reason why this requirement is here (important in case of decomposition or in case of justified choice).
- `DocumentType`: the type of the source document: a report, a meeting, a drawing, etc…
- `ASIL-Level`: the ASIL-Level (from A to D) when this information is known.

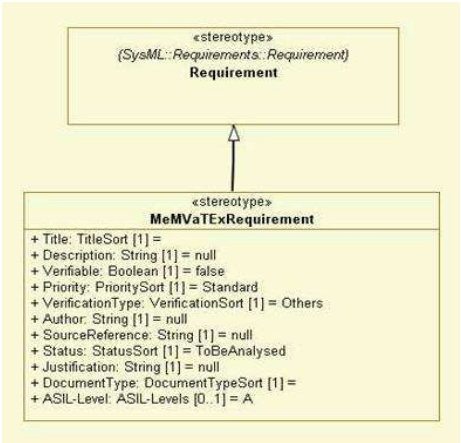

Figure 3 - The `MeMVaTExRequirement` stereotype

The usage of this stereotype will be illustrated later on.

4. Solution models

In automotive, and more generally in the embedded real-time system domain, the trend is to switch from programming to composition for the management of the overall engineering information. The objectives are to control the complexity, to increase the quality of the software and to reduce the development cost of systems [10].
The use of models, even if they not perfectly fit the specifications, provides the only known solution to abstract and reduce details of an application. By the concepts of "views", the model-based methodology focuses on a domain specific modelling and provides a way to analyze the model and refine it.

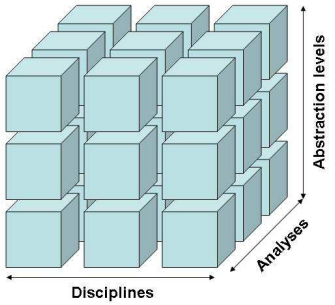

Figure 4 - The solution model space

The Figure 4 shows the three dimensions of the space of solution models. The *abstraction levels* represent the knowledge level of the application, in the case of a development from scratch. They are mandatory to handle efficient development and manage complexity of automotive embedded systems. The *disciplines* allow for observing our application with respect to different perspectives. They correspond to the engineering skills involved such as the system description, function design, software development, hardware and software architecture. With such decomposition, the *analysis* only considers a specific domain view of the system, at a particular level of abstraction. This "ideal" decomposition of a system raises the tricky problem of the relationship between the levels.
We adopt a similar approach of decomposition in the project. The *abstraction levels* are those of EAST-ADL2. The *disciplines* are the ones of vehicle feature description, control/command modelling, software design, architecture description and allocation. In the next subsections, we present the different model elements used for the solution model at the analysis and design levels, and we illustrate the software design of functions and the allocation activity of these functions onto a hardware architecture.

4.1. Temporal and Functional Modelling

At the analysis and design levels of the EAST-ADL2 process, the functional modelling is based on the metaclass `ADLFunctionType` which extends SysML blocks. Our methodology allows making a clear distinction between the design at the higher abstraction levels (Analysis and Design) and the execution at the lower levels (Implementation and Operational: AUTOSAR[6] (*AUTomotive Open System Architecture)*. All `ADLFunction Type` have the capacity to describe the internal behaviour of a function, possibly with hierarchy. These elements are connected and communicate through ports (which extend SysML ports) and specific connectors. At the Implementation Level, the application software is modelled as an atomic unit without any hierarchy called `AtomicSoftware Component` [2].

EAST-ADL2 does not support the expression of temporal constraints associated with `ADLFunction Type` or `AtomicSoftware Component`. The only element that deals with time is the `ADL requirements`. They consider the temporal needs such as jitter, period, max and min duration. `ADLrequirements` can express temporal requirements by the way of `EndtoEndDelays` of `ADLFunctionType` or `EndToEndDelay` between `ADLPortFlow`. The unit used for time expression is the classical clock (the chronometric one).
We extend EAST-ADL2 by the time models of MARTE for expressing temporal constraints that can be either linked to a chronometric time but also with a logical time. Examples of logical time are the camshaft and crankshaft angular positions in an engine. The period of these logical clocks depends on the engine rotation speed. The `clockconstraints` of MARTE is a way to express relations between clocks.
A `clock` is perfectly defined in MARTE. A clock has a `clockType` with different properties (nature, resolution, max and min offset) and a `unit` (logical or chronometric).
In EAST-ADL2, the structure of the application is described hierarchically using `ADLFunctionType/ Prototype`. `ADLbehaviour` is a property of an `ADLFunctionPrototype`.
The behaviour of `ADLFunction` is given by `runnableEntity`. A `runnableEntity` is an UML `CallBehaviourAction` on which we apply with the `timedProcessing` stereotype of MARTE. By this way, `timedEvent` characterizes the start and stop of a `runnableEntity`. A `timedEvent` is linked to a `clock` whose type can be `chronometric` or `logical`. The `duration` property of a `TimedProcessing` is a `TimeValueSpecification` whose expression may combine both chronometric and logical clocks, intervals and instants. CVSL[7] is the language supporting such expressions.

[6] See AUTOSAR web page : *http://www.autosar.org*
[7] CVSL : Clocked Value Specification Language

4.2. Hardware architecture modelling

In EAST-ADL2, the hardware architecture may be modelled from the Design Level. At this level, the hardware architecture is described at a high level of abstraction and can be used for simulation. This level factorizes general elements of the hardware architecture. It contains ECUs, power supply, sensor and actuators, their connectors and ports. In EAST-ADL2 the hardware elements extends SysML blocks notion, that are usually used for describing the hardware architecture of embedded systems. The details of theses elements are not given at this level.

In the EAST-ADL2 approach, the refinement of the hardware architecture is performed in the next level, i.e. Implementation Level. This level references the AUTOSAR profile. Indeed, all hardware architecture elements at the Implementation Level are factorized through the abstract metaclass `HW_Element` from the AUTOSAR profile. The hardware elements described at the Design level are expected at the Implementation Level with more details or properties, like for example the memories size and the number of processors in the microcontroller of an ECU. Moreover, the model brings more information about the hardware architecture by adding new hardware elements like communication between ECUs, specific devices (timers, DAC…), peripherals, I/O (analog, digital), etc.

4.3. Allocation Modelling

Finally, in order to implement an application onto a hardware architecture, it is necessary to be able to associate a particular application element to a particular hardware element. In the EAST-ADL2 language, allocations constraints corresponding to requirements inherited from the SysML requirements can be expressed from the Design Level. Indeed, there is a specific requirement called `AllocationConstraint` to express an allocation between an `ADLFunction`, or an AUTOSAR element (application part) and an ECU. At this level, it is only a choice of allocation and not an actual allocation. Actually, allocations are performed at the Implementation Level by using AUTOSAR features.

5. Link to the V&V methods

The V&V activities concern 2 aspects:

- Verification of the realization ("*Do we build the product right?*"). It is the analysis of the works that have been done, generally document analysis, code inspection, unit and integration testing.
- Validation of the application ("*Do we build the right product?*"), this is a test phase whose objective is to show that intended services are fulfilled. This test phase is realized on the product.

The next figure shows V&V activities in a standard V cycle development process.

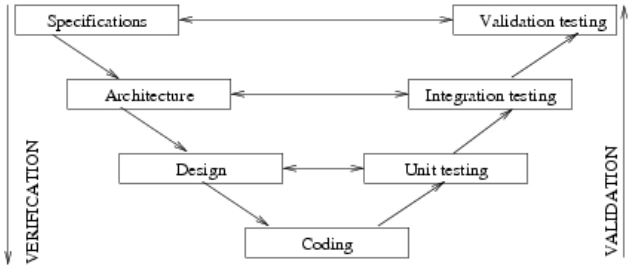

Figure 5 – V&V activities in a V-cycle development process.

In the development process, the V&V is characterized by activities that are performed by a specific V&V team, independently of the activities performed by the realization team. In order to model verification activities, we propose the modelling of different activities by "use cases". These "use cases" show the concerning persons, and necessary elements (procedure, document, etc.). The modelling is independent of the project but reflects the enterprise process of the company.

Verification activities need to be linked to requirements. It implies that:

- Every requirement is analyzed in the design phase. It implies the verification of the traceability and the justification of that verification.
- Every requirement is taken into account in the design. It concerns the `satisfy` links that show how requirements are realized by the elements of the solution models.
- Every requirement is taken into account in the verification. It concerns the `verify` links that show how requirements can be verified by test cases.

In SysML, a test case is intended to be used as a general mechanism to represent any of the standard verification methods for inspection, analysis, demonstration, or test. SysML has the capability for representing test cases and attaching them to their related requirements or use cases. A test case can be an operation or a behavioural model (Interaction, State Machine, Sequence or Activity Diagram).

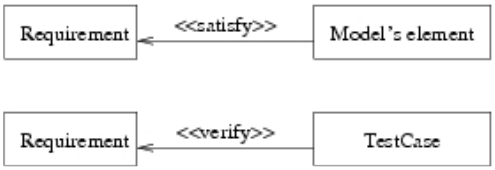

Figure 6 – Verification activities for Requirements

Tests can be executed in the context of a test scenario where test objects, their interactions, inputs and expected outputs are detailed. Then, actual outputs are confronted to expected outputs, giving the test verdict. Four test verdict possibilities are defined: pass, fail, inconclusive, and error. More descriptions on test activity can be found in [13].

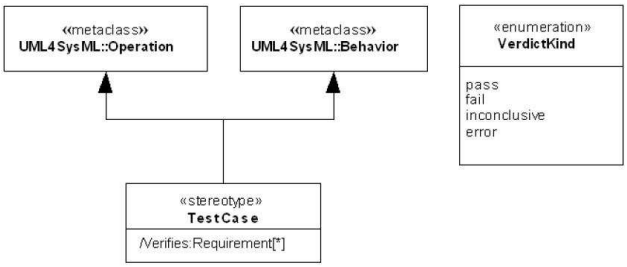

Figure 7 – SysML TestCase streretoypes.

The model-based design has several advantages for test and verification. These advantages are now supported by powerful tools that offer an environment for executable specifications, dynamic behaviour analysis, and algorithm design. Performing V&V early at the design level avoids costly prototypes. And correcting errors early at the Design Level is more effective than at the Implementation Level.

6. Traceability management

Once the specificities of requirements models, solution models and V&V means are presented, explaining traceability management implies to describe traceability mechanisms that are used for:

1. Relating requirements of the same abstraction level.
2. Relating requirements through successive abstraction levels.
3. Relating requirements to other elements from solution models or V&V means.

The definition of this traceability into the MeMVaTEx methodology is important since this usage will guarantee that requirements management will be possible also in the modelling structure of the system, and not only outside the modelling process. This will help the analyst in building a solution that checks all the requirements. This approach is different of the one that uses a specific requirement management tool for tracing requirements. The main benefit of our approach is that the modeller of the solution system can follow all the input requirements directly in the model, and not by using an external tool that adds another annotation complexity to the modelling despite an efficient management of the requirements.

Traceability links used in MeMVaTEx are those proposed by SysML, but they concern `MeMVaTEx Requirement` elements and not `Requirement` elements from SysML. They are presented in Figure 8.

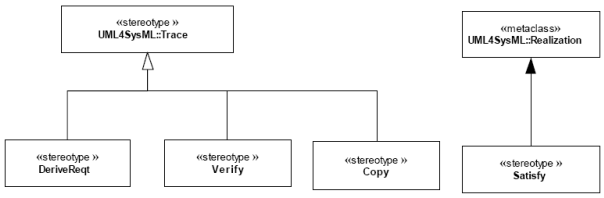

Figure 8 - Requirement streretoypes in SysML

These SysML relations are not useful for every usage, and our methodology clarifies the usage of each traceability link.

In a same EAST-ADL2 level (1), requirements can only be decomposed or refined. We thus use:

- The `DeriveReqt` dependency relationship between requirements for a requirement A (the client) refined into a requirement B (the supplier).
- And the `requirement containment` relationship for the decomposition of a parent requirement into several ones.

For considering requirements of different levels (2), we use the previous relationships and also the `copy` dependency relationship. This last one is related to requirements that appear in a level and that are unchanged when considering the next EAST-ADL2 level.

For relating requirements to other elements (3), the traceability links that are useful in this case are the followings:

- The `satisfy` dependency relationship that relates a requirement and a model element that fulfils the requirement
- The `verify` relationship between a requirement and a test case that can determine whether a system fulfils the requirement. This link was detailed in the previous V&V section.

7. Illustration on an automotive case-study

In this section, after a short presentation of the knock control application, we illustrate the proposed methodology by some elements of the case-study.

In a four stroke engine, the knock phenomenon is a self ignition that borns in the combustion chamber due to high pressure and temperature. When it occurs, this abnormal ignition generates a shockwave that disturbs the combustion, and have a negative impact on the engine lifetime, the comfort, the consumption and the torque (see Figure 9). The knock control consists in the noise estimation (capture, acquisition and filtering of knock samples signal) and the correction by calculating the advance of the ignition angle.

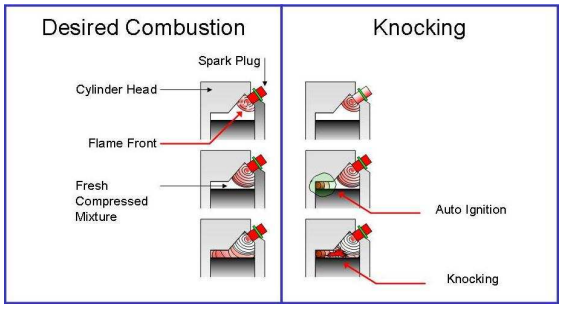

Figure 9 – The knock phenomenon

7.1 Requirement models

For the case study, we define requirements models that are based on the `MeMVaTExRequirement` profile presented in the previous section. An example is given in Figure 10.

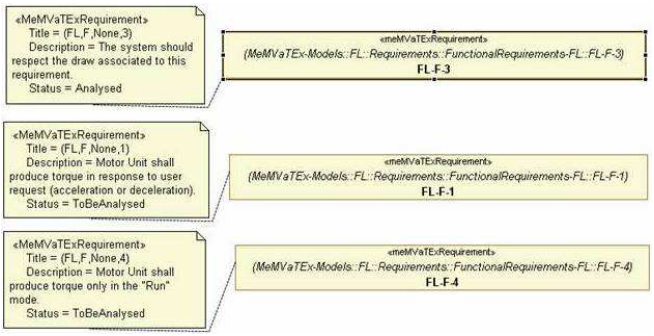

Figure 10 - Some Functional Requirements from FL EAST-ADL2 level

In this example, we show three functional requirements (UML elements in the right hand side of the figure) of the first EAST-ADL2 level: the Feature Level (FL). These requirements are stereotyped by the `MeMVaTExRequirement` stereotype, and three properties are here displayed for each one: the `Title`, the `Description` and the `Status`.

7.2 Solution models

Modelling Acquisition with EAST_ADL2 and MARTE

We illustrate the concurrent use of EAST-ADL2 and MARTE to express the structure and the behaviour of an application. The Figure 11 is an EAST-ADL2 diagram representing the structure of the acquisition. The execution of acquisition is triggered by events whose occurrences are linked to an angular time base and a chronometric time base. To express temporal relations and constraints on behaviours, we create two clocks types in the model, namely the `angularClock` and the `IdealClock`, and we instantiate three clocks (℃AM, ℃RK, `IdealClock`) from these types.

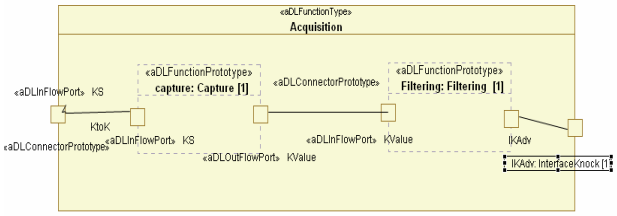

Figure 11 - ADL FunctionType of Aquisition

The behaviour of the acquisition is expressed through an activity diagram stereotyped by `TimedProcessing`. The Figure 11 shows the temporal properties applied to the activity. The duration is the `Min` function between two different clocks. The start and stop of the activity is modelled by `timedEvents` (`TE_ITDC` and `TE_KWE`).

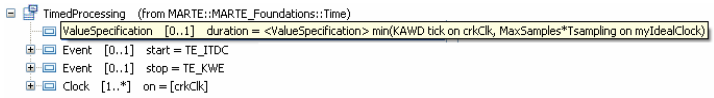

Figure 12- TimeValueSpecification for Acquisition

The temporal characterisation of `ADLFunction` makes it possible to construct timing chains on `ADLFunctions` and their connectors and permits two kinds of verifications. The obtained results are linked to the temporal requirements by a requirement relation. The allocation phase described in the next section, must takes into account these constraints in order to make the best association between functions and hardware components.

Modelling the allocation of functions onto the hardware components

In the MeMVaTEx context, several requirements concerning the architecture must be satisfied. The following example illustrates the link between a requirement and a model solution in our case-study. In this example at the Design Level an allocation requirement expresses that the acquisition function shall be allocated to the ECU. Thus, in order to satisfy this requirement, on the one hand we use the hardware architecture diagram that describes the architecture with an ECU connected to a sensor and an actuator, and on the other hand we use the acquisition function. To perform the allocation, we use the `AllocationConstraint` stereotype by creating a reference link towards the allocated element here the acquisition function and another reference links towards the hardware element here the ECU. Some tagged values may be added to complete the `AllocationConstraint` stereotype. For example, it is possible to specify some aspect about requirement traceabilities. At the Design Level, the Figure 13 represents the hardware architecture connected to a partial application software of the

case-study, and the Figure 14 represents the example of allocation.

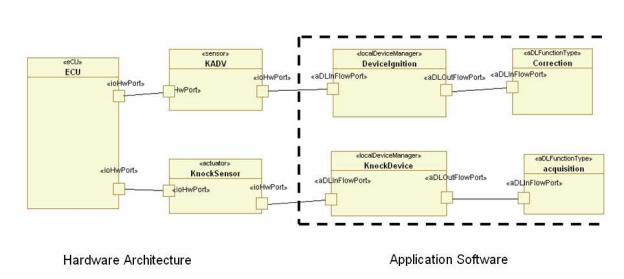

Figure 13 - Hardware architecture at the Design Level

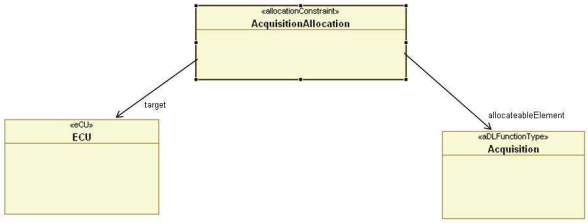

Figure 14 - Allocation at the Design Level

7.3 V&V means

The Figure 15 illustrates a requirement diagram realized with requirements from the knock case study. Requirements are classified by EAST-ADL2 levels. In the diagram, a traceability link from Feature Level to Design Level is shown: AL-F-12 is a functional requirement at the Analyse Level. It is derived from the requirement VL-F-9 at Feature Level, and then refined to DL-F-7 at Design Level.
The three requirements are respectively satisfied by `Knock_Correction`, `Engine_Control`, and `Threshold_Calculation` blocks; each block is represented by a Block Definition Diagram.

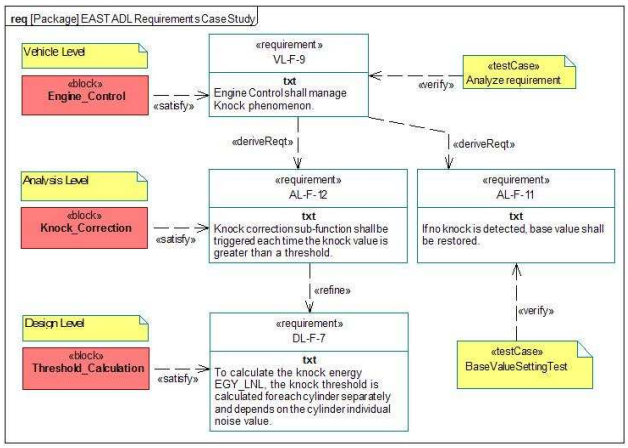

Figure 15 – A requirement diagram and its traceability.

In this example, a test case is created to verify the requirement AL-F-11: "*If no knock is detected, base value shall be restored*". The test case, described by a simple activity diagram in [13], compares actual values, and pre-defined values then, returns a verdict. A sequence diagram may be sketched out to complete the sequence of actions to be realized.

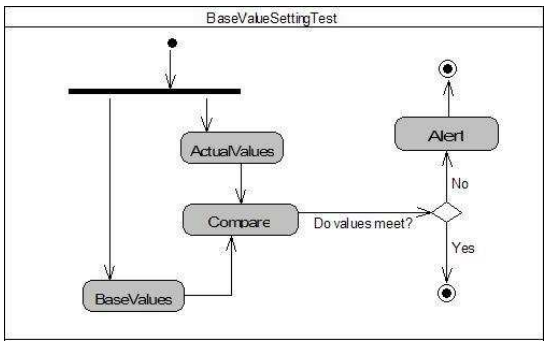

Figure 16 – A test case realized by an activity diagram.

A requirement at an upper level may also be verified by a document or code analysis. The analysis verifies if the requirement is in the right place, responds to a need, or conforms to the specification book. In the case study, "*Engine Control shall manage Knock phenomenon*" at Feature Level is such a requirement. The conformity of requirements is analyzed by independent analysts and system engineers. We represent this verification by a "use case" diagram in the Figure 17.

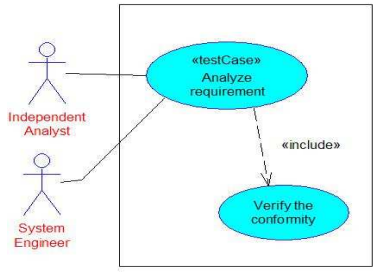

Figure 17 – A use case verifying VL-F-9 requirement.

7.4 Traceability links for requirements

In this section, we illustrate different traceability links defined in the section 6. We illustrate some of the identified traceability links.
For traceability link in the same requirement model, we illustrate in the Figure 18 the `requirement containment` relationship for the decomposition of a parent requirement into several ones.

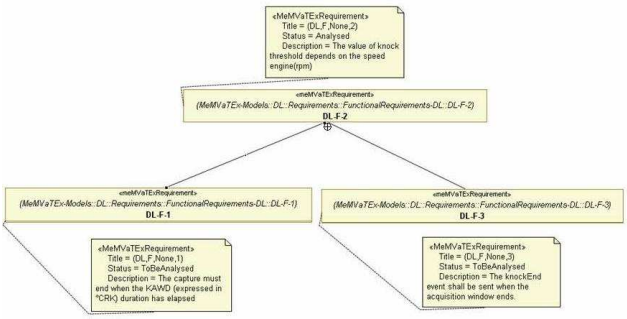

Figure 18 - Traceability links in the same requirement level.

In this figure, three functional requirements from the Design Level (DL) are defined (DL-F-1, DL-F-2 and DL-F-3), and the DL-F-2 requirement is decomposed into the DL-F-1 and DL-F-3 ones. This traceability links are present in the requirement definition part of the requirement models for each level.

In the Figure 19, we illustrate the traceability links between requirements in two successive levels. The illustrated links is the `copy` one which concerns non functional variability requirements. This is done in a specific diagram because this link relates requirements from different levels.

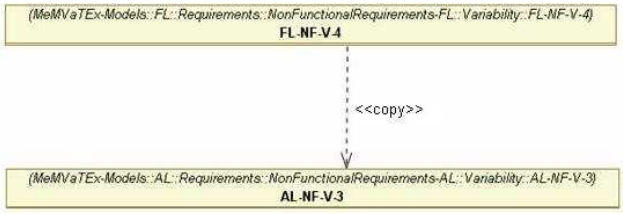

Figure 19 - Traceability links between requirement from levels FL to AL

The FL-NF-V-4 requirement in EAST-ADL2 FL (Feature Level) is copied in the EAST-ADL2 AL (Analysis Level). The names of the requirement are different since this corresponds to distinct requirements: one for each level. But, as the variability has to be mentioned in the early FL level and since this variability can only be managed in the solution in the AL level, we have to copy this requirement from the FL level to the AL one.

8. Tool architecture

Our methodology has been applied to the case study using tools which are presented in this section. We define a consistent tooled methodology for requirement capture and modelling, solution modelling, validation result feedback integration in a consistency management.
A list of currently deployed tools for industrial projects in embedded SW design, described in [10] was used to identify State-of-the-art file formats for interoperability and interaction: RIF (*Requirements Interchange Format*) [15] & XRI (*Extensible Resource Identifiers*) [16] for requirements, XMI (*XML Metadata Interchange*) [14] for profiles and models exchanges.

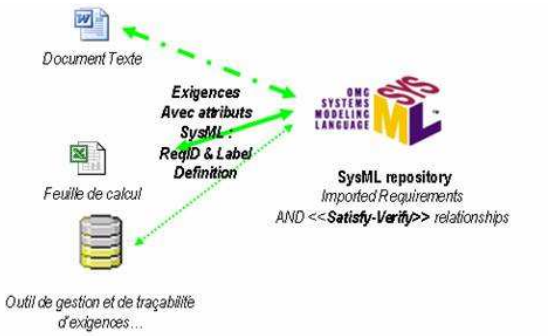

Figure 20 - Generic Approach

Then, we chose tools for supporting those files interfacing as smoothly as possible, able to support our methodology.

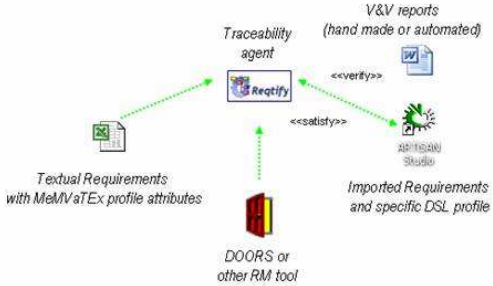

Figure 21 - MeMVaTEx Approach

One key feature of our methodology is the need to aggregate several profiles: MARTE, EAST-ADL2, AUTOSAR and SysML. We decided to develop our own UML dedicated profile, called RPM (standing for Requirement Profile of MeMVaTEx) as the DSL (Domain Specific Language) support, importing a

subset of needed stereotypes from profiles mentioned earlier.
The used tools for the methodology are:

- For requirements writing: MS Word, MS Excel or any tool allowing the text inputs, as they are the state of practice in automotive domain.
- For requirements traceability: Reqtify or its open source version TopCased-TRAMWAY. This tool allows almost any link with UML/SysML modelling tools, text editors, spreadsheets and testing tools. It will behave as the traceability gateway between requirements, models and V&V results. Recently, Geensys has announced a RIF support which strengthens us in this choice.
- For EAST-ADL2 and UML modelling: ARTiSAN Studio[8] was chosen for the creation of the requirement and solution models. Indeed, it offered the most advanced SysML support when the project has started and has now the most complete RPM support and a powerful API to connect other tools. Furthermore, ARTiSAN Studio provides full UML2.1 support needed for MARTE profile.
- For information release, documentation and reports: MS Word or similar tool.

9. Conclusion & perspectives

We presented our methodology for requirements traceability in the field of automotive applications. Following the EAST-ADL2 abstraction levels, the methodology is mainly based on a triptych composed of the requirement models issued from the requirement expressions, the solution models which answers to these requirements by actually implementing functional specifications onto actual hardware while satisfying non-functional specifications, and finally the V&V process based on links established between the requirements and solution models. Then, we illustrated the methodology through some parts of a knock controller: a realistic case study of the automotive domain. Finally, we presented the tools that are used in the methodology.

The next development of the methodology concerns the integration of “heterogeneous” models in order to improve the V&V process in the methodology.

10. References

[1]. A. Albinet, J.-L. Boulanger, H. Dubois, M.-A. Peraldi-Frati, Y. Sorel, Q.-D. Van. *Model-based methodology for requirements traceability in embedded systems*, 3rd ECMDA workshop on traceability, June 07, Haifa, Israel.

[2]. *ATESST project - The Modelling Approach in ATESST – Overview*. *Overview of the EAST-ADL2 (D.3.1)*. And *Report on behaviour modelling with the EAST-ADL2 (*D.3.2, *version 1.0)*. Reports of the Advancing Traffic Efficiency and Safety thought Software Technology (ATESST) project. *http://www.atesst.org*.Final versions, 2007.

[3]. P. Cuenot, P. Frey, R. Johansson, H. Lonn, M.-O. Reiser, D. Servat, R. Tavakoli Kolagari and D.J. Chen. *Developing Automotive Products Using the EAST-ADL2, an AUTOSAR Compliant Architecture Description Language*. 4th European Congress ERTS Embedded Real Time Software. Toulouse, France, January 2008.

[4]. P. Cuenot, D.J. Chen, S. Gerard, H. Lonn and M.-O. Reiser, D. Servat, C.-J. Sjostedt, R. Tavakoli Kolagari, M. Torngren and M. Weber. *Managing Complexity of Automotive Electronics Using the EAST-ADL*. Proceedings of the 12th IEEE International Conference on Engineering Complex Computer Systems (ICECCS 2007), Washington, DC, USA, 2007.

[5]. V. Debruyne, F. Simonot-Lion and Y. Trinquet. *EAST-ADL an Architecture Description Language, Validation and Verification Aspects.* In IPIP World Computer Congress - Workshop on Architecture Description Language, Toulouse, France, August 27, 2004.

[6]. Government Electronics & Information Technology Association EIA norm for Processes for Engineering a system. EIA Standard 632. April 1998.

[7]. Institute of Electrical and Electronics Engineers. *IEEE Guide for Developing System Requirements Specifications (including IEEE 1233a)*. IEEE, New York, 1998. IEEE Standard 1233. 1998.

[8]. MeMVaTEx glossary. Public report, *http://www.memvatex.org* . November 2007.

[9]. MeMVaTEx State of the Art. Public report, *http://www.memvatex.org* . April 2007.

[10]. W. Milam. *MBSE: What is it really?, In* SAE Engineering Propulsion Controls Symposium, Detroit, USA, September 05.

[11]. OMG Specification. *OMG System Modeling Language (OMG SysML™) Specification*. Final Adopted Specification ptc/06-05-04. May 2006.

[12]. OMG Specification. *A UML Profile for MARTE, Beta 1*. OMG Adopted specification ptc/07-08-04. August 2007.

[13]. OMG Specification. *UML Testing Profile v1.0*. Final Adopted Specification ptc/05-07-07. July 2005.

[14]. OMG Specification. *XMI MOF 2.0 / XMI Mapping Specification, v2.1.1*. December 2007.

[15]. HIS Working Group, results, Simulation and Tools Information on RIF 2004: *http://www.w3.org/2005/rules/wiki/RIF_Working_Group*

[16]. XRI from XERIF project (XML-BasEd Requirements Interchange Format). *http://2005.reconf.de/AcevedoM_247.pdf*

[8] See ARTiSAN web page : *http://www.artisansw.com*