Requirement management from System modeling to AUTOSAR SW Components

A. Albinet[1], L. Quéran[2], B. Sanchez[1], Y. Tanguy[3]

1. Continental Automotive SAS, 1 Av. Paul Ourliac BP 83649 31036 Toulouse
2. Geensoft, 120, rue René Descartes 29280 Plouzané
3. CEA, LIST, Laboratory of model driven engineering for embedded systems (LISE), Boîte Courrier 94, Gif sur Yvette 91191 France

Abstract: Problems detected in the V&V phases or even after deliveries can be huge regarding the cost associated to an early detection during the development cycle. This is usually linked to lost or misunderstanding during exchange of specification between the different teams of the development process. Furthermore, the description of software component as proposed by AUTOSAR, suffers from the impossibility to give a description of such component at a higher abstraction level.

This paper intends to present the methodology and the toolset achieved within the EDONA [1] work package 1. First, the EAST-ADL2 [4] system architecture description to AUTOSAR [2] software components modeling is described. In a second part, this paper focuses on the problem of requirement linking and traceability management during the development. In the last part EDONA results regarding verification and validation during the modeling process are discussed. This work is tested in EDONA against an industrial use case provided by Continental. This use case consists in an understandable function within vehicles, the Turn Signal Indicator, which will be used all along this paper to illustrate the proposed method and toolset.

Keywords: Modeling, Methodology, Requirements, EAST-ADL2, AUTOSAR.

1. Introduction

Regular chaos reports have shown that five of the top ten challenged factors for software intensive system project are strongly related to requirements specification being not reliable, inconsistent, unclear, or basically lost during exchange of specification between the different companies or teams at the different stages of the development process. Such errors can occur during contents exchanges and communication issues, leading to loss or mismatch of information. In the automotive, such issue is facilitated by the OEM-Suppliers organization and the multi discipline environment (system, hardware, software, basic software).

In this context, requirement traceability and analysis is a key issue in a design flow for electronic embedded systems. Industrials from IT have proposed and developed standards and engineering tools which partially cover these needs. The relationship between the initial expression of requirements and their impact on solution models is not fully established. Despite a lot of efforts, requirement management and traceability still remains a challenging problem in the automotive industry. Automotive applications design process should comply with safety standards (ISO26262) and customer expectations which impose vertical, horizontal and bi-directional traceability of requirements.

AUTOSAR is already recognized as a major success for interoperability, integration and communication needs in the automotive industry. Despite of that, the description of software component as proposed by AUTOSAR, suffers from the impossibility to give a description of such component at a higher abstraction level. Being able to represent a system at several abstraction levels or from several points of view is a key requirement in order to deal with system complexity and to allow system verification earlier than during the AUTOSAR implementation phases.

As a response to this need the EAST-ADL2 language has been defined, and taken into account by the methodology promoted in EDONA. Also the methodology plays an essential role in order to provide a common understanding of the tasks assigned to each actor, the roles involved actors should play, and the explicit definition of the inputs and outputs work products expected for each considered tasks.

The EDONA project focuses on processes, method and tools to improve the efficiency of embedded software development in the automotive domain. It aims at providing an open and seamless development platform supporting the whole development cycle. In particular, this platform offers modeling tools dedicated to the description of automotive software during the modeling steps proposed by the methodology. These models are

defined at different abstraction level (organic view, functional view, software view) with separate languages: EAST-ADL2 for system modeling and AUTOSAR for software modeling. These modeling tools are directly coupled with a requirement traceability management tool in order to keep requirements as a central consideration during the development and ensuring the consistency of modeled requirements gathered at functional analysis description steps, to detailed design of software components. In addition, validation tools are integrated in the toolset to verify that functional and non functional properties are properly respected by the models, and to make an early detection of possible integration issues considering hardware architecture, that can be distributed.

This paper intends to present the methodology and the toolset previously mentioned. First, the modeling phases from EAST-ADL2 system architecture description to AUTOSAR software components modeling are described. In a second part, this paper focuses on the problem of requirement linking and traceability management during the development. In the last part EDONA results regarding verification and validation during the modeling process are discussed.

All these results have been achieved within the EDONA work package 1 led by Continental, leveraging some work already done in several projects, in particular the MeMVaTEx [3] project which provides a methodology for requirements traceability, ATESST project which provides a new EAST-ADL2 profile with Safety features and AUTOSAR link, and TIMMO project which introduces a new timing language TADL for EAST-ADL2 and AUTOSAR. This work is tested in EDONA against an industrial use case provided by Continental. This use case consists in an understandable function within vehicles, the Turn Signal Indicator, which will be used through this paper to illustrate the proposed method and toolset.

2. Requirement Engineering

The TSI (Turn Signal Indicator) is a good example which looks simple but turns out to be quite complex at the end. The TSI "System" can accept a lot of different inputs depending on human activation or vehicle events and can activate various actuators as it is described in Figure 1. Also depending on the vehicle product line, the sensors/ actuators and even the hardware architecture can be different: Steering Column Stick (SCS) or Steering Wheel Button (SWB). Weak requirements engineering with this kind of "simple" function can be quickly critical to design if requirement models are not traced into solution models. Using the EDONA platform traceability tool will solve part of the problem as it is described in the following.

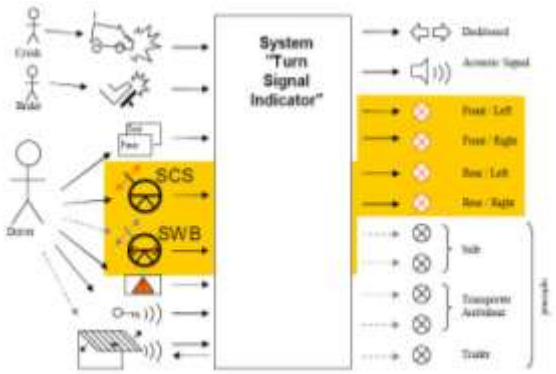

Figure 1 : TSI "System" Overview.

In system and software development, Requirement Engineering (RE) is one of the main activities which are difficult to manage correctly due to the amount of heterogeneous information used. These documents are coming from different sources and have different forms. They can be for example text files under word or any kind of text editors, MATLAB™ or STATEMATE™ models or even verbal requests that need to be formalized.

In first steps requirements, which specify what the system should be or perform, are collected from the different stake holders and integrated into a data-base tool named DOORS™. Then they are analyzed and compared to existing requirements. During this clarification phase with the Customer, requirements still undergo significant changes. When this phase is ending, the requirements are refined in Requirements Specifications that can be compared with existing requirements. The goal is to reuse as much as possible existing requirements linked to existing solution to optimize the effort. This step of RE can be done in different phases of the Development cycle (see Figure 2).

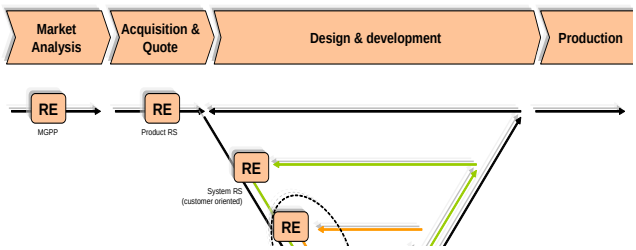

Figure 2: Requirement Engineering versus Development life cycle.

Depending on the level of refinement, managed documents will specify various requirements types: system or SW functional requirements or Non functional requirements like HW environment or testing requirements. They can be created or already

existing as reusable assets. From this documentation data-base, the developers can prepare their Requirements Specification package which represents entry documents needed for design. Next steps will be to link requirements package to the model solution. This will be described in chapter 4, but for now the next chapter will propose how to design efficiently a system model solution.

3. System modeling

The EAST-ADL2 specification is the main deliverable of the ATESST European project. This chapter does not intend to provide a complete overview of the language, for this matter please refer to available documentation or papers like [4] or [5]. Still a basic reminder of the EAST-ADL2 structure and concepts is given in the first part, and then the focus is placed on EDONA improvements in the modeling tool. Finally, the interoperability issue between EAST-ADL2 and AUTOSAR and EDONA tooling support for this are discussed.

3.1. A short overview of the EAST-ADL2

The EAST-ADL2 consists in a metamodel describing the language concepts and their relationships to each other. The goals of this domain specific language are twofold, firstly it brings some abstraction capabilities regarding AUTOSAR, and secondly it provides side modeling features that enable early analysis and validation during the system development.

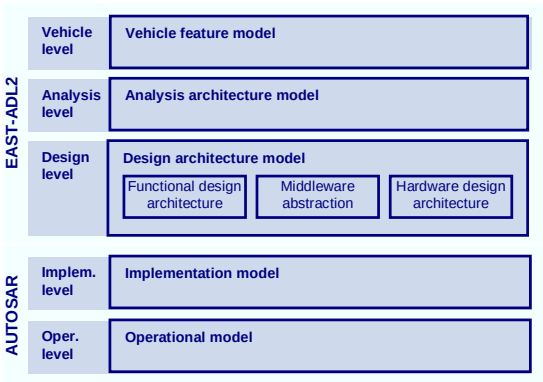

Figure 3 : Modeling abstraction levels in the EAST-ADL2

The **Figure** 3 gives an overview of the model abstraction levels proposed by the EAST-ADL2 to describe automotive systems. The vehicle level aims at defining the product from a feature point of view and capturing possible variability shared by a set of products. These features are derived into important functions at the analysis level. These functions are then refined at the design level, to reach a fine-grained functional architecture description. At this step, abstractions of the hardware and AUTOSAR basic software are introduced. The AUTOSAR language also takes part in this process as the modeling language used for the implementation and operational levels.

3.2. Modeling tools for the EAST-ADL2

The EDONA project targets the construction of an open platform facilitating the realization of chains of development trade modular, interoperable and adaptable to the various needs of the actors and trades of the car industry. In particular, the first work package provides the tools for EAST-ADL2 modeling. During the ATESST project, the EAST-ADL2 has been implemented as a UML profile. One advantage of such implementation is to benefit from existing UML tools that can offer a full graphical modeling support for a reduced development cost. EAST-ADL2 is implemented over an Eclipse-based open source UML graphical editor, Papyrus. EDONA leverages this work initiated during ATESST by improving the profile implementation and Papyrus customization features.

One enhancement is related to the profile implementation itself. The EAST-ADL2 profile has been implemented as a "static" profile, which means that the stereotype application relies on a Java implementation rather than being dynamically instantiated at runtime by reading the profile description. This is achieved by converting the profile into an EMF data model and generating the model access code from this model. As a result, the profile implementation becomes quite close to a classical EMF data model implementation, and enables the use of some EMF compliant tools (model transformation engines for instance) directly referring to the EAST-ADL2 profile. Our implementation takes advantage of the static profile nature to provide an implementation to the derived properties defined in the profile. A derived property is usually a read-only property which value can be automatically computed depending on the model context. Without such implementation derived properties are often ignored or manually filled in which implies a risk of modeling error and a loss of time. Another improvement related to the "static" nature of the profile is the contextual selection of the icon associated to stereotypes. The UML metamodel allows to add several image to a stereotype but does not provide any mechanisms to select which image is supposed to be used during modeling. This choice can be automated in Papyrus when a static profile is used, the chosen image being return by an additional method ("getImage") implemented in the static profile.

The second enhancement brought by EDONA is related to the graphical representation of EAST-ADL2 concepts in the editor. Specific icons have been applied to most stereotypes and are used in the editor wherever it makes sense.

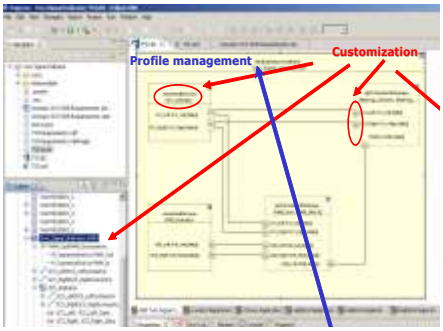

Figure 4 Papyrus editor graphical customization for EAST-ADL2

The au-dessus figure shows the icons associated to EAST-ADL2 concepts used in the different part of the editor (model explorer, creation tools, diagrams, property view) for a better understanding and readability. The new icon set defined for language support appear here as a replacement for default UML related icons. The palette containing creation tools has been customized for EAST-ADL2, meaning that it becomes possible to create EAST-ADL2 element directly without the need to create a UML element first and then apply a stereotype on it. This customization has been initiated in ATESST and extended in EDONA with the possibility to specify post actions to any creation tool, in order to create a pattern rather than a single element, or to define default values.
Finally, an integration of EMF validation is included and allows the specification of OCL or Java modeling rules that can be verified with batch validation against the model.

3.3. EAST-ADL2 and AUTOSAR interoperability

The EAST-ADL2 does not cover the implementation and operational levels. Actually this would be useless as AUTOSAR already exists and plays this role perfectly. The problem is that using several languages and therefore several modeling tool in the development introduce discontinuity in the process and likely interoperability issues.
Such interoperability consideration is largely simplified by the technical choices made by Papyrus and ARTOP the basic support for AUTOSAR used in EDONA. Both tools consist in a set of Eclipse plug-in with a data model that relies on the EMF and offer the same kind of Java API to access and manipulate models. The interoperability between these formalisms is resolved in EDONA by a model transformation component, named ARGateway, taking the most detailed EAST-ADL2 model (Design Architecture Model) as input and producing an AUTOSAR model. The "translation" of concepts between these languages is not quite difficult for most of them as EAST-ADL2 was initially defined as an abstraction of AUTOSAR. Moreover, both languages use the same kind of component model (with variants of Components, Ports, and Connectors) to depict functional or software architecture.
One major difficulty comes from the fact that a Design Architecture model and AUTOSAR model do not depict the system from the same abstraction level. This concerns the functional architecture from EAST-ADL2 which is supposed to be converted in software architecture in the AUTOSAR model, not only by a refinement process but also by an architecture refactoring step.

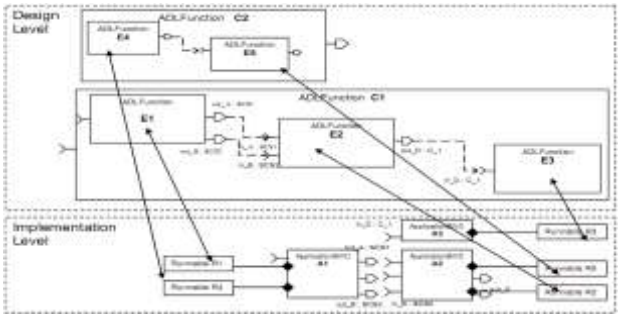

Figure 5 Example of functional to software mapping (extracted from [6]

The **Figure 5** is extracted from [6] where the main mapping rules between EAST-ADL2 and AUTOSAR are given and illustrated how the functional and software architecture may differ. Given the fact that an elementary (which is not further decomposed into sub functions) ADLFunctionType is translated into a RunnableEntity, the way RunnableEntity should be grouped into ApplicationSoftwareComponentType in the software architecture can completely differs from the functional architecture. Reason why this architecture differs may vary: allocation constraints of software component on a physical hardware topology, timing matters, safety considerations…
In a first version, ARGateway implemented two mapping heuristics.

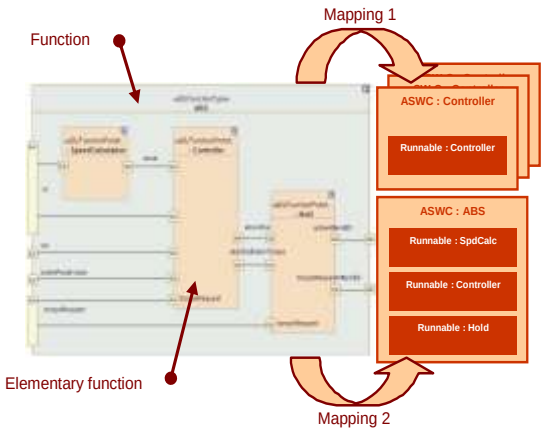

Figure 6 Function – Software mapping strategies implemented in first version of ARGateway

As shown in the Figure 6 above the implemented mapping choices were the following:

- Each elementary function transformed into a software component with a single Runnable
- Each composition of elementary function transformed into a single software component, and the composed elementary functions transformed into runnable and mapped to the software function.

These mapping choices could not be easily configured by the user and the result would requires in most situation to be manually re-factored in the AUTOSAR modeling tool used to complete the modeling at the implementation level. The new version of ARGateway realized in EDONA tackle these limitations and introduces some mapping flexibility by driving the transformation with a mapping model provided by the user.

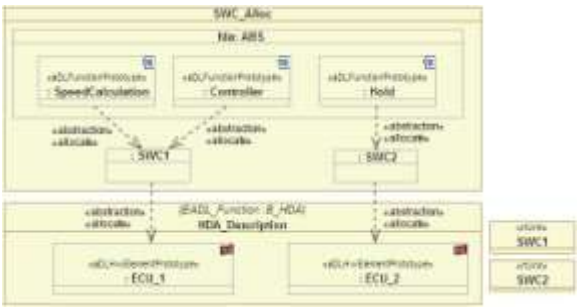

Figure 7 Example of allocation model

The **Figure 7** is given as an example of allocation model that can be used by ARGateway to map functions to software components. To define allocations, a minimal set of concepts from the MARTE profile is used. MARTE is a specific language for the modeling and analysis of real time and embedded systems. This language is not detailed in this paper, any additional information can be found in the language specification [7] . For our concern, only "Allocate" (or alternatively "Assign") and "RTUnit" from MARTE are used. "Allocate" describes a mapping of an entity on another, function to software, or software to hardware in this case. "RTUnit" is used here to represent a software component. ARGateway generates the AUTOSAR model with two pass, the first pass is pretty closed from the first mapping strategy shown on **Figure 6**, then the second pass reads the allocation model in order to create software component, associate runnables to these components and possibly re-factor the communication means (using connector or inter runnable variable) depending if runnables belong to the same software component or not.
The AUTOSAR model generated by ARGateway is obviously not complete as it is generated from a model define at a higher abstraction level, containing less details, but avoids a tedious and error prone manual step of AUTOSAR model creation. The generated AUTOSAR model respects AUTOSAR XML definition and can be used in any AUTOSAR modeling tool to go further on the implementation model.

4. Requirement traceability and analysis

Requirement traceability and analysis may appear as a theoretically simple problem. But the practice of it faces a major difficulty linked to the essence of the problem: tracking requirement throughout a project life cycle often implies interfaces with the used tools, and these tools are often different both on a company basis and on a life cycle stage basis.

In spite of its intrinsic qualities, UML2 is rarely used as the primary source in which requirements are defined in industrial projects. Requirement specifications often start with an office suite (word processor, spreadsheet) and, in contexts in which the importance of these requirements has been recognized, are often captured in databases with more or less requirement specific features. An example of such a specialized database is MKS INTEGRITY or TELELOGIC DOORS, which centralizes Continental's requirements in its Turn Signal Indicator project.

But even in the most advanced cases of specialized databases, few solutions are offered to connect requirement management to other tools with the appropriate granularity, and among those, to the tools used by development teams.

What is missing is the glue between the various tools used in a project life cycle.

Building on these grounds, GEENSOFT has created a product called REQTIFY, which makes it possible to extract and analyze the relations between requirements and cover links expressed in most industrially used tools. It is often used by project leaders and/or people involved in quality assurance (QA). However, until recently, it only offered features to simplify the concrete work of defining cover links in a limited number of proprietary tools.

Having recognized that tool integration is of utmost importance in practical requirement traceability, MyReq offers this tool integration framework. It is built on Eclipse which appears today as the integration platform on which many software editors and many industrial users are developing their own tools. Eclipse offers both a tested and proven base for development and an interoperability solution: a plug-in architecture, a set of views of common interest and a bunch of useful integration services such as a selection service and a Drag and Drop (DND) framework.

Being essentially a tool to facilitate interoperability on a practical point of view, MyReq uses these features. But as it focuses on tool integration, MyReq rejects the ambient paradigm of distribution ("tools that hardly work together except hopefully one day") and strives to keep minimal dependencies, which, as will be shown later, especially applies to UML profile management.

Focusing on facilitating and accelerating the work of development teams, MyReq also offers distinctive features, among which:

- A default configuration that works out of the box, although it can later be tailored to specific needs.
- A real time requirement analysis engine, because coverage information is more efficiently provided in earlier stages of the process.

4.1. From textual requirements to models

In the EDONA work package 1, MyReq has been applied to the Turn Signal Indicator use case provided by Continental.

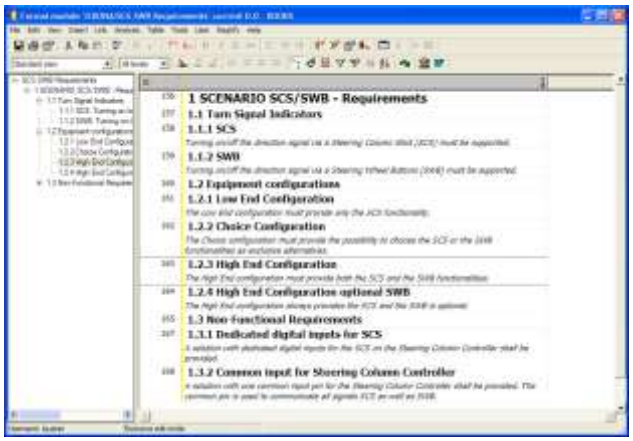

Figure 8 : TSI requirements in DOORS.

The TSI requirements are defined in a DOORS (Figure 8 above) database and in MICROSOFT WORD documents. A common feature of both specification formats is that requirements expressed in natural language and enhanced by attributes that only partly map to the formal EAST-ADL2 profile. For example (see Figure 9), a requirement may be typed as a "Functional req.", which can understandably be mapped to an "EAST-ADL2::Requirements::FunctionalRequirement".

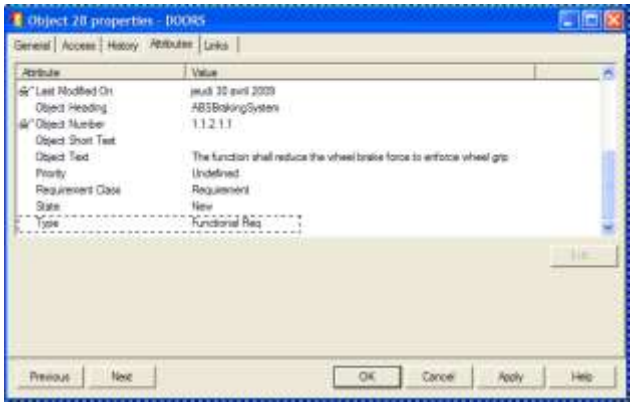

Figure 9: DOORS requirement attributes

At this point, the immediate problem is that UML2 models can only handle requirements as UML2 classes to which the appropriate profile stereotypes are applied. Manually creating these requirements in UML2 models is feasible with intensive use of clipboard, but it rapidly proves to be a both tedious and error prone process. MyReq connector for Papyrus developed in EDONA offers actions to automatically create the requirement models, thereby exposing in UML2 models the requirements created by any tool for which a connector has been developed.

The following table presents the main connectors that have been developed for MyReq, both in EDONA and proprietary projects.

Connector	Status
Eclipse JDT	Open source
Eclipse CDT	Open source
Papyrus 1	Open source
Eclipse MDT Papyrus	Open source
Open Office	Proprietary
Autosar Builder	Proprietary
Reqtify	Proprietary

Noticeably, MyReq REQTIFY connector integrates most commercial tools used in industrial projects in Eclipse.

Earlier in EDONA project, MyReq connectors for EAST-ADL2 and SysML were written using the seducing "static" profile feature. But it rapidly turned out that profiles may evolve and do not always maintain backward compatibility. Connectors to modeling tools were thus reviewed to only depend on Eclipse UML2 API. The dependencies on a given profile version are configured as regular expressions in a preferences page that can be modified by the user either to support new versions of an already known profile or new profiles that were foreseen initially.

Figure 10 : MyReq attribute mapping preferences

The screenshot in Figure 10 gives a hint of the way the previously mentioned attribute mapping problem is handled.

MyReq consequently supports any profile in both Papyrus 1 and Eclipse MDT Papyrus. It could easily be extended to support any modeling tool that adheres to Eclipse UML2.

4.2. Covering requirements in models

MyReq exposes requirements from external tools to development teams working with tools built on the Eclipse platform. MyReq also goes beyond that point by offering the easiest, most user-friendly means of interaction/interoperability: drag and drop (DND).

The first DND based feature is that the user can drop requirement defined in connected tools to create and manually layout a requirement model. As already mentioned, this greatly alleviates the work of copying or updating requirements completely manually, while letting full control over layout in contrast with the automated import functions.

The second DND based feature is parameterized cover link creation. For example, dropping a functional requirement on a ConcreteVVCase is translated into the creation of an ADLVerify abstraction whereas dropping the same requirement on another functional requirement creates an ADLDeriveReqt realization (see Figure 11). A default configuration for this is provided, but it can at any time be modified by the (admittedly advanced) user to meet his/her particular needs.

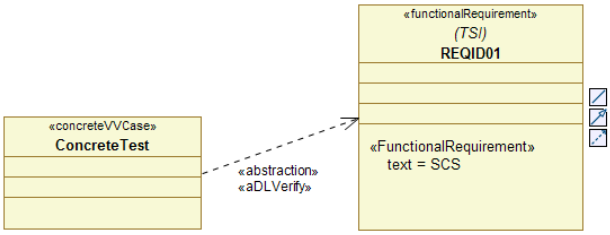

Figure 11 : Result of DND on a concrete VV test

MyReq is a practical tool in that virtually any operation involved in tracing requirements can be performed in one DND.

Among other noticeable features, a Requirement view presents automatically updated error/warning markers, which makes it possible to instantly get an overview of what has been covered and what remains to be covered.

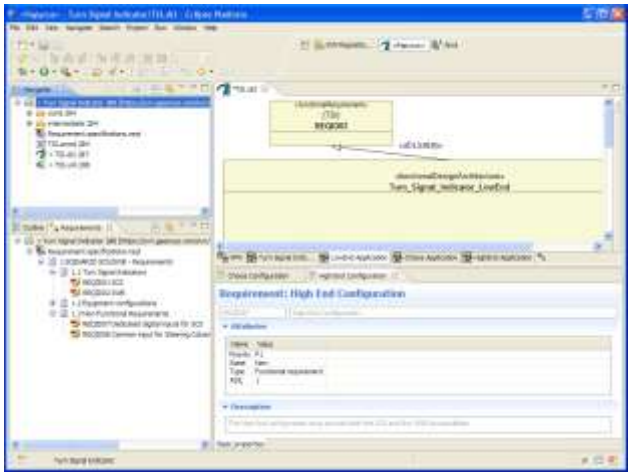

Figure 12 : Turn Signal Indicator edited with Papyrus and MyReq

Access to the source definition of requirement is also simplified by an action that, through the MyReq REQTIFY connector, opens the source document at the appropriate location.

4.3. Traceability analysis

This task is usually mostly performed by QA people and project leaders. Although MyReq offers a first level of traceability (limited to a depth of 1), it does not directly provide deeper analysis neither reporting features (Figure 13). These advanced features can be seamlessly accessed in REQTIFY.

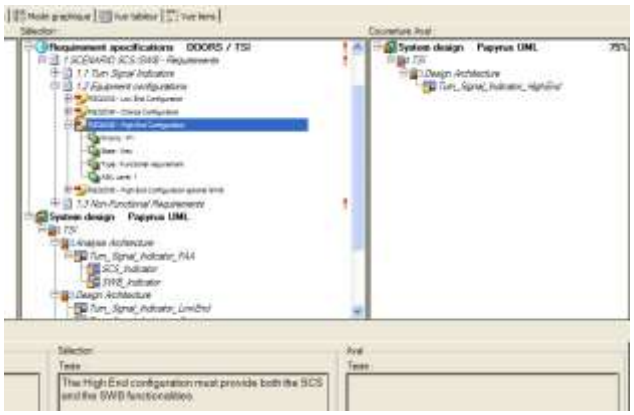

Figure 13 : Turn Signal Indicator coverage analysis in REQTIFY with Papyrus connector

The traceability analysis features typically answer the following questions:

- Are the specified requirements taken into account throughout the project life cycle?

- What are the impacts of modifying a given requirement?
- What requirements are no longer fulfilled given a bug?

In support to its traceability analysis and extensive reporting features, REQTIFY also offers a function that directly opens Eclipse on the appropriate view showing a covering element or requirement.

4.4. Final overview

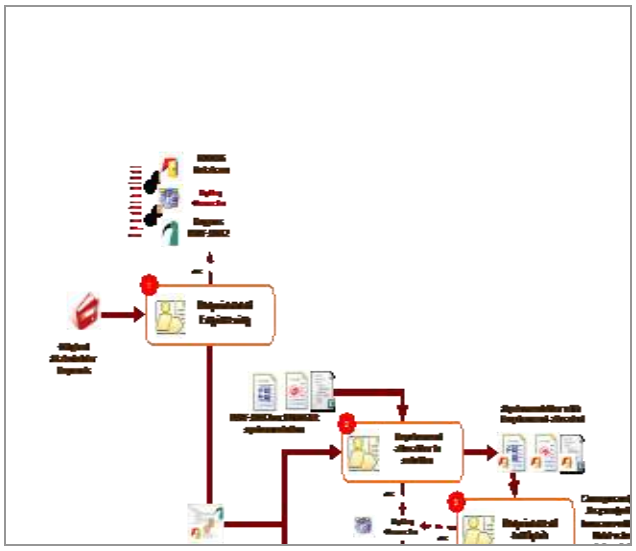

Figure 14: Full requirement traceability process

As illustrated in the Figure 14, the tools previously described provide a continuous tool chain, from requirement specification to implementation on one hand, and verification and validation on the other hand. Cost reduction in the whole life cycle is expected as a result of this continuity and the efficiency of the tools used to achieve it.

5. Conclusion

We have presented a seamless Model Driven development with the EDONA platform on an automotive application. The EDONA platform is supporting Requirement Engineering as main activity in all levels of the development cycle synchronized easily with the other activities like model solution design, coding or even testing thanks to MyReq traceability plug-in. In the same way, the modeler Papyrus is supporting directly the EAST-ADL2 profile to model all high levels of the architecture. For the implementation level, the ARGAteway plug-in allows the transformation of EAST-ADL2 artifacts into AUTOSAR SW Components. Finally requirements covering analysis is simplified thanks to REQTIFY and MyReq. This integration of all these plug-ins within the same platform guarantees interoperability between the different tools.

The next steps will be to introduce the variant handling into the Solution model and finalize the introduction of MARTE to integrate easily timing properties into the model and to allow Timing analysis as soon as possible in the development phases.

6. References

[1] Edona project : www.edona.fr

[2] The AUTOSAR Consortium : www.autosar.org

[3] MeMVaTEx project : www.memvatex.org

[4] The ATESST Consortium: "*EAST-ADL 2.0 Specification*", www.atesst.org, 2008.

[5] P. Cuenot, P. Frey, R. Johansson, H. Lönn, M-O Reiser, D. Servat, R. Tavakoli Koligari, D.J. Chen. : "*Developing automotive products using the EAST-ADL2, and Autosar compliant architecture description language*", ERTS, Toulouse, 2008.

[6] The ATESST Consortium: "*D3.2 - Report on the behavior modeling in the EAST-ADL2.0*", 2008.

[7] OMG: "UML Profile for Modeling and Analysis of Real-time and Embedded Systems (MARTE)", OMG Document Number: ptc/07-08-04, 2007.

7. Glossary

AUTOSAR:AUTtomotive Open System ARchitecture (www.autosar.org)

ATESST: Advanced Traffic Efficiency and Safety through Software Technology (www.atesst.org)

UML: Unified Modeling Language

EMF: Eclipse Modeling Framework