Grant FFI 2013-01296

Synligare utveckling av inbyggda fordonssystem – visualiserad kravhantering och samverkan

Visibler Development of Embedded Automotive Systems - Visualised Requirements Management and Collaboration

Report type — Deliverable D3.1

Report name — Report on Tooling

Dissemination level	**Public**
Status	**Final Release**
Version number	**2.0**
Date of preparation	**2016-05-20**

Authors

Editor	E-mail
Jan Söderberg	jan.soderberg@systemite.se

Authors	E-mail
Henrik Kaijser	henrik.kaijser@volvo.com
Henrik Lönn	henrik.lonn@volvo.com
Mikael Stolpe	mikael.stolpe@arccore.com
Jan Söderberg	jan.soderberg@systemite.se
Urban Ingelsson	urban.ingelsson@semcon.com
Ali Shahrokni	ali.shahrokni@systemite.se

The Consortium

Volvo Technology Corporation AB - Autoliv AB - Arccore AB- Systemite AB - Semcon Sweden AB

Table of contents

1 Introduction

This document is the report on results of work package 3 (WP3) of Project Synligare.

The overall effort in Project Synligare is to address specification of complex automotive embedded systems to provide an overview, ease review and management of requirements by means of defining structured information models for representing system specification, which will enable automatic generation of views and metrics. The purpose is to improve collaborative development of automotive embedded systems in comparison with the typical situation in which there is no common specification language. This involves addressing challenges like relating requirements and design with respect to configurations in a product family by applying various relations and searches in structured requirement information based on existing specification languages like EAST-ADL. Further, it involves management of specifications and requirements in accordance with ISO 26262. The aim is that the outcomes of Project Synligare should be tried on an example system, to validate the structured information models, the views and the metrics.

1.1 Purpose

The work in WP3 will result in different tool implementations, supporting the concepts and methods defined in the project [1], including technical validation on tool level. The tool implementations will be formally tried and evaluated in WP4, in scenarios emulating real industrial cases. The different tool technologies used in the implementation represents not only technologies widely in active use in the industry today for the various roles in Embedded Automotive Systems, but are also technologies well familiar within the project consortium.

1.2 Scope

The scope of WP3 is to implement tool support for the scenarios described in Chapter 2, for the tool technologies described further in this report.

2 Supported Scenarios

The implementation in EATOP, SystemWeaver and EnterpriseArchitect supports the specific use cases and scenarios described in this section. These scenarios represent a realistic project set-up including an OEM and Suppliers, and work as a basis for the practical integration and validation of the Modeling, Model Exchange, View and Metrics generation in the different tools.

The specification of these scenarios can be found in Section 2.2 in [1].

A technical challenge in managing the information in distributed projects with a high degree of collaboration is that the same information needs to be represented in multiple tool environments. This means that the same information can be altered – by mistake or on purpose – in the various locations. Still there must always be means of synchronizing and consolidate the information from different sources. This kind of synchronization and consolidation is today performed ad-hoc, with poor tool support. By means of the EAXML format, extended to support versioning and configuration management, such consolidation will get proper tool support.

2.1 Scenarios

Scenario	Supported in Tool
Requirements and Features	EATOP, SW, EA
Requirements and Functions	EATOP, SW, EA
Requirements and Hardware Components	EATOP, SW, EA
Functional Allocation	EATOP, SW, EA
Feature Tree with increment and products	Not validated
Scenario Filtering	Not validated
High level filtering from project level	Not validated
Analysis	EATOP
AUTOSAR Generation and Diff	Not validated
RFQ	EATOP, SW, EA
Annotation for SIL, Optional and Product Variants	Not validated
High Level Quality Measurements and Views	Not validated
Project Earned Value / Schedule	Not validated
Exchange – One-Way	EATOP, SW, EA
Exchange - Round-Trip	EATOP, SW, EA

2.2 Metrics

Metric	Supported in Tool
Requirement Validation Progress	Not validated
Requirement Allocation Progress	EATOP, SW

Realization Progress	SW
Verification Progress	Not validated
Safety-related Progress Metrics	Not validated
Product Metrics	EATOP

2.3 Views

View	Supported in Tool
Generic Tree View	EATOP, SW, EA
Viewing Related Model Objects	EATOP, SW, EA
Creating connector elements	EATOP, SW, EA
Compare and Merge	EATOP, EA
Increments of Features	Not validated
Features and Requirements	EATOP, EA
Functions and Requirements	EATOP, EA
Hardware Components and Requirements	EATOP, EA
Allocation	EATOP, EA
Feature-based Architecture Views	Not validated
Requirements vs Architecture	EATOP
Components, Ports and Signals	EATOP, SW, EA
Related Modeling Concepts	EATOP
Item Definition	EATOP
Functional and Technical Safety Concept	EATOP
Error Propagation Modelling	EATOP, SW
Illustrating Metrics	EATOP, SW

© 2016 Synligare

3 EATOP and ARTOP

EATOP is an open-source project which has the purpose of providing an Eclipse-based implementation of the EAST-ADL standard. EAST-ADL is a domain specific architecture description language used in the automotive industry. One important feature of EATOP is ability to give access to an EMF-based version of the EAST-ADL meta-model. It does this by serializing and deserializing an EAXML file and presenting the result in the EAST-ADL Explorer.

ARTOP is similar to the EATOP project with the distinct difference that it focuses on the AUTOSAR domain instead. Compared to EAST-ADL it does not have such a high-level representation of a system but has a more detailed description level.

3.1 Browser enhancements

The view EAST-ADL Explorer is used to display the model in a tree-view manner. Larger models often become difficult to view and navigate due to the nature of the models. There are often references to other parts of the models or one prototype might hold 100 children of the same type and the user is overloaded with information. There is therefore a need to enhance the view.

The enhancements provided are the following

- Element categories: If there are more than three elements of the same type in a model object these are grouped and stored in a virtual node.
- Context view: Elements often need to be considered together with its related elements. The context view can show:
    - Incoming and outgoing references.
    - If a FunctionPort is selected which is referenced in the context of a SafetyConstraint the ASIL level of that SafetyConstraint will be shown.
    - If a RequirementsModel object is selected the Requirement Hierarchy will be displayed.

3.2 TableView Plug-in

It is possible to get details about each object in the EAST-ADL model by using the Properties View. This can be very helpful whilst either trying to understand the model or looking for problems and errors. However, it can be difficult to compare properties of many objects at the same time. Currently, the way to do this is to look at one object and then look at the other and try to discern the difference between them. Due to this a way to look at the properties in a table was desired and a prototype has been developed.

The plug-in has support for multiple selection and groups properties of the objects together in columns. Each row is thus represented by one of the selected objects in the EAST-ADL Explorer. As can be seen in Figure 1, several objects have been selected and different properties are presented in the Table View to the right. The Table View does not simply take the standard properties of the selected objects from the Properties View but specifies which properties each object should provide to the table. This has the advantage of filtering out only those properties which might be advantageous to compare and not cluttering the interface with unnecessary information.

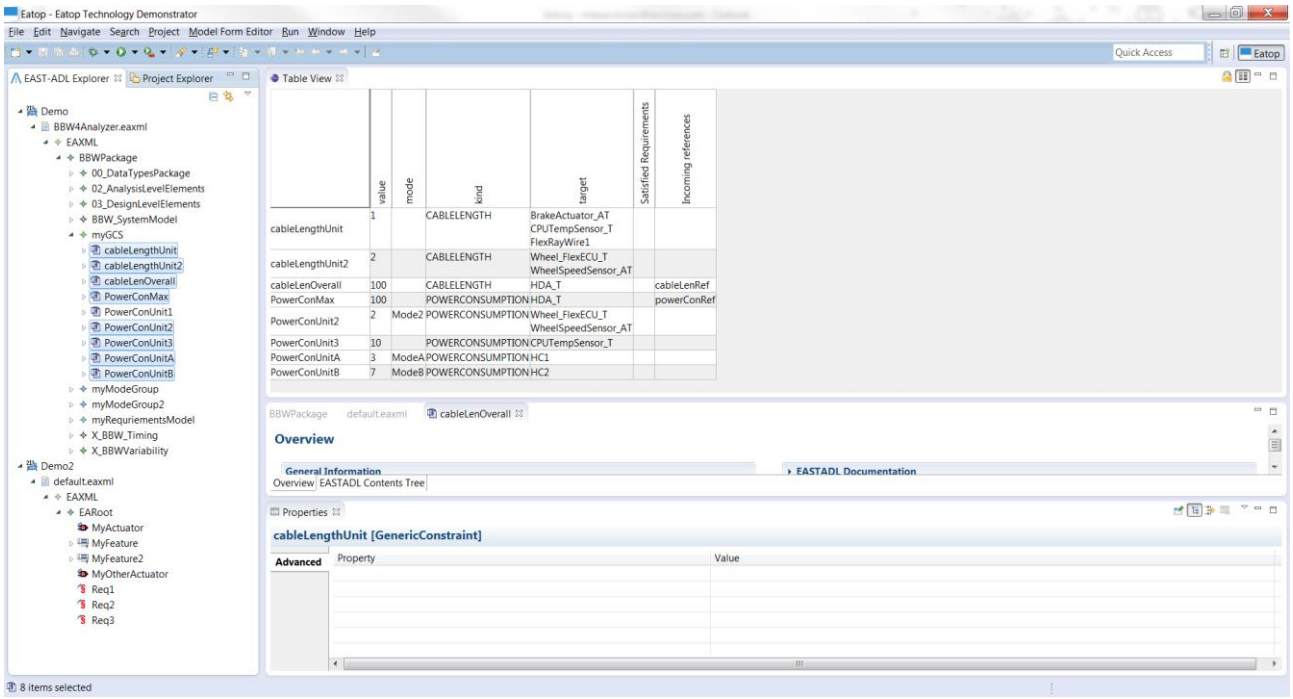


Figure 1, multi-select support in the TableView.

Furthermore, more advanced properties can be viewed in the table such as the “Incoming references” and “Satisfied Requirements” - columns seen in Figure 2. The TableView is still a prototype and a large development area of the TableView would be to increase the number of ways to access the information stored in the objects. It is currently possible to edit single elements in the table which is then propagated to the object in the model.

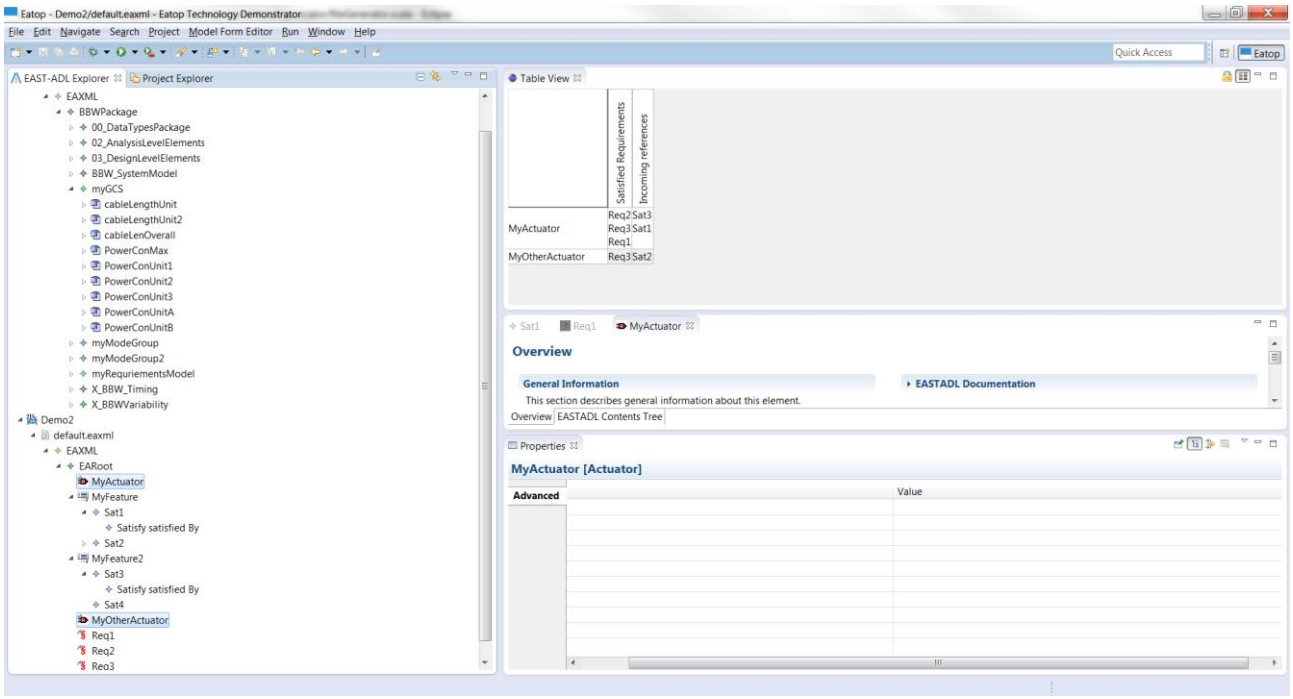


Figure 2, special properties shown in the table.

3.3 Model Compare

EMF Compare is a plug-in developed by Eclipse which has been integrated into the Synligare EATOP demonstrator. The primary use case is to diff two objects or files and compare the changes done to them. It is a good EMF replacement for the traditional text-based comparison done via SVN or GIT. Instead of comparing the changes done on each line it gives a more visual way to view which model objects has been changed, removed or added.

3.4 Create Connector Plug-in

When working with models in the EAST-ADL Explorer a common process is to create connectors between ports. This can be done through the explorer by creating a connector and manually linking the ports it should be connected between. It can be a rather tedious and time consuming process. Therefore, it is suggested to create a plug-in for EATOP which can assist the user when creating such connectors.

The concept focuses on aiding the user by visually presenting the ports which should be connected in a logical way. The trigger point should be the component prototype in order for the correct context can be used to check for validation. Figure 3 and Figure 4 demonstrate a mocked example of how it might function. As can be seen on the left side, when triggering the plug-in it sorts all the possible ports associated with the prototype which are possible candidates for creating the connector. Furthermore, when selecting one port the plug-in makes basic validation checks for possible matching candidates for the port. Validation should be kept rather simple, confirming only direction and client / server relations.

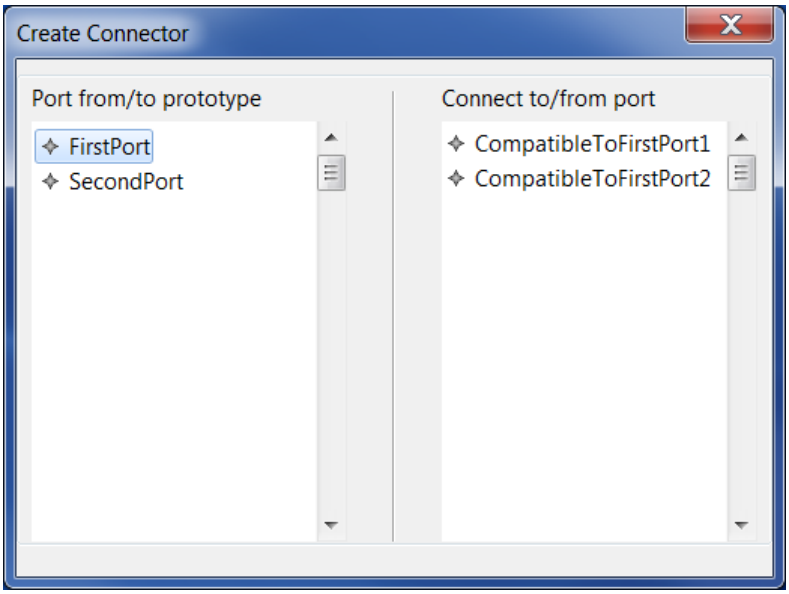


Figure 3, selecting the first port

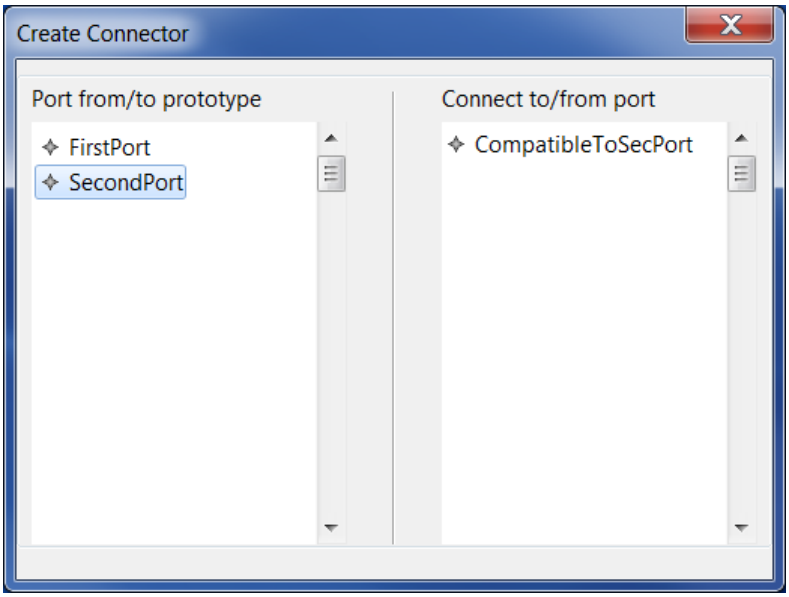


Figure 4, selecting the second port

After confirmation from the user the connector is created and the user should be informed about the result in an appropriate way. It is believed that a feature with this ability might increase the development process of users working with the EATOP tool environment.

3.5 Error Model Synthesis

Error propagation modelling is the basis for failure modes and effects analysis (FMEA) and fault tree analysis (FTA). EAST-ADL supports the structural definition of propagation models, and the propagation logic can be defined using different notations, depending on which kind of analysis and which tool is intended.

The Error Model Synthesis plugin takes a nominal model representing the functionality under development. It then automatically creates an error propagation model that structurally identical to the nominal model. On this basis, semi-automatic manipulations can be made to get an error model that reflects how errors are assumed to propagate.

The manipulations that are supported are

- Combining Ports

  Two or more ports in the nominal model may have an equivalent meaning from a fault propagation viewpoint. To make the propagation model clearer, these ports can be automatically combined, and the plugin manages its references to the target port in the nominal model and the connectors.

- Combining Prototypes (sub-models)

  Two or more prototypes (sub models) in an error model type may be combined to one, if it is deemed that this pair or group of functions can be represented by a common error model. A set of selected prototypes can be automatically combined, and the plugin manages its references to the target functions in the nominal model. The ports of the collapsed set will appear on the new, combined, prototype and the connectors are maintained.

Once an error propagation model is completed, the plugin is able to invoke an analysis tool, for instance HiP-HOPS.

3.6 HiP-HOPS Bridge

HiP-HOPS is a fault tree analysis and failure modes and effects analysis tool, and this plugin exports the EAST-ADL error model to HiP-HOPS format and invokes the analysis.

The HiP-HOPS bridge transforms a selected EAST-ADL error model to HiP-HOPS XML format.

This plugin is used to set versions on model elements and to detect version inconsistency between model elements and their error models.

3.7 Version Manager

The plugin relies on user defined attributes to annotate each model element with a version. The plugin initializes the model elements to a desired version number, and allows version increments of 0.1 or 1 on each element.

It also allows comparison of an ErrorModel's declared target version and the actual version of the element. In case of inconsistency, a warning is issued. In order to represent the consistency between an error model element and a target element, the plugin supports synchronization by setting the error model's target version to the current version of the target element.

3.8 Metrics

The Metrics plugin supports computation and visualization of the following metrics

1. **Allocated Requirements**
   A Requirement is considered allocated if there is at least one Satisfy Relationship linking it to a system element.
2. **Verified Requirements**
   A Requirement is considered verified if there is at least one Verify Relationship linking it to a verification case or procedure
3. **Realized Features**
   A Feature is considered realized if there is a Realize Relationship to a system element.
4. **Safety Goals covered by Functional Safety Concepts**
   A Safety goal is considered covered by a Functional Safety Concept if all its requirements

are derived by at least one functional safety requirement. A Requirement is regarded as a functional safety requirement if it is linked from a Functional Safety Concept.

5. **Functional Safety Concepts covered by Technical Safety Concepts**
   A Functional Safety Concept is considered covered by a Technical Safety Concept if all its requirements are derived by at least one technical safety requirement. A Requirement is regarded as a technical safety requirement if it is linked from a Technical Safety Concept.

Each type of metric is presented in its own view. The result of a metric calculation is shown in a pie chart.

3.9 Model Overview

The Model Overview plugin supplies a graphical overview of an EAST-ADL model. The overview shows for the currently selected model, elements on the different abstraction levels of EAST-ADL (Vehicle Level, Analysis Level, Design level and ImplementationLevel) and also “categories” (EnvironmentModel, SystemModel, Requirements, Variability, Timing and Dependability). Tooltips will show the names of the categories, the levels and the elements.

3.10 Linear Property Analyzer

Model elements can be annotated with properties like cost, weight or power consumption in a particular mode. These properties can be summed linearly over a package or a structure. This plugin calculates the sum of the property, in a certain mode and for a selected structure.

EAST-ADL uses GenericConstraint to annotate properties, and each constraint defines its kind and value and specifies the modes in which it applies. The plugin traverses the containment structure and sums up all GenericConstraints that are associated to elements in the structure. If the selected element is part of a type-prototype hierarchy (Functions and hardware components), its virtual tree is used instead of a containment hierarchy.

3.11 SGraphML Graphical Editor

The EATOP model is primarily edited and browsed as a tree structure. The graphical editor plugin complements the tree structure with a graphical editor, allowing drag and drop of tree elements onto a diagram which can be saved along with the EAXML file of the model.

The Graphical Editor provides a diagram view which is a direct mapping of the EAST-ADL model in the tree view. The user drags and drops elements from the tree view onto the diagram, and the editor represents the elements in a way that is consistent with the model and the semantics of dropped elements. On dropping elements with relations to existing elements, relations are added to the diagram. On clicking elements in the diagram, the corresponding element in the tree view can be found.

The diagram is stored in an sgraphml file, a format which extends graphml with diagram information, similar to the proprietary ygraphml extension from yworks.

3.12 Place-and-Route of Diagrams

This package is a plugin component that can supply placement and routing functionality to plugins, such as the SGraphML Graphical Editor. The elements are considered as graph nodes to be placed and the relations are considered as graph edges that, depending on their type, may be routed. The package places nodes and routes edges to accentuate the structure of the shown information by placing nodes that are connected to each other close to each other, and by routing edges around the nodes. The capabilities of the package are as follows:

- Elements with in and out ports are placed and routed to follow a left-to-right concept with inputs on the left and outputs on the right. This shows information flow in a way that is appropriate for functional architecture models and error propagation models.
    - Feedback loops are detected and handled.
- Elements with other types of relations are placed so that relation-based structure is shown and relations are represented by straight line edges (not routed) with a minimum of intersections.
- Sub-diagrams within diagram nodes (representing a hierarchy) are handled.
    - Edges between nodes on different levels of hierarchy or between sub-diagrams are handled and are used to advice the placement.
    - The geometry of a given diagram element is adapted to adjust to the dimensions of its sub-diagram.

3.13 Requirements Allocation Assistant (RAA)

EATOP Requirements Allocation Assistant is an editor that eases manual allocation of requirements to system model elements. RAA displays two tree views in parallel, one for requirements and one for model elements. The requirements and model elements can be in the same file or in two different model files.

RAA provides the following functionalities:

- Search and filter for both requirements and model elements based on type and attribute value.
- Hint functionality – offers suggestions of model elements for a selected requirement that could be allocated to. The suggestions are determined based on allocation links existent on a higher abstraction level.
- Allocation – creates satisfy links between one or more requirements and one or more model elements.

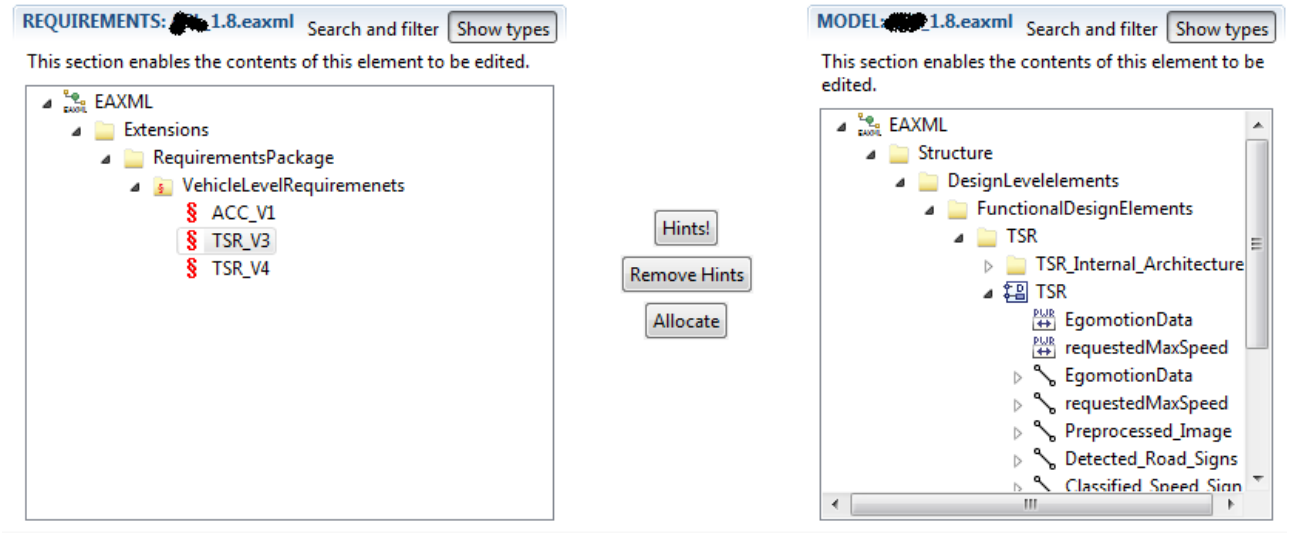


Figure 5 Requirement Allocation Assistant while looking for TSR in both requirements and design

4 SystemWeaver

SystemWeaver is a generic collaborative system modeling platform developed and sold by Systemite AB. The specific functionality of SystemWeaver is defined by the installed meta model and view extensions. The Meta model in SystemWeaver supports the core part of EAST-ADL, and extensions like the Requirements and Dependability packages.

For the Synligare project, Systemite hosts a server at synligare.systemite.net, and a suitable SystemWeaver client, along with all extensions to support special views developed for the Synligare project, can be installed by using the ClickOnce installation script available at http://apps.systemite.se/Synligare/setup.exe.

4.1 EAST-ADL Exchange

The EAST-ADL Exchange is based on transformation between the EAXML format and the native SystemWeaver XML format. The SystemWeaver XML format can then be used for various synchronization and consolidation operations towards a SystemWeaver database, including support for incremental imports and versioning.

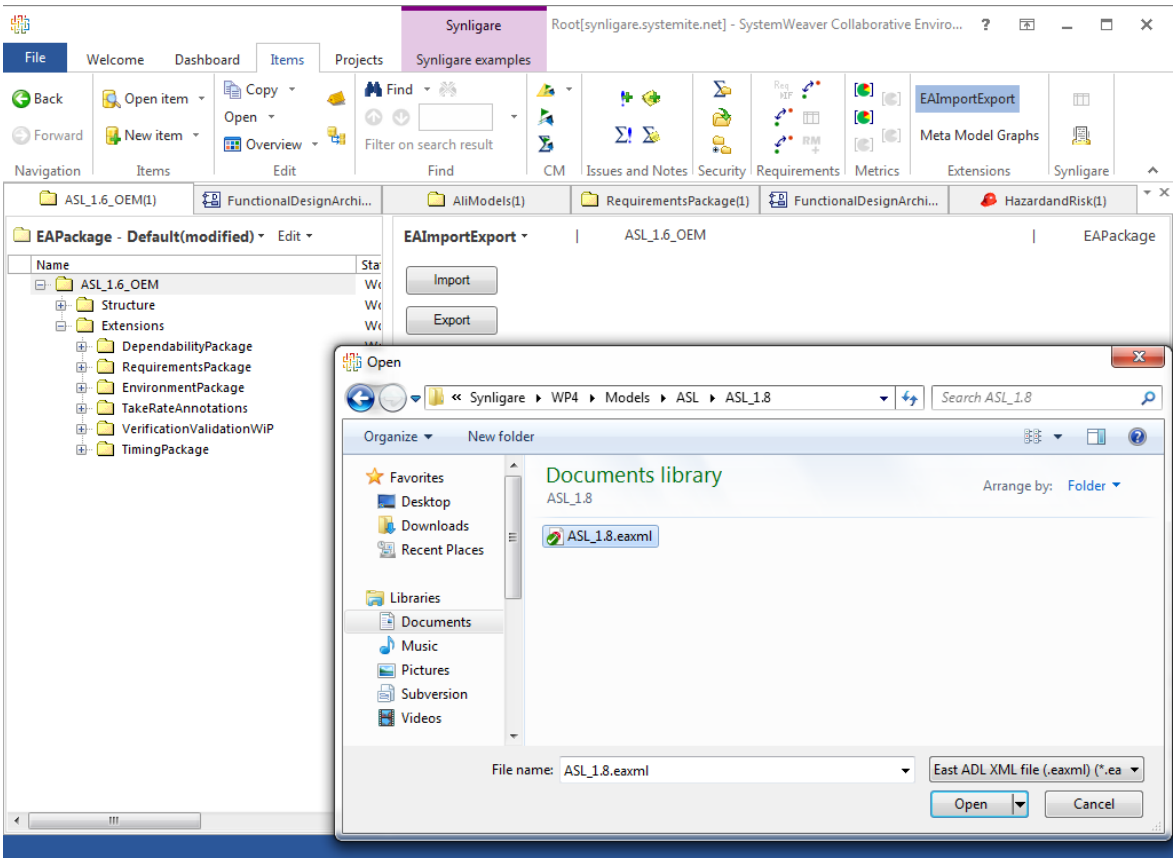


Figure 6 Import from and export to EAXML

The SystemWeaver EAST-ADL solution includes a plugin for import and export to EAXML. To activate the plugin, select an EAPackage on the left and click on EAImportExport in the top ribbon. Press the import button to import and select a valid EAXML file. The elements in the EAXML file will then be imported into the SystemWeaver database and be placed under the currently selected EAPackage. To export an EAPackage, press the export button and specify file name.

Note that the current version of the import/export plugin has some limitations. For instance the timing package is not included. A complete list of the support and limitations is provided at /Synligare/WP3/SystemWeaver/SynligareImportExportSupport.rtf.

4.2 Metrics

In SystemWeaver there is a possibility to define different kinds of metrics by specifying a traversal path of the data in the Definition tab. The data can then be represented in a variety of forms. As the below figures show, the results can be presented as grids, charts, and reports.

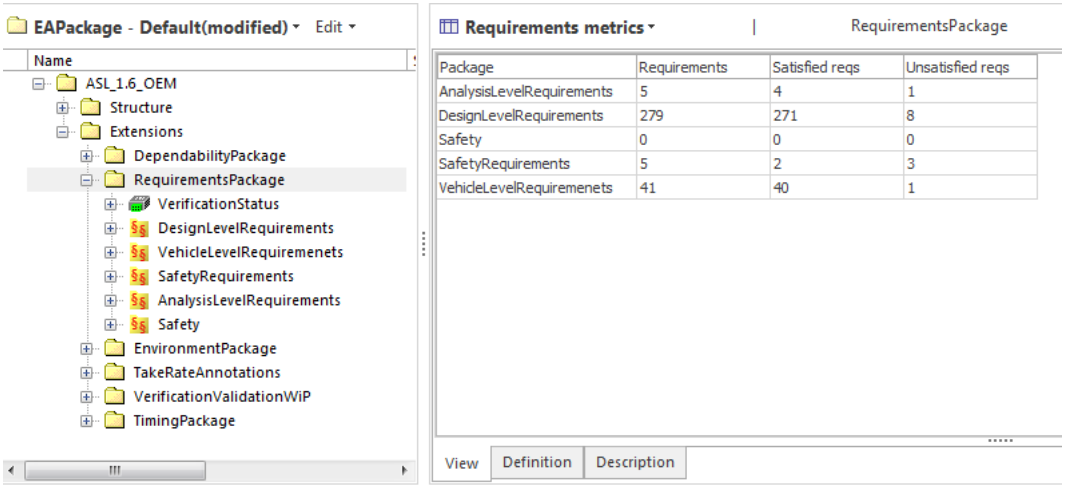


Figure 7 Requirement allocation metrics

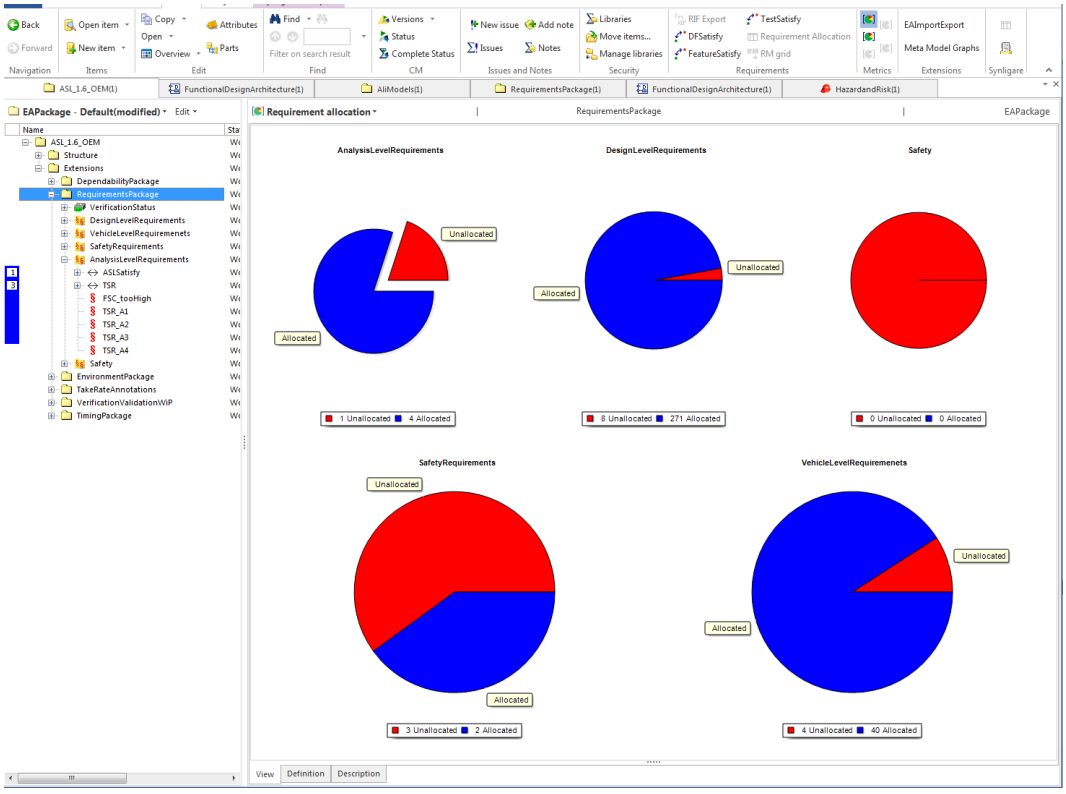


Figure 8 Requirement allocation metrics

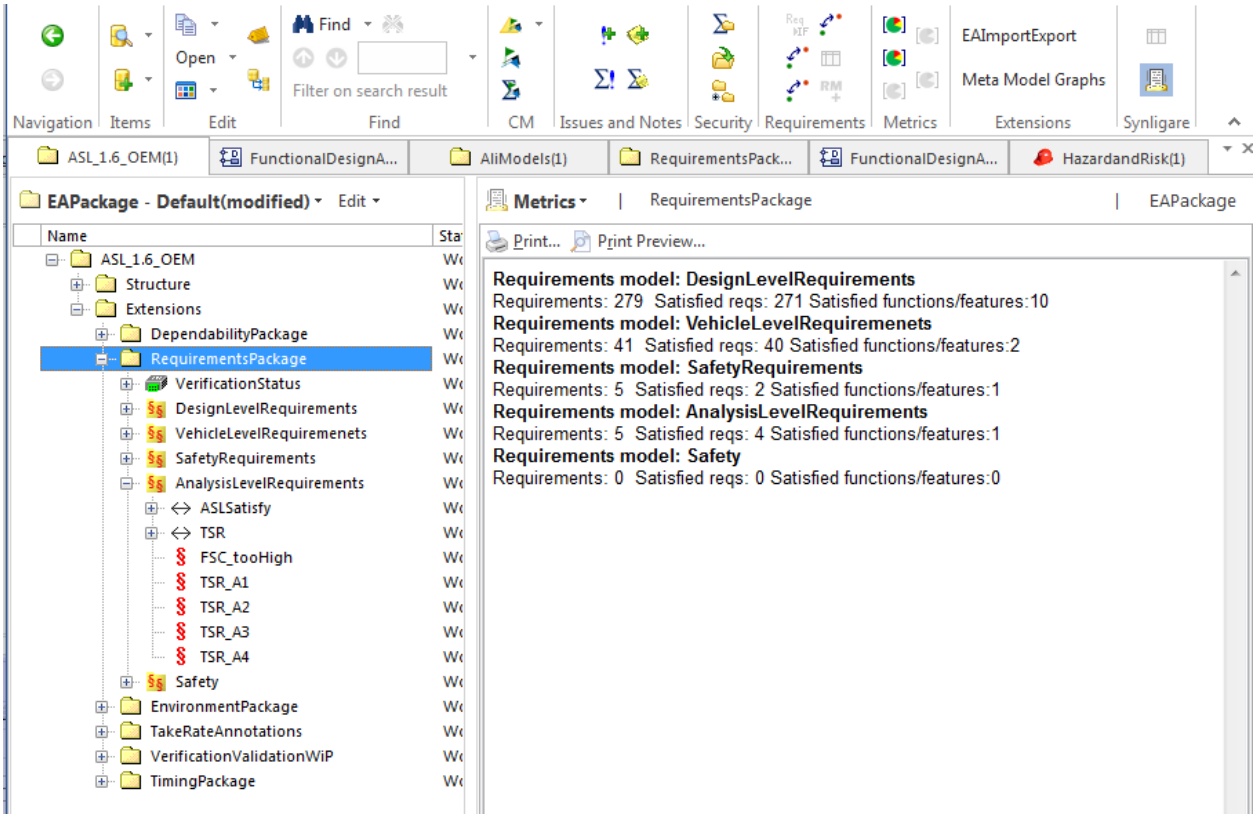
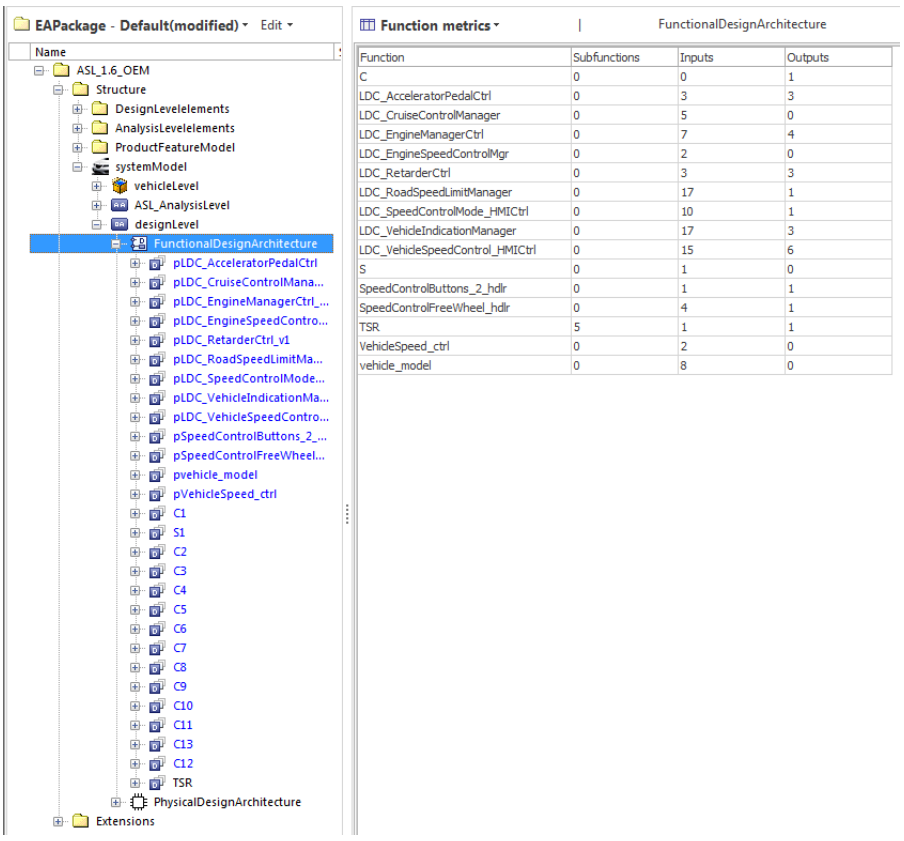


Figure 9 Requirement allocation metrics report

The following figures shows that the metrics can be calculated on levels deeper than one step to calculate the ports of all components in a design architecture.



Function	Subfunctions	Inputs	Outputs
C	0	0	1
LDC_AcceleratorPedalCtrl	0	3	3
LDC_CruiseControlManager	0	5	0
LDC_EngineManagerCtrl	0	7	4
LDC_EngineSpeedControlMgr	0	2	0
LDC_RetarderCtrl	0	3	3
LDC_RoadSpeedLimitManager	0	17	1
LDC_SpeedControlMode_HMICtrl	0	10	1
LDC_VehicleIndicationManager	0	17	3
LDC_VehicleSpeedControl_HMICtrl	0	15	6
S	0	1	0
SpeedControlButtons_2_hdlr	0	1	1
SpeedControlFreeWheel_hdlr	0	4	1
TSR	5	1	1
VehicleSpeed_ctrl	0	2	0
vehicle_model	0	8	0

Figure 10 DesignFunctionType metrics

4.3 Allocation

Another view in SystemWeaver is the allocation view which can be used to allocate a certain type of element to another type. The figure below shows the allocation of DesignFunctionTypes on ECUs.

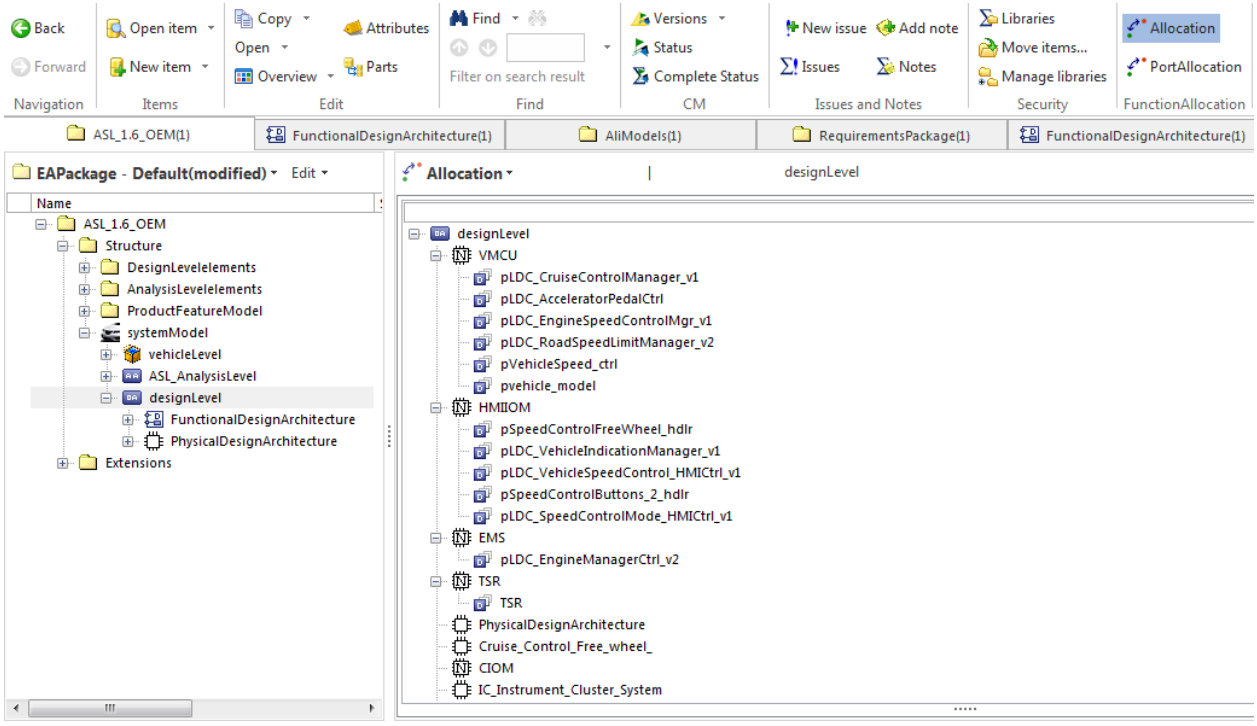


Figure 11 Example of allocation view for DesignFunctionType to HardwareComponentType

4.4 Table view

Another type of view in SystemWeaver is the grid or table view. Following a certain definition, the user can specify what the rows and columns in the grid represent. The following grid lists all the requirements in a requirement model and also shows the requirement description or text. The Synligare database includes several grid definitions similar to the one in the example figure.

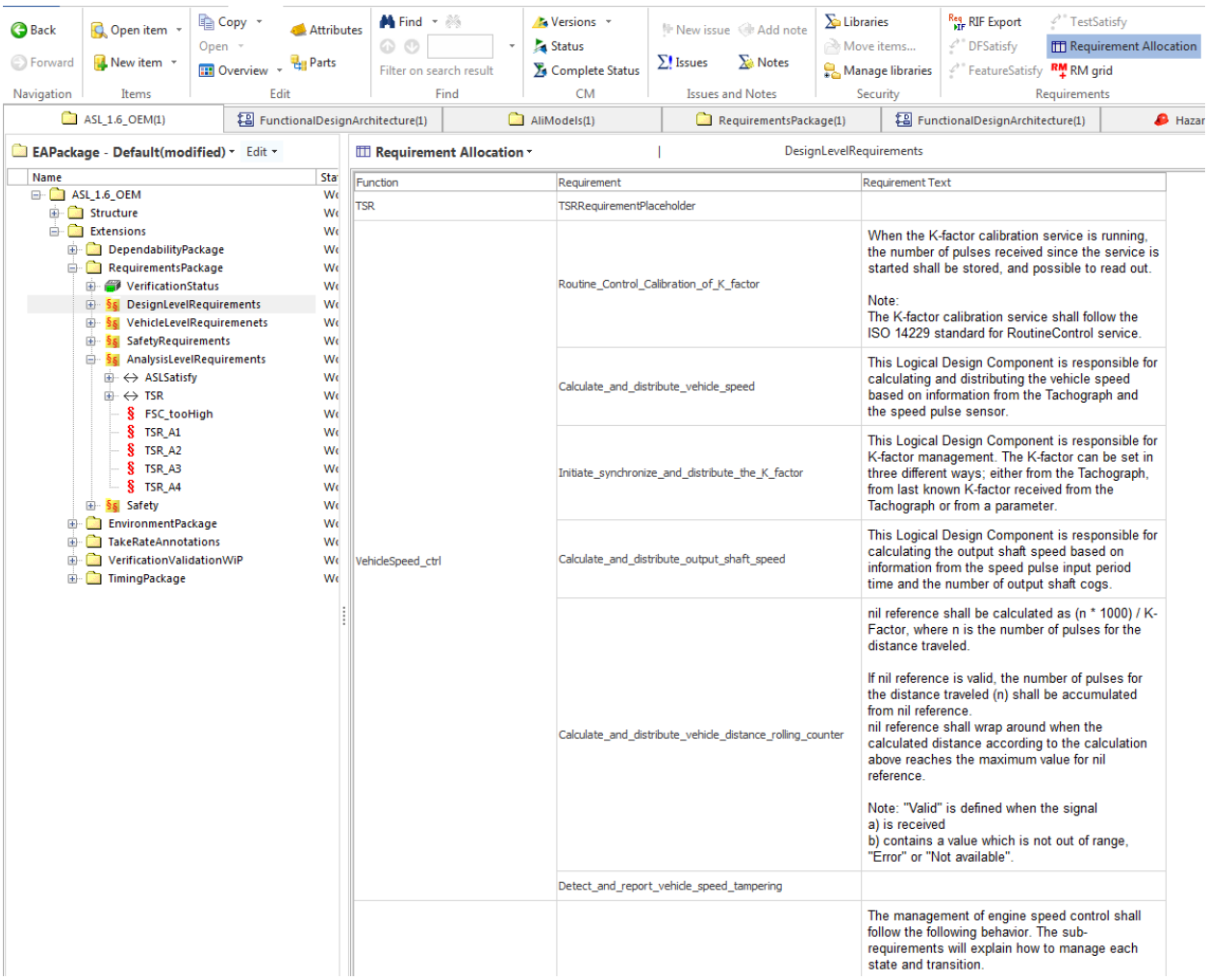


Figure 12 Requirement allocation grid

4.5 Graphical representation

Another way of representing the data in SystemWeaver is the graphical view as seen in the following two images. Following a definition, SystemWeaver synthesizes the graph based on the selected context in the tree view. The first graph shows the connection between hazards, hazardous events, and safety goals in a HARA analysis. The second graph represents a design graph for DesignFunctionTypes in a design architecture. To the right of Figure 14, a neighborhood graph is visible that shows the selected component in the middle with a red periphery and all the components that directly communicated with that component.

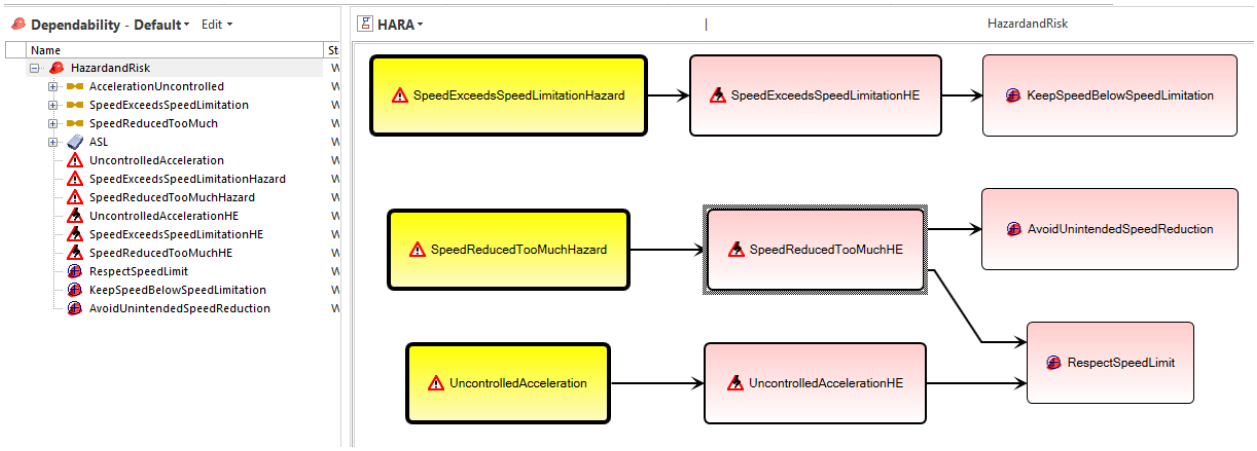


Figure 13 Graphical representation in SystemWeaver

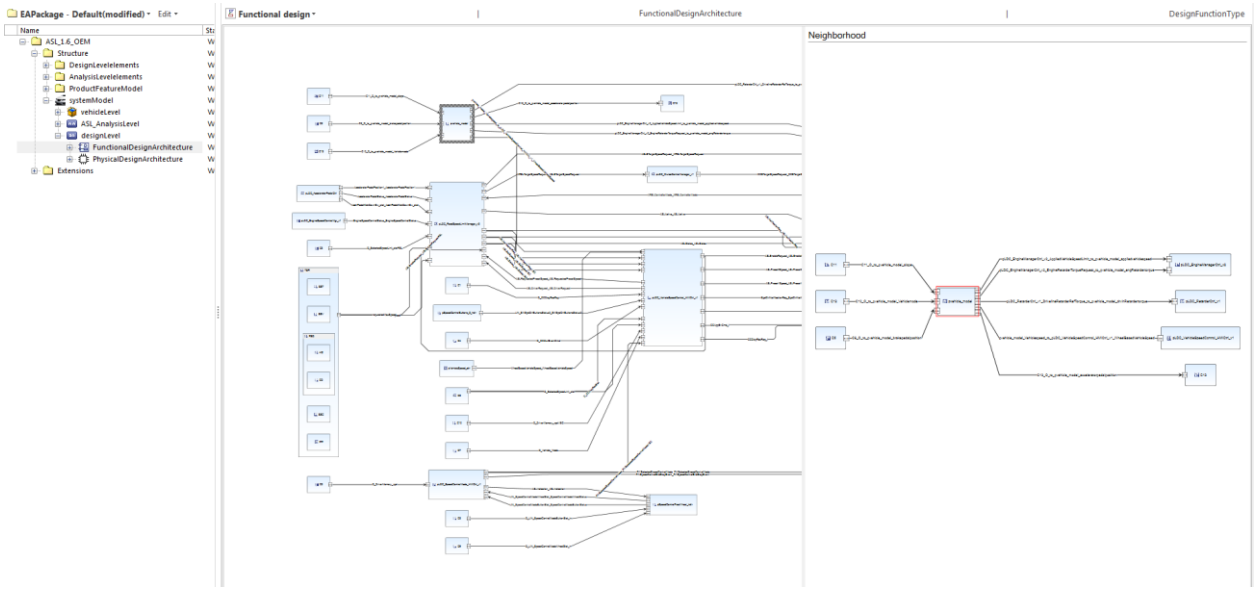


Figure 14 Functional design graph and neighborhood information

4.6 Graph export

The graphs in SystemWeaver can be exported to graphml and sgraphml. Graphml is a format widely used by yEd, which is a free graph tool. Sgraphml is a proprietary format inspired by graphml but customized to the needs of the Synligare project. The sgraphml file can be imported to EATop as seen in Figure 15.

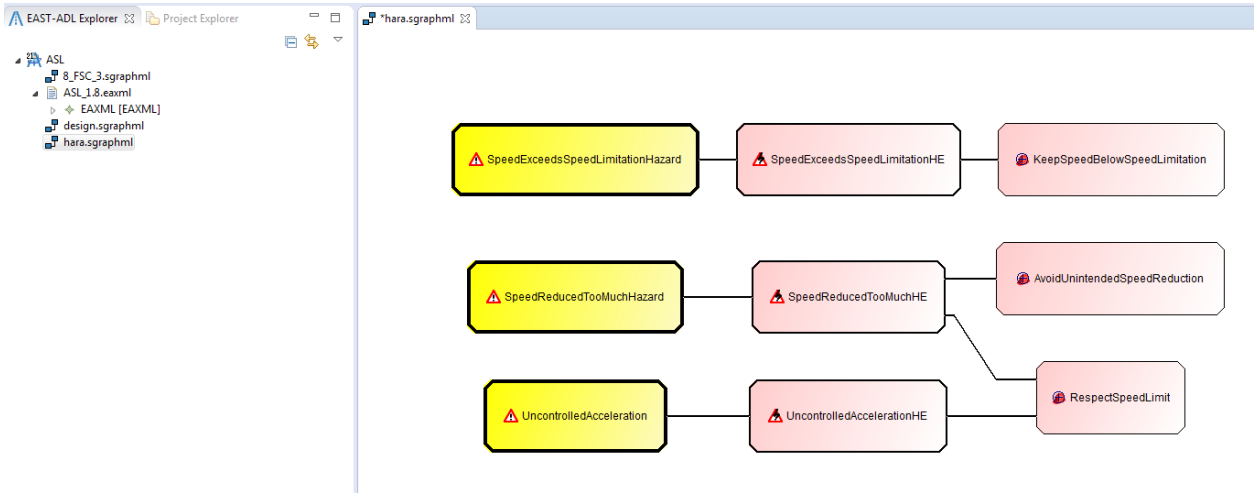


Figure 15 Graph exported from SystemWeaver, imported to EATop using the sgraphml format

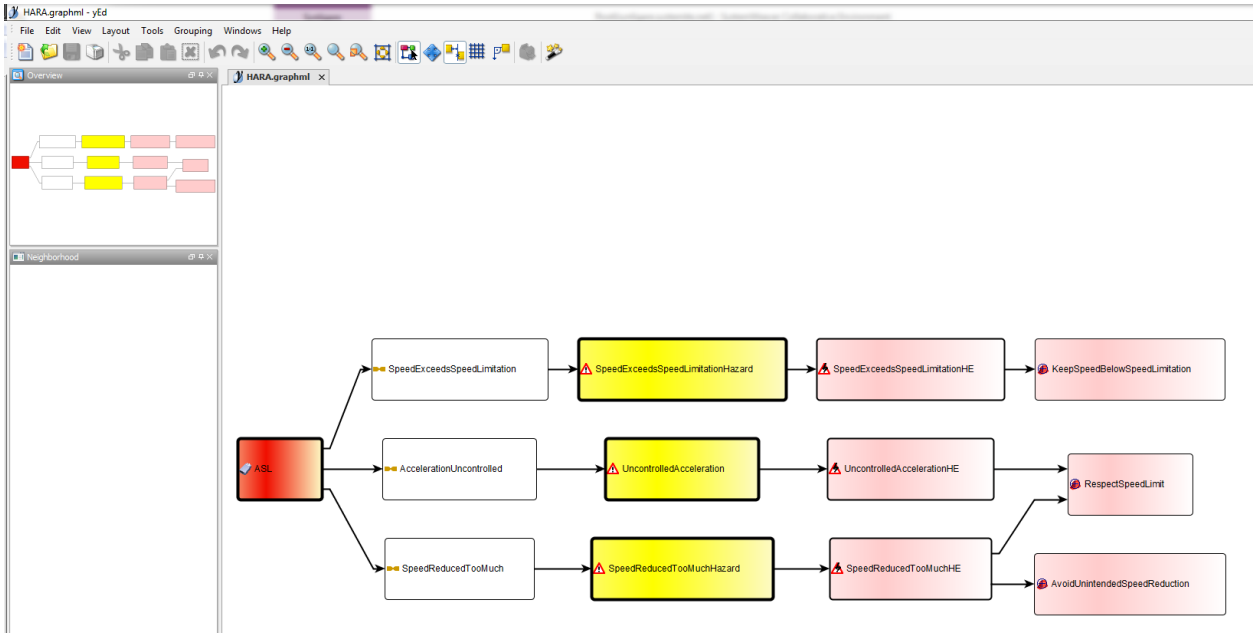


Figure 16 Graph exported from SystemWeaver, imported to yEd using the graphml format

5 EnterpriseArchitect

The Enterprise Architect UML tool is a widely used, low cost UML editor. With an EAST-ADL profile, it can also be used as an EAST-ADL editor.

5.1 Enterprise Architect EAXML Exchange

The Enterprise Architect EAST-ADL Exchange tool allows EAXML files to be imported and exported to Enterprise Architect.

5.1.1 Description

The Enterprise Architect EAST-ADL Exchange tool is implemented as a stand-alone application that reads and writes EAXML files. The Visual Basic API of Enterprise Architect is used to create and read elements in an EAST-ADL model in Enterprise Architect.

The import and export from EnterpriseArchitect has been prototyped in a first step.

6 Summary

This deliverable has documented the tooling prototypes developed in the Synligare project.

7 References

[1] EATOP Eclipse Open Source Project: EAST-ADL Tool platform. http://www.eclipse.org/eatop

[2] Sparx Systems Inc: Enterprise Architect. www.sparxsystems.com

[3] Synligare Consortium: Synligare Deliverable D1.1 Needs Identification. http://www.synligare.eu/

[4] Synligare Consortium: Synligare Deliverable D2.1 Modeling Concepts and Methods. http://www.synligare.eu/

[5] Systemite AB: SystemWeaver. http://www.systemite.se