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 D3.2

Report name — Tooling Prototypes

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
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.

This chapter introduces WP3 by defining purpose, scope and method.

2 EATOP Demonstrators and plugins

2.1 EATOP Platform

The EAST-ADL Tool Platform, EATOP, provides a reference implementation of EAST-ADL where tool plugins with various purposes can be plugged in. It is provided as an open source Eclipse project.

There are two demonstrators of the EATOP Platform available that are relevant here, the Synligare EATOP Demonstrator and the EATOP Platform Demonstrator. These are described in the next sections.

2.2 Synligare EATOP Demonstrator

The Synligare EATOP Demonstrator is developed within the Synligare project. It consists of the EATOP platform and the plugins developed within the Synligare project.

2.2.1 Installation instructions

The latest Synligare EATOP demonstrator is available in several versions targeting different Operative Systems and 32/64 bit processors. Download the suitable one from the following locations:

- www.synligare.eu/demonstrator/EATOP_Synligare_linux_x86.zip
- www.synligare.eu/demonstrator/EATOP_Synligare_linux_x86_64.zip
- www.synligare.eu/demonstrator/EATOP_Synligare_macosx_x86.zip
- www.synligare.eu/demonstrator/EATOP_Synligare_macosx_x86_64.zip
- www.synligare.eu/demonstrator/EATOP_Synligare_win32_x86.zip
- www.synligare.eu/demonstrator/EATOP_Synligare_win32_x86_64.zip

To install, simply unzip the archive to a local folder and then run eatop.exe to start.

2.2.2 Build Instructions

To build the Synligare EATOP demonstrator maven is needed, the max supported version to build itis version 3.0.5. This can be downloaded from https://maven.apache.org/download.cgi. The source code can be found on the develop branch of the github repository: https://github.com/Arccore/synligare/tree/develop. When maven is installed and the Synligare repository is cloned, navigate to the root folder of the repository and use the following command to trigger a build: “mvn clean install -f releng/org.eclipse.eatop.releng.builds/pom.xml -P platform-luna”. This assumes that maven is a part of your environment variables.

2.3 EATOP Synligare Integration Policy

In order to facilitate the development of tools needed for the scenarios Git is used as a version control system for the source code. For finished plugins the result should be delivered to SVN. The policy for branching, naming of plugins and integration into EATop is described in this section.

2.3.1 Core Branches for Git

Master – this is kept up to date with EATop master and is what we create our branches from when starting work on a new plugin or feature. Its state is managed by the admins of the repository which is currently Mikael Stolpe and Mattias Ekberg.

Develop – this is the merged branch of all our finished plugins and features. It is used to have a good branch to build the demonstrator from. It is also kept up to date with master to have all the latest features from EATop. Important notice about this branch is the target platform. As always with committing to a branch it is the responsibility of the committer to ensure the code is functioning correctly. However, due to the recent issues with the target platform further care should be taken to ensure this is functioning properly. It is important to test it by updating and setting it anew before doing a commit. This is done in order to ensure nothing is stored locally in the cache.

Beginning Work With New Features

This should be handled as described above by branching from the master branch. The usual policy is to categorize these kinds of branches into feature/<branch name>. When the plugin is finished and tested it is then merged into develop for the demonstrator to be built. Furthermore, if it should be integrated into EATop the process should be started for this as well. However, it might be good to have a testing period before this is done.

This method gives us some overhead with the merging but simplifies things in the end since we have a plan to merge many features to EATop. With this policy the step feature -> develop merge will possibly become problematic and annoying. This problem could be avoided if the branch creation was done from develop. However, that would make the merge into EATop a larger problem since we would have to reverse any changes which isn't relevant for that plugin

This is not set in stone of course, but initially this is a good way to get started and if it creates too much overhead we could open this discussion further down the line.

Continuing work on an already merged plugin

There might be need to continue developing or fix bugs after a feature have been merged into develop. When this happens one should create a branch which branches from the previous branch on the related plugin. If the branch is created from master the previous work would not be available. If it is created from develop there would be too much work to separate the changes done towards master when submitting the fix. By creating a new branch from the previous one it is also possible to review the work done before merging it into the previous branch and later develop. If a patch have already been submitted to EATop for this feature then a simple diff between the branches can be done and submitted as a bugfix or improvement.

Delivering finished plugins

When a plugin is considered finished and has been merged to develop the responsible developer should also deliver a built plugin or an update-site the SVN. This should be stored on the SVN at the path /Synligare_SVN/WP3/EATOP/EATOP_update-sites/.

2.3.2 Demonstrator

As mentioned above the develop branch will be used to build the demonstrator. This should not have to be done at a specific time such as weekly or nightly. The contributions this project will have will most likely not be frequent and small but larger and infrequent. Therefore, it should suffice to build a demonstrator when a new commit is made to develop. This could be checked every week and if changes has been made a new demonstrator is created for use within the project.

2.3.3 Integrating to EATop

This can be quite an intricate process and can therefore take time. For small commits and fixes this is usually a rather quick process, however for larger commits such as the Table View it requires the approval of more than one committer and it could therefore take several months to get approval. It is the first approval of a large feature or plugin which takes the longest time. After this initial commit it is much quicker to do bug fixes or changes. The decision was made that each company which will do a contribution to EATop should attempt to have a committer to the EATop project at Eclipse to simplify this process. It is also important since we need to keep the responsibility over the code to each company.

2.3.4 Naming of plugins

Currently, there are three ways of name plugins and the packages related to them.

1. For applications intended for open source and contribution for to EATop the naming should be org.eclipse.eatop.app.<company>.<appl> where <company> is the name of the company responsible and <appl> the name of the application being developed. When the plugin is finished and should be contributed to EATop the <company> part should be removed.
2. Infrastructural components which should be contributed to EATop should have the naming convention org.eclipse.eatop.<section>.<company>.<name>. Where <section> should be appropriate part of eatop it relates to. This could be empty if there is no related eatop part. <name> is the name of the component. As with 1. the company name should be removed when contributing to EATop
3. If there are applications which will never be contributed to EATop the name should be com.<company>.<appl>.

2.4 EATOP Demonstrator

The EATOP Platform Demonstrator is developed by the Eclipse EATOP project, and contains the plugins included in the EATOP platform.

2.4.1 Installation

The latest EATOP Platform Demonstrator is available at https://www.eclipse.org/eatop/download.php. There are different versions for 32 and 64 bit Windows, but no versions targeting other Operative Systems. Download the desired version, unzip the archive and execute eatop.exe to start. This demonstrator does not yet include any Synligare plugins.

2.4.2 Build Instructions

In order to build the EATOP Platform Demonstrator from source, follow the instructions at http://wiki.eclipse.org/EATOP/environment.

2.5 Update sites and binaries

There are two categories of plugins that have been developed within the Synligare project. The first category contains the plugins in sections 2.6 - 2.9 that have already been approved to become part of the EATOP Platform. Therefore, there are no update sites available to avoid having more than one location where these plugins will evolve.

The second category contains plugins that are candidates for being added to the EATOP platform in the future. For these plugins there are update sites available so they can be installed in the EATOP Platform demonstrator.

All plugins developed within the Synligare project are part of the Synligare EATOP Demonstrator.

2.6 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.

2.6.1 Description

It is accessed by selecting the two model objects that should be compared, then pressing right click and selecting Compare With -> Each Other <EObject> (in Dialog) or (in Editor). This will open either a dialog window or editor to work with. It also has functionalities in place to merge copy objects from one side to the other. These are accessed from the upper-right tool-bar in the editor.

2.6.2 Installation Instructions

The functionality is included in the Synligare EATOP Demonstrator and installed according to Section 2.2.1. It will be included in the EATOP Platform Demonstrator in the near future.

If EATOP is already installed and this plugin needs to be added separately:

Open Help-Install New Software

Type “www.synligare.eu/files/sites/org.eclipse.eatop.” in the Work-With entry

Select components and complete installation

2.7 Connector Creator

A common use case when working with EAST-ADL models is to create connectors. This usually requires the creation of a connector, two child objects and setting the reference of both of these to the respective ports. When working with larger projects this can become quite a tedious process. The Connector Creator is a plug-in which attempts to simplify this process.

2.7.1 Description

The Connector Creator works by providing a dialog which lists all the available ports inside the prototype, when a port has been selected the dialog filters and displays the possible ports this port can be connected to in another list. After the user has selected one port from each side the plug-in can create a connector between the ports. It also provides functionality to delete a connection, if one is present. Since having two identical connections would be superfluous the option to create a connection is replaced by delete connection if two ports are already connected.

The Connector Creator is accessed by right-clicking the prototype and pressing “Create connection”, after which the dialog is opened. If no dialog can be opened due to no ports being available the user will be prompted about this.

2.7.2 Installation Instructions

The functionality is included in the Synligare EATOP Demonstrator and installed according to Section 2.2.1. It will be included in the EATOP Platform Demonstrator in the near future.

If EATOP is already installed and this plugin needs to be added separately:

Open Help-Install New Software

Type “www.synligare.eu/files/sites/org.eclipse.eatop.” in the Work-With entry

Select components and complete installation

2.8 Table View

Each model object holds a number of properties which define them and comparing these between two or even more objects of the same type are a common use case. This can become particularly tedious when working with the default properties view which only displays the properties of one selected object. Furthermore, there might be a need to change the same value of several objects. The Table View hopes to remove some of the time required whilst performing these actions.

2.8.1 Description

The Table View lists the object or objects selected in the EAST-ADL explorer in a table with the properties of these objects as the columns of the table. Where it is applicable it is also possible to edit and multi-edit the properties of the objects. Certain properties which requires adding child objects to the model object are not editable. It also displays certain non-default properties such as incoming references.

The Table View can be opened by clicking Window -> Show View -> Other -> General -> Table View.

2.8.2 Installation Instructions

The functionality is included in the Synligare EATOP Demonstrator and installed according to Section 2.2.1. It will be included in the EATOP Platform Demonstrator in the near future.

If EATOP is already installed and this plugin needs to be added separately:

Open Help-Install New Software

Type “www.synligare.eu/files/sites/org.eclipse.eatop.” in the Work-With entry

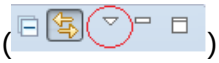
Select components and complete installation

2.9 Tree View Enhancements

The view EAST-ADL Explorer is used to display the model in a tree-view manner. Larger models often becomes 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 are therefore a need to enhance the view.

2.9.1 Description

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. This functionality can be accessed via the small down arrow in the top-right corner of the EAST-ADL Explorer () then selecting Model Element Appearance -> Categorize model elements.
- Context view: It is opened by selecting Window -> Show View -> Other -> Eatop Example Views -> Explorer Context. 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.

2.9.2 Installation Instructions

The functionality is included in the Synligare EATOP Demonstrator and installed according to Section 2.2.1. It will be included in the EATOP Platform Demonstrator in the near future.

It cannot be installed separately.

2.10 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.

2.10.1 Description

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

- Select Error Model
- Right Click and select Fault Tree Analysis

2.10.2 Installation instructions

The feature is included in the Synligare EATOP Demonstrator. For the EATOP Platform Demonstrator, install the feature by following the following procedure:

If EATOP is already installed and this plugin needs to be added separately:

Open Help-Install New Software

Type “www.synligare.eu/files/sites/org.eclipse.eatop.volvo.hiphopsbridge.updatesite” in the Work-With entry

Select components and complete installation

2.11 Version Manager

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

2.11.1 Description

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.

Operation:

- Initialize version by right clicking top level of model
- Increment version of individual element by right clicking and selecting +0.1 or +1
- Set target version to current version of target element by right clicking and selecting Synchronize target version
- Assess version consistency by selecting Error Model, right click and select "Check version consistency"

2.11.2 Installation instructions

The feature is included in the Synligare EATOP Demonstrator. For the EATOP Platform Demonstrator, install the feature by following the following procedure:

If EATOP is already installed and this plugin needs to be added separately:

Open Help-Install New Software

Type "www.synligare.eu/files/sites/org.eclipse.eatop.volvo.versionmanager.updatesite" in the Work-With entry

Select components and complete installation

2.12 Error Model Generator

The Error Model Generator is a plugin which supports model synthesis of error propagation models.

2.12.1 Description

Error propagation modeling is the basis for failure modes and effects analysis (FMEA) and fault tree analysis (FTA). This plugin automatically generates the error propagation model from a nominal model and supports common model manipulations.

Operation:

- Select functional element
- Right click and select "Generate ErrorModel"

2.12.2 Installation instructions

The feature is included in the Synligare EATOP Demonstrator. For the EATOP Platform Demonstrator, install the feature by following the following procedure:

If EATOP is already installed and this plugin needs to be added separately:

Open Help-Install New Software

Type “www.synligare.eu/files/sites/org.eclipse.eatop.volvo.errormodelgenerator.updatesite” in the Work-With entry

Select components and complete installation

2.13 Metrics

The Metrics plugin supports computation and visualization of model metrics.

2.13.1 Description

The following metrics are supported:

- Allocated Requirements
- Verified Requirements
- Realized Features
- Safety Goals covered by Functional Safety Concepts
- Functional Safety Concepts covered by Technical Safety Concepts

The plugin presents the metric result as a circle diagram in a separate view for each metric.

Operation:

- Open the view by either
    - selecting window-Show view – Metrics – the desired metric, or
    - click on right mouse button in the explorer view, and select Metrics – the desired metric
- Click in the explorer tree to select which part of the model the metric shall be calculated for

2.13.2 Installation instructions

The feature is included in the Synligare EATOP Demonstrator. For the EATOP Platform Demonstrator, install the feature by following the following procedure:

If EATOP is already installed and this plugin needs to be added separately:

Open Help-Install New Software

Type “www.synligare.eu/files/sites/org.eclipse.eatop.volvo.metrics.updatesite” in the Work-With entry

Select the Metrics feature and complete the installation.

2.14 Model Overview

The Model Overview plugin supplies a graphical overview of an EAST-ADL model.

2.14.1 Description

The Model Overview plugin contributes a view that shows a graphical overview of the currently selected model. The overview shows elements in 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.

Operation:

- Open view by selecting window-Show view – Other – Visual Model Overview
- Click on any model element in the treeview to view the model overview
- Click on the toggle button “Lock view” in the view toolbar to freeze updates upon further model element selection

2.14.2 Installation instructions

The feature is included in the Synligare EATOP Demonstrator. For the EATOP Platform Demonstrator, install the feature by following the following procedure:

If EATOP is already installed and this plugin needs to be added separately:

Open Help-Install New Software

Type “www.synligare.eu/files/sites/org.eclipse.eatop.volvo.visualizer.updatesite” in the Work-With entry

Select components and complete installation.

2.15 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.

2.15.1 Description

EAST-ADL uses Generic Constraint to annotate properties, and each constraint defines its kind and value and identifies which in which modes 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.

Operation:

- Select desired top element
- Right click and select Summation Analyzer-Analyze
- (If multiple properties ar annotated): Select desired GenericConstraint kind

- (If modes are applicable): Select desired mode

2.15.2 Installation instructions

The feature is included in the Synligare EATOP Demonstrator. For the EATOP Platform Demonstrator, install the feature by following the following procedure:

If EATOP is already installed and this plugin needs to be added separately:

Open Help-Install New Software

Type "www.synligare.eu/files/sites/org.eclipse.eatop.volvo.linearpropertyanalyzer.updatesite" in the Work-With entry

Select components and complete installation

2.16 SGraphML Graphical Editor

The EATOP model is primarily edited and browsed as a tree structure. This 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.

2.16.1 Description

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.

Operation:

- Select in the menu File-New EAST-ADL Diagram
- Drag and drop any element onto the diagram
- Drag and drop types with contained prototypes – All prototypes and connectors will be added to the diagram inside the periphery of the type.
- Select and move shapes in the diagram
- Drag and drop elements contained in existing elements
    - On top of the existing shape – The new element will reside inside the existing shape.
    - Beside the existing shape – This results in a containment relation being shown between the containing and contained shape
- Right click a shape in the diagram and select goto element

- Delete shapes in the diagram – this deletes from the diagram but not from the model

2.16.2 Installation instructions

The feature is included in the Synligare EATOP Demonstrator.

If EATOP is already installed and this plugin needs to be added separately:

Open Help-Install New Software

Type “www.synligare.eu/files/sites/org.eclipse.eatop.volvo.sgraphml.gefeditor.updatesite” in the Work-With entry

Select components and complete installation

2.17 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.

2.17.1 Description

There is an interface consisting of primitive functions used to input a diagram consisting of nodes, edges and sub-diagrams into the place-and-route package. Similarly, there are a set of functions that correspond to the place-and-route algorithms. Further, another set of primitive functions can be employed to read back coordinates for the nodes and edges of the diagram.

2.17.2 Operation

Typically, the operation of the primitive functions and the calling of the algorithms is hidden from a user by integration into a plugin. For example, in SGraphML Graphical Editor, a diagram (or a subset of a diagram) can be placed-and-routed by (selecting a subset of a diagram and) right-clicking the canvas of the SGraphML Graphical Editor and selecting “Arrange Layout” in the appearing menu. After a while the diagram is updated with the new coordinates and geometries of the nodes and the new routing of the edges.

2.17.3 Installation instructions

The feature is included in the Synligare EATOP Demonstrator as part of the SGraphML editor. For the EATOP Platform Demonstrator, the SGraphML editor can be installed separately. See Section 2.16.2 for installation instructions.

2.18 Requirements Allocation Assistant Editor

This plugin enables the allocation of requirements to model elements located in the same EAST-ADL model file or in two separate files. It provides searching and filtering functionality as well as a hint functionality that shows suggestions of model elements for a selected requirement based on allocation connections existent on higher abstraction levels.

2.18.1 Description

The Requirements Allocation Assistant is an editor that helps performing allocations of requirements to model elements located in the same file or in two different model files. The editor displays two tree views, the one to the left for requirements and the one to the right for model elements, and provides searching and allocation functionalities.

Operation:

- Right click on the file in the explorer containing the model and then choose Open With > Allocation assistant
- Drag and drop the file containing the requirements model (or the same file used at the previous point, if that one contains also the requirements) on the gray rectangle displayed on the Allocation Assistant editor.
- Click on Search and filter – search for requirements or model elements by element type and element attribute value.
- Select a requirement in the left tree view and click on Hint button – this will search for requirements on a higher abstraction level that were allocated to model elements and based on these findings will show suggestions of model elements that the selected requirement could be allocated to.
- Select one or mode requirements on the left side and one or more model elements on the right side and click on Allocate.
- Click on Save in the eclipse toolbar (or press Ctrl+S) to save the changes in the models before closing.

2.18.2 Installation instructions

The feature is included in the Synligare EATOP Demonstrator. For the EATOP Platform Demonstrator, install the feature by following the following procedure:

If EATOP is already installed and this plugin needs to be added separately:

Open Help-Install New Software

Type “http://www.synligare.eu/sites/org.eclipse.eatop.semcon.allocationassistant.updatesite” in the Work-With entry

Select components and complete installation

3 SystemWeaver

3.1.1 Description

SystemWeaver is a commercial tool developed and marketed by Systemite AB.

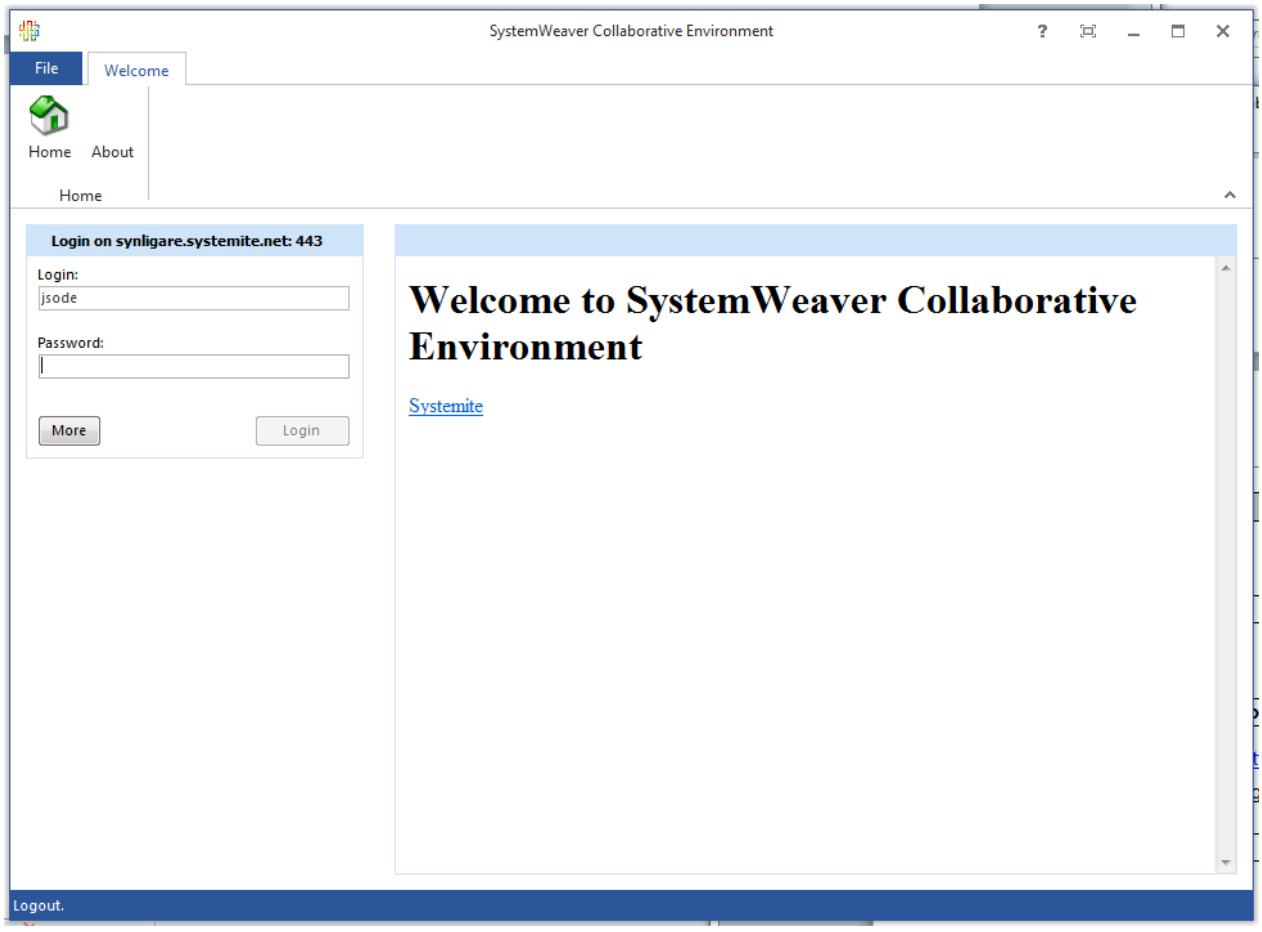
SystemWeaver is in use at various automotive companies, including Volvo Car Corporation and AB Volvo.

More information on SystemWeaver can be found at www.systemite.se.

3.1.2 Installation instructions

The SystemWeaver tool client can be installed from http://apps.systemite.se/Synligare/setup.exe.

Accounts at the synligare repository is available only to consortium members. The address to be used for the login is synligare.systemite.net, with port = 443:



3.2 Modelling Support

The modeling support includes the EAST-ADL core meta model and the Requirements and Dependability extension packages.

In the Synligare project the support for EAST-ADL has been extended, and now supports allocation of requirements to DesignFunctionPrototypes.

3.3 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.

3.4 Views

As discussed in D3.1 SystemWeaver provides configurable views. These views can be represented in form of reports, grids, and graphs. For example view D3.1

4 EnterpriseArchitect EAST-ADL Exchange

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.

4.1.1 Description

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

- EAXMLImport

  The tool can import and sync an EAXML-file into a package structure. Full round-trip is not supported, and only a subset of EAST-ADL is supported.

  Tag values are used to control import and export:

  - Packages can be stereotyped with <<EA_UPDATED>>

    These packages will not be imported during a roundtrip. But it will be exported. Used for shared model parts

  - Packages can be stereotyped with <<EA_SKIP>>

    These packages will not be imported during a roundtrip and it will not be exported. Used for skipped model parts and local (not shared) model parts

  Prior to import, the top level package is selected. The name of the package shall match the filename of the .eaxml-file. The file is put into C:\Temp directory

- EAXMLExport

  The tool can export model to an EAXML-file Only parts of EAST-ADL is supported

  Prior to import, the top level package is selected. The name of the package shall match the filename of the .eaxml-file.

- Tool AddPortClassifier

  Tool is used to fix ports added in EA with correct classifiers.

4.1.2 Installation instructions

The Enterprise Architect EAST-ADL Exchange tool is copied to a local directory and run as a Windows application. This tool is available from www.synligare.eu.

Trial versions and licenses for the Enterprise Architect tool can be acquired from www.sparxsystems.eu.

- Copy tools and templates to a local directory
  - AddPortClassifier.exe
  - EAConnector.dll
  - eastadl_2-1-12.xsd
  - EAXMLExport.exe
  - EAXMLImport.exe
- Create Folder C:/Temp

- Extend tools menu
    - Tools>Customize, Chose Tools tab
    - Click Add button
    - Name: tool name
    - Add path to “tool.exe”
    - Initial directory: C:\Temp
    - Add the following tools to the menu in the same way:
        - EAXMLImport
        - EAXMLExport
        - AddPortClassifier

5 Summary

This deliverable has described tooling prototypes used to assess and validate tooling concepts developed in the Synligare project.

6 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] Synligare Consortium: Synligare Deliverable D3.1 Report on Tooling. http://www.synligare.eu/

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

[7] HiP-HOPS: www.hip-hops.eu