TOPCASED Requirement: a model-driven, open-source and generic solution to manage requirement traceability

A. Raphaël FAUDOU[1], B. Tristan FAURE[1], C. Sébastien GABEL[2], D. Christophe MERTZ[2]

1: ATOS ORIGIN INTEGRATION, 6 impasse Alice Guy, BP 43045, 31024 Toulouse Cedex 03, FRANCE
2: CS Systèmes d'Information, Rue Brindejonc des Moulinais, BP 15872, 31506 Toulouse Cedex 05, FRANCE

Abstract: Model based engineering has proven maturity and efficiency in industry but deliverable and certification constraints have not yet been completely adapted to this new approach. The consequence is that people use more and more models to capture and refine requirements but still have to manage textual requirements in parallel and ensure traceability between those requirements.

The TOPCASED project (http://www.topcased.org) has started to address this issue by providing two features about requirement management: import textual requirements into models and create easily textual requirements from model elements. This paper aims to give an overview of the current state of those features. At first, the requirement import process provides a simple solution to extract requirements from documents and inject them into requirement models. Once imported, traceability links can be created between high level textual requirements and model elements as simply as a "drag and drop" operation. So, the system automatically creates a new low level requirement linked to the upper requirement and referencing a semantic model element. The TOPCASED requirement system suggests a management solution composed by a set of preference pages, two complementary views for handling requirements ("upstream" for imported read-only requirements and "current" for refined requirements) and a manager helping to upgrade version of an upstream document. This paper will cover in detail each item described above including the document import process. At the end we will give feedback from operational projects and will present the future directions.

Keywords: TOPCASED, Eclipse, modeling, MDD, requirement, open-source.

1. Introduction

Requirements are key artefacts in avionics development. In a full model-based process requirements can be formalized as model elements or groups of model elements. But there was some delay before industry could switch to such a process, meaning that most of requirements would remain in a "textual" statement during the transition phase. So the tools supporting this process should allow creating and managing links between model elements and "textual" requirements.

Until TOPCASED [1] 2.1.0 release, there was no support for such a feature in the platform.

Latest avionics program gave the opportunity to integrate this missing component into the platform. The first study case was launched in 2008 on SAM [2] modeling language. The process became mature in 2009 and inspired other avionics programs with similar needs. This success story led to new developments to generalize the approach taken with SAM on all other modeling languages implemented in the TOPCASED platform.

This paper presents the requirements engineering process used on avionics program and gives a quick overview of the complete tool chain supporting requirement management. Finally, we will focus on industrial deployments and future enhancements.

2. Needs

2.1 Requirement definition and terminology

Requirements engineering covers the specification and design activities throughout the product development cycle. The understanding of the customer needs is translated into requirements. On each intermediate level of product development, it is verified that the activities satisfy the requirements on the appropriate level, finally demonstrating that the product delivered to the customer meets his needs.

A requirement is a statement that defines one or more constraints on one or more blocks of the system to develop, or one or more characteristics of this (these) block(s) and is identified by a label. It must be possible to validate a requirement and verify its implementation.

Refined requirements – also called current requirements all over this paper – are downstream requirements that specify one upstream requirement.

2.2 Approach definition

The input of the work described below was the requirement model produced by the Reqtify tool export facilities: a model containing a set of upstream requirements.

The Reqtify software, developed and commercialized by the company Gensys, is a

solution for managing requirements traceability and impact analysis across several documents.

The component TRAMway, especially designed for the TOPCASED platform is a light open-source adaptation of the Reqtify commercial solution. TRAMway component provides a simple traceability solution working on several documents reflecting the project lifecycle (specification, design, validation plan, test plan, etc.). An Open Office document parser is provided with this tool.

As the TRAMway meta-model was not complete enough to fully describe the requirement information needed by avionics software specification document, it was extended with essential concepts such as requirement refinement, allocated, untraced, and unaffected relationships, problematic requirements and even requirement project configuration. With this extension, a “Refined requirement” references two different elements: one model element (created from a modeling tool) and one upstream requirement (imported from TRAMway tool). We could then link model and textual requirements while keeping requirement traceability between upstream and “current” requirements which is requested when developing under avionics certification constraints.

Those complements to the initial TRAMway component and their generalization to all modeling languages gave birth to the TOPCASED-REQ component.

3. Context

3.1 Organization

TOPCASED-REQ is now included in the TOPCASED platform distribution. This component is the result of CS and ATOS ORIGIN work in an open innovation approach. CS developed and maintained the first contribution oriented to SAM models and ATOS ORIGIN generalized the approach so that it could be applied on any modeling language implemented in the TOPCASED platform (amongst them UML/SysML and AUI). Those both tasks were complementary.

3.2 Platform

TOPCASED is an open-source software platform primarily dedicated to the development of critical embedded systems or software with a model based engineering process.

TOPCASED platform is built around Eclipse technologies and especially on Eclipse modeling stack, a set of frameworks that facilitate the management of models (load, edit, render, query, split, compare, validate, transform, generate, import, export…). TOPCASED provides several components that manage or consume models.

3.2.1 Modeling editors

TOPCASED provides a framework able to generate a modeling editor from the meta-model. Some of the editors generated from this framework are included in the standard TOPCASED platform distribution.

UML editor: helps users to create UML models through diagrams compliant with the OMG UML specification. UML editor provides these following diagrams: Use Case, Class, Profile, Activity, Component, Composite Structure, State Machine, Sequence, and Deployment.

SysML editor: provides functionalities from UML editor and specific functionalities for SysML diagrams like Requirement, Block Definition, Internal Block, and Parametric. The TOPCASED SysML editor uses a SysML meta-model derived from the OMG SysML Profile.

SAM editor: avionic-oriented language derived from SART, it allows the graphical design of functional structured analysis through two kinds of diagram: System and Automaton.

3.2.2 Simulation

TOPCASED simulation tool provides a framework and toolkit to animate models from UML/SysML and SAM meta-models. The framework has been defined as generic as possible and thus provides API to plug the simulator to another meta-model. The toolkit also provides tools to simulate models.

It is possible for a simulation to define a scenario that can then be run in command line or interactively. In that case, the user can play the scenario step by step and select any of the different possible events available for the current state. According to the chosen event, it can trigger one or another transition and reach a new state. Simulation can also randomly be executed.

3.2.3 Code Generation

TOPCASED is bundled with several “Model to Text” (M2T) transformation engines technologies like Acceleo or XPand which allow users to define custom generators.

It is also bundled with predefined generators:

- UML2C generating C code from a UML Model
- UML2Java generating Java code from UML Model
- UML2RTSJ generating real time Java from UML Model
- SMUC generating Ada, Java or C code following the “state” design pattern from a UML model containing a State Machine

3.2.4 Reverse engineering from code

TOPCASED provides features to create UML models from a Java plugin (Java2UML) and from Java Archive (JAR).

3.2.5 Documentation generator

TOPCASED provides a framework that can retrieve model data or diagrams and insert that information at given places in a document.

4. Global Approach

The global approach of TOPCASED-REQ is decomposed in three distinct parts:

- generating a requirement file starting from a document file (xlsx, docx, xls, doc, csv) or from a Reqtify export,
- attaching the requirement file to a TOPCASED model and creates requirement traceability links between model elements and upstream requirements,
- exporting requirement traceability across generated documentation or traceability matrix.

5. Importing requirements from Reqtify and doc2model approach

5.1 Reqtify import and export

As previously evoked, the differences between the TRAMway and TOPCASED-REQ requirement meta-model (also called “Requirement meta-model” in this document) are relatively weak.

The model transformation from TRAMway to Requirement and its opposite appeared to be easy since Requirement is inherited from TRAMway meta -model. That’s why, we wrote two ATL [3] model transformations which are respectively: Requirement to TRAMway (import) and TRAMway to Requirement (export) .

The TRAMway to Requirement transformation has been integrated into a user-friendly interface (Cf. Figure 1) and takes place into the SAM Requirement process notably at the beginning, when a new requirement model needs to be created and attached to an existing SAM resource.

The user needs to provide the access path to its semantic model (a SAM model in the example) and the access path to an XML resource (exported from Reqtify) containing upstream requirements. Other optional information can be provided at this stage.

During an update operation, the model transformation described above also occurs (Cf. §6.4).

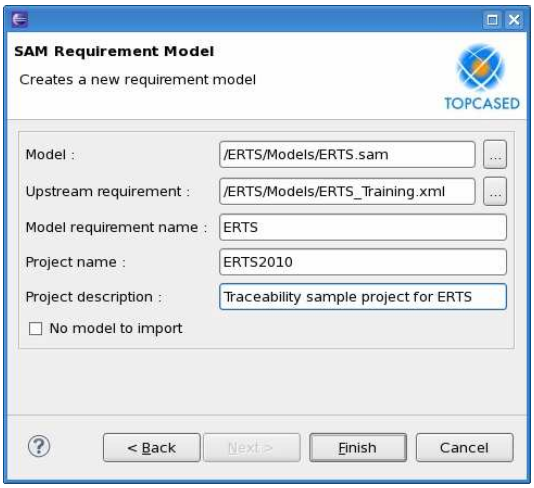

Figure 1: Requirement wizard creation

5.2 Doc2model approach

Doc2Model is a TOPCASED technology now contributed to Eclipse. It is a generic parser of documents which creates an EMF Model from a configuration file.

The technology recognizes regular expressions and styles for text documents. It recognizes also column number for spreadsheet documents. In addition, the technology provides a Java API to parse different types of document.

5.3 TOPCASED requirement import

The “TOPCASED Requirement import” wizard is a front-end interface that provides the user a way to import its requirements from any document. This wizard is built upon doc2model framework. The wizard allows a user to define dynamically a doc2model configuration file without knowing the doc2model language.

The user can:

- define styles to recognize requirements and requirement attributes. The styles are automatically imported from the document, and the selection is easy for the user.
- define regular expression for requirements identification and requirement attribute identification. Current version of Doc2Model browses elements paragraph after paragraph. So a regular expression is analysed in each paragraph.
- To help the user in defining regular expressions, a widget is available in TOPCASED-IMPORT wizard. This widget allows the user to check if a regular expression matches a specified string. The widget can also display the result of grouping expressions.

- define a column number: Doc2Model engine creates elements when a column number is matched.

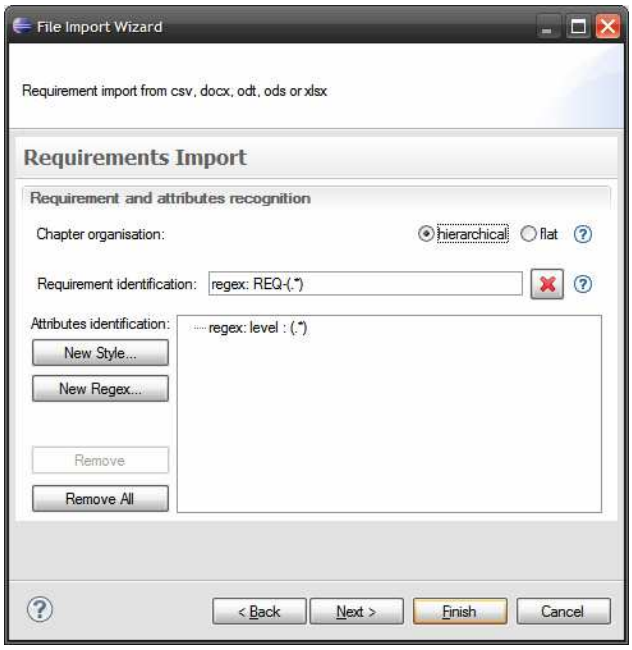

Figure 2: Requirement recognition definition

At requirement identification step, the user can choose if the document is recognized in a hierarchical or flat way. If the hierarchical way is chosen, the output requirement model will be structured according to the structure of the document, i.e. the heading styles from the input document define a hierarchical structure.

The requirement meta-model provides to user the possibility to define custom attributes with specific keys and values. The Requirement import allows users to map the recognized elements with defined attributes.

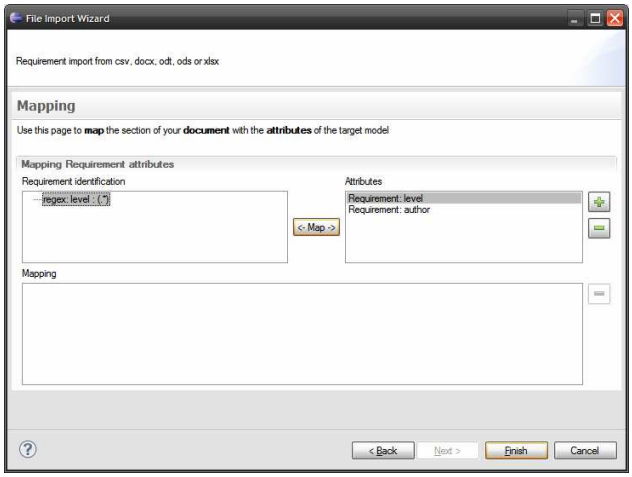

Figure 3: mapping between recognition and requirement attributes

At the end of the process, a requirement file is generated and can be attached to a model.

To use requirement traceability in TOPCASED, it is necessary to generate a Requirement model file but the TOPCASED requirement import can also generate UML and SysML models, with the possibility to apply stereotypes of a specified profile. For UML the generated elements in the models are Classes and for SysML the elements are Requirements.

5.4 Requirements filter

There are two ways to get a requirement file. It can be useful for the user to customize its requirement model before attaching it to a given SAM or UML/SysML model and create traceability links. TOPCASED provides a feature to filter requirements that can analyse the requirement attributes and ids. The user has the possibility to define a regular expression that will filter the requirements he wants to keep.

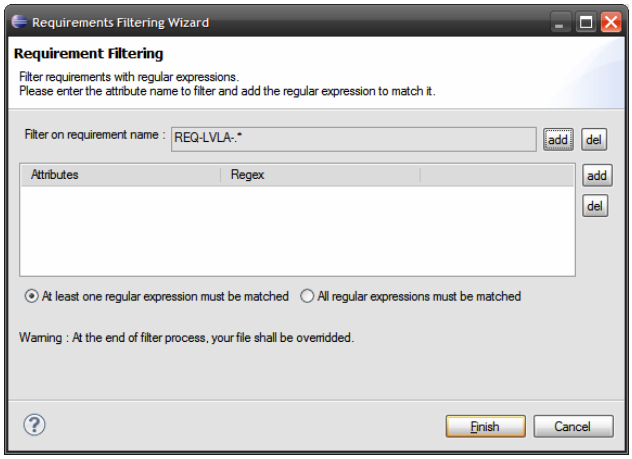

Figure 4: option to filter imported requirements

6. Requirement Traceability in modeling process

The implemented solution comes in addition to any modeling editor. Our initial objective was here to make possible coverage of model elements graphically represented, notably those supported by the SAM [3] Modeler in a first time, then those coming from other Modelers (UML, SysML, AUI, etc.).

6.1 Requirement project configuration

Although a default configuration exists for current requirements, the user can customize some behaviour such as the attribute list or the naming pattern. These operations can be performed according to two levels (workspace or project) and must be done before starting a new traceability project.

At any time during the modeling process, it is possible to add, remove, replace or alter the order of attributes; the requirement model is then updated according to those changes.

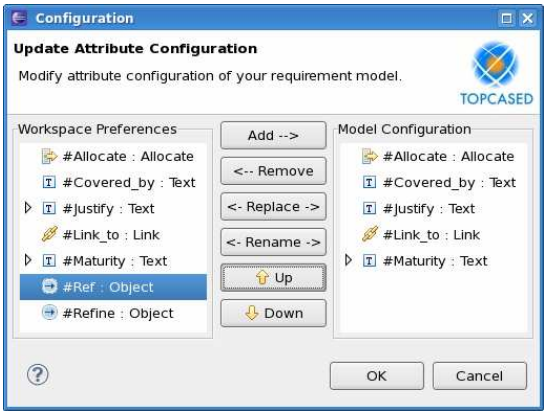

Figure 5 : Attribute modification dialog

6.2 Views

Two views split the information contained by the requirement model into two distinct parts.

- The upstream view: when a semantic model linked to a requirement model is opened into its modeling editor, the content of the upstream view shows the upstream part imported in read-only. This tree hierarchy is structured in documents containing sections in which upstream requirements are stored. This point of view reflects the original document.

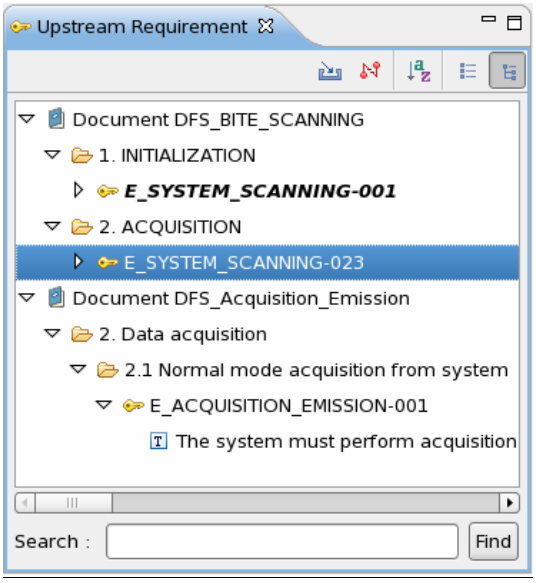

Figure 6 The upstream requirement view

- The current view: this view is the most important point of the application. Current requirements are organized according to their hierarchical container. These hierarchical containers provide a concrete link to a model element; that's why we have the feeling to visualize a real overview of the model structure.

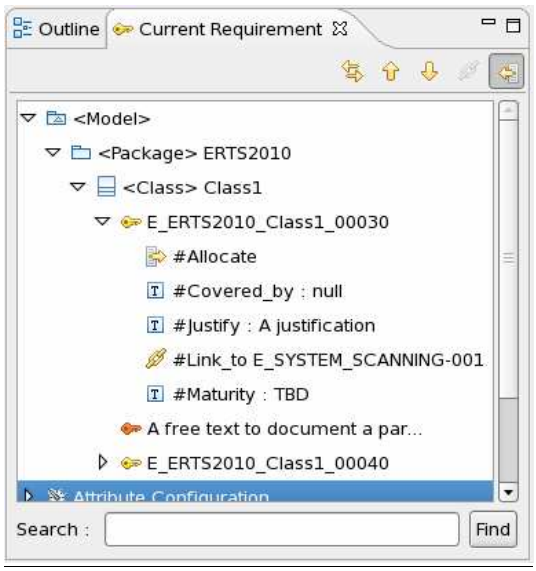

Figure 7 The current requirement view

6.3 Interactions with the modeling editor

Interactions by drag and drop with the modeling editor allow the end user to enrich or organize its requirement model. Creation operations are a key point since they make both a relation between upstream and current requirements via dedicated attributes, but also between a current requirement and a semantic model element via their hierarchical container.

From upstream view to Modeler: this operation has for effect to create a new current requirement and to attach it to the corresponding hierarchical container. Depending on the project configuration, one or several attributes of the current requirement reference the upstream requirement.

From upstream view to current view: if the drop target is a hierarchical container, a new current requirement is created inside. If the drop target is a current requirement, attributes of kind AttributeLink are added to this requirement and reference the upstream requirement at the origin of the drop operation.

Move operations play also an essential role since they allow arranging information aimed to be injected into generated documents, as it is already done for producing specification documents in an industrial context.

Current view to Modeling editor: this operation is used to change the hierarchical container of one or several current requirements, the drop target being the new container to set. The current view, listening to requirement model changes, is refreshed.

Interactions inside Current view: drag and drop operations are supported in order to alter the order of current requirements inside or outside their respective containers.

6.4 Document update and impact analysis

An interesting feature, based on “EMF Compare” framework, enables to upgrade the traceability to a new version of an upstream document. Two requirement models (the new and the previous) are compared and the impacts on traceability are computed. Only impacted upstream requirements that are referenced at least one time by a current requirement are considered and processed. Impacts are logged into the problem view and current requirements are marked as impacted. The operator will have to check manually and validate those impacts before going further in its job; requirement model is blocked at this stage in a read-only state.

6.5 Collaborative work

The requirement management system is compliant with team work. Users can use TOPCASED control command to split a model file into several sub models file resources. That enables collaborative work on the same global model through different resources that can each be put in version control system.

7. Export capabilities

Documentation generation: During the traceability process, it is important to have a visibility of the covered elements.

At any time a ratio percentage of coverage is visible for end user in TOPCASED status bar, but it does not provide a complete view of elements.

TOPCASED provides a documentation generation tool able to inject in template document (docx, odt) content from models: Gendoc.

To use Gendoc it is necessary to import the host document and to create a model representation of the document.

Once the document is imported, it is possible to link the model with the different data sources and to generate the final document.

To define dynamic parts in the document, the user inserts “generated fragments” which reference a predefined template. These templates are M2T templates (Acceleo or XPand) generating Docbook files.

To manage the Requirement traceability capabilities of TOPCASED, a set of templates is provided in the Templates library project. These templates enable to display, for instance, a table with requirements linked to a given element.

8. Industrial application

There are currently some industrial applications of TOPCASED-REQ component: specification teams in avionics industry have integrated the tool in their process to manage requirements between their UML or SAM specification models and their upstream requirements.

The goals of these teams using TOPCASED-REQ are both providing documentation to the design teams and also satisfying quality insurance and certification constraints. Generated documents demonstrate how the upstream requirements are covered by model elements.

Currently TOPCASED-REQ component is used by teams of a dozen of persons for UML and around twenty for SAM. Documentation generation has proven its efficiency through generation of several hundred pages that conform to firm quality process (almost 500 pages long for both UML and SAM).

9. Future orientations

The original need was to offer a system able to easily perform a full traceability between textual requirements and SAM model elements to support a given requirement process. The requirement management system had to be non intrusive on existing platform (TOPCASED) but also extensible for future developments.

Other experimentations carried out later on UML have proven the feasibility and the extensibility of this approach to other languages and process : requirement are structured on the same way, data are presented using the same views, traceability links are done the same way; so main functionalities could then be adapted to the UML scope.

Our current developments follow two main axes:

- Ensure a real degree of genericity regardless of the used modeling language. This task consists in merging the two contributions in order to provide a unique solution compliant with all TOPCASED modeling editors (existing or to come). Initial solution implemented for SAM is too specialized with respect to avionic needs and the global architecture is also closely dependant of the SAM modeling editor. Solution implemented for UML (and other languages), from the initial SAM approach brings new fresh ideas but still requires some efforts to reach the same level of maturity even if major features have been applied.
- Adapt the architecture for a wider use, notably for languages already defining requirement notions (e.g SysML). Until now, the process uses a triplet of models: the DSL model, the graphical model (diagrams) and the requirement model. This solution cannot be applied for

SysML models where the notion of requirement model does not make sense as SysML already provides requirement concept. So the objective is to keep the same principle without having a dedicated requirement model – SysML in this case will store the requirements and links to the model elements – so we have to adapt the requirement views (upstream and current views) so that they are not linked to a specific requirement meta-model but can display requirements from many different sources.

10. Conclusion

Feedbacks are promising for this model-driven tool chain. At any time, a team member can prove and justify his design choices according to traceability defined between requirements and model elements.

We have seen that TOPCASED-REQ is a solution to manage traceability links between upstream documents and model elements. Now, it would be interesting to consider traceability at different stages of the V-cycle and not only at the specification level.

11. Acknowledgement

We thank the Airbus teams from their interactions, advices and feedbacks helping us to improve the work performed.

12. References

[1] P. Farail, P. Gaufillet, A. Canals, C. Le Camus, D. Sciamma, P. Michel, X. Cregut, M. Pantel, "The TOPCASED project: a Toolkit in Open Source for Critical Aeronautic Systems Design", ERTS 2006, 25-27 January 2006, Toulouse, France.

[2] A. Canals, S. Gabel, P. Gaufillet, "Les composants SAM et OCL du projet TOPCASED", NEPTUNE 2009, 26-27 May 2009, Paris, France.

P. Gaufillet, S. Gabel, "Avionic Software Development with TOPCASED SAM", ERTS 2010, 19-21 May 2010, Toulouse, France.

[3] A. Canals, C. Le Camus, M. Feau : An Operational Use of ATL: Integration of Model and Meta Model Transformations in the TOPCASED Project. DASIA 2006 - Data Systems in Aerospace, Proceedings of the conference held 22-25 May, 2006, Berlin, Germany. Edited by L. Ouwehand. ESA SP-630. European Space Agency, 2006. Published on CDROM., p.40.1

13. Glossary

MDD: model-driven development

TOPCASED: Toolkit in OPen source for Critical Applications and SystEm Development

SAM: Structured Analysis Model

SART: Structured Analysis for Real Time

OMG: Object Management Group

UML: Unified Modeling Language

SysML: Contraction of System and UML

SMUC: State Machine from UML Compiler

M2T: Model 2 (to) Text

API: Application Programming Interface

ATL: ATLAS Transformation Language

EMF: Eclipse Modeling Framework

DSL: Domain Specific Language