The TOPCASED project: a Toolkit in Open source for Critical Aeronautic SystEms Design

Patrick Farail, Pierre Gaufillet, Agusti Canals, Christophe Le Camus, David Sciamma, Pierre Michel, Xavier Crégut, Marc Pantel

The TOPCASED project: a Toolkit in Open source for Critical Aeronautic SystEms Design

Patrick Farail[1], Pierre Gaufillet[1], Agusti Canals[2], Christophe Le Camus[2], David Sciamma[3], Pierre Michel[4], Xavier Crégut[5], Marc Pantel[5]

1: AIRBUS FRANCE, 316, route de Bayonne F-31060 Toulouse - France

2: C/S, ZAC de la Grande Plaine - Rue Brindejonc des Moulinais - BP 5872 - 31506 Toulouse cedex 5 - France

3: Anyware Technologies, Prologue 2 - Rue Ampère - BP 87216 - 31672 Labège Cedex - France

4: FéRIA-ONERA, -DTIM, 2, avenue Edouard Belin, F-31400 Toulouse - France

5: FéRIA-IRIT-ENSEEIHT, 2, rue Charles Camichel, BP 7122, F-31071 Toulouse cedex 7 - France

Abstract: The TOPCASED project aims at developing an open source CASE environment for critical applications and systems development. Its main benefits should be to perpetuate the methods and tools for software development, minimize ownership costs, ensure independence of development platform, integrate, as soon as possible, methodological changes and advances made in academic world, be able to adapt tools to the process instead of the opposite, take into account qualification constraints. This paper focuses on the meta-modelling principles used in the TOPCASED CASE environment. It includes a tool to automatically generate graphical editors for specific languages based on their meta-model. This generation includes a customization stage before and after the generation. It has been used to develop editors for UML2, Ecore, SAM and AADL meta-models. Models are also the way tools of the environment communicate with each other thanks to a model bus. Importing models created with external tools is possible thanks to transformation of external models to TOPCASED ones. Transformations are made using ATL. This paper describes the MDE approach used in TOPCASED and gives insights on the whole project.

Keywords: *Open source, Model driven engineering, Critical systems design, Meta-modelling, Model editor generator, Model bus*

1. Introduction

The creation of industrial systems relies on numerous tools on which it is essential to capitalize in order to optimise development costs. However, the lifetime of critical systems such as aerospace products is often about 10 to 30 years, and currently, no software editor is able to commit for such a long time at an acceptable cost. To counter these risks, the CNRT [1] (National Center for Research & Technology) Aeronautic & Space partners have put forward the TOPCASED project (standing for Toolkit in Open source for Critical Applications and SystEms Development) which aims to develop an open source CASE environment with the following goals :

- to perpetuate methods and tools used for software developments,
- to minimize ownership costs,
- to ensure independence of development platforms,
- to integrate, as soon as possible, methodological changes and advances made in the academic world,
- to be able to adapt the tools to the process, and not the opposite,
- to take into account qualification constraints.

This article describes the TOPCASED project, especially its architecture – platform, how the tools are built, how they communicate, ... – and technical choices – a meta-modelling approach [2], the use of Eclipse [3], EMF [4], Model Bus [5], ... The relationship with some other projects (ASSERT [6], ModelWare [7], TopModL [8], Cotre [9]) are also discussed. We then present the current status of the project and the tools developed during the first semester of 2005. At last, the roadmap of the coming developments will be presented.

2. Requirements and TOPCASED architecture

The design of critical systems and softwares currently requires the use of many different models at the various steps of both the system development process and its life-cycle. These models allow to express the various aspects of the system : both static and dynamic, functional and non-functional, at the systems and at the components level, software and hardware, clients and providers, ...
Many kind of tools have to be available to handle these models: textual and graphical editors; translators from one model type to an other; code, test and documentation generators, version control systems, model validation tools, ...

The main technical requirement for a modern extensible and evolutive CASE tool is that it should be able to provide users with an easy access to the various models of a given system and to their associated tools.

The TOPCASED project is based on Model Driven Engineering [10] (MDE) for this purpose:

- meta-modeling allows to describe all the modeling languages in a common framework;
- model bus allows to access easily to the various tools;
- model transformations allows to relate the various models and adapt models to the various tools involved in a project;
- generative programming allows to easily produce both textual and graphical model editors.

In the current industrial processes, these software tools will have clearly a more and more important place. Unfortunately, experience shows that it is really difficult for enterprises to keep mastering their software tools for the whole lifetime of their own products (lifetime which can be as long as several dozen of years). Indeed, the current industrial plan consists in sub-contracting the implementation and diffusion of the tools to specialized editors which are de facto the owners. Ownership problems usually arise when the interests of the industrial user and of the editor diverge, or when the editor is bought by an other company or disappears, or when the technology evolves. It is mainly for this reason that the TOPCASED project has turned to free (like in freedom, not in costless) and open source software.

The fact that the tools documentation and source codes are open guarantees that it will be successfully maintained, possibly, in the worst case, at the price of creating a specific development branch, depending on the requirements of the industrial users.

Working with open source software brings also some other advantages :

- it allows to adapt the tools to the processes, where it is today required to do the opposite most of the time.
- the availability of the software interfaces ensures a good interoperability.
- the supported platforms are not limited, like it is often the case, to the most common platforms at a given time.
- the tools and their source code being available, the academic actors can integrate very quickly their new techniques, and use the tools for the training courses.

The cost of the tools becomes the cost of development, deployment and maintenance, where today the most common proprietary tools are invoiced in proportion to the number of users. Moreover, this cost can be shared between the users, and be reduced by the reuse of existing open source components.

The implementation of the TOPCASED project relies on the open source plugin-based Eclipse CASE platform. The first tools developed are therefore Eclipse plugins. Nevertheless, the TOPCASED architecture also allows to integrate smoothly external tools.

Of course, as TOPCASED addresses critical systems and software design, it has to take into account the processes which imply some products and tools qualification constraints, processes which are quite frequent in the embedded systems domain (DO-178B, DO-254, ECSS, etc.).

3. Model Driven Engineering and meta-modelling

The "*Object Management Group*" (OMG: *http://www.omg.org/*) was created in 1989 to provide frameworks and standards for the integration of object-oriented applications, mainly the CORBA [11] and UML [12] technologies.

The growing diversity of techniques and platforms used in software developments, as well as the emergence of non-CORBA based middlewares, like EJB from Sun and .NET from Microsoft, have taken the pre-eminent role away from CORBA. Then, OMG refocused its strategy and standards to support the MDA approach [13,14].

MDA addresses the complete life cycle analysis design and programming aspects, providing an interoperability framework for defining modular and interconnected systems, and with the will to offer more flexibility in system integration and system evolution. A system design is organized around a set of models and a series of transformations between models, into a layered architecture.

Central to MDA is the principle of defining different models at different levels of abstraction and linking them together to form an implementation. MDA separates the conceptual elements of an application from the representation of these elements on particular implementations technologies. For that purpose a distinction is made between "*Platform Independent Models*" (PIMs), representing the conceptual design of the application, and "*Platform Specific Models*" (PSMs) which are more solution oriented.

Transformations between models are key elements in the MDA approach: vertical transformations between PIMs and PSMs to express realizations, or horizontal transformations between PSMs for integration features.

Underlying these model representations and transformations is the notion of "*meta-model*". The ability to express and transform models requires a rigorous definition of the (textual or graphical) notations supporting these models. Therefore, the notations to express models must themselves be described into models, which are called "*meta-models*". For example, the UML meta-model formally describes the notation of the different UML diagrams, giving so an unambiguous and precious common base to all tool providers and users.

The importance of meta-models has been recognized by OMG who proposed the "*Meta Object Facility*" (MOF). The MOF [14, 15] provides a meta-modeling hierarchy as well as a standard language

for expressing meta-models. It proposes a structure of data in four different levels, so called the "*four-layer meta-model architecture*":

- the bottom M0 level, or *Application level* gives the data values i.e. the extension of an application,
- the M1 level defines the model i.e. the intension for an application,
- the M2 allows to define a schema, or language, for the application model: it is the meta-model level,
- the M3 level, or *MOF level,* proposes general concepts, defining the MOF meta-language, i.e. a schema for a meta-model: it is the meta-language level.

The *four-layer meta-model architecture* is now widely accepted and was proposed (before the OMG MDA-MOF proposals) in other standards like the ISO/IEC standard for IRDS ("*Information Resource Dictionary System"*) [16] and CDIF ("*CASE Data Interchange Format*") from EIA [17].

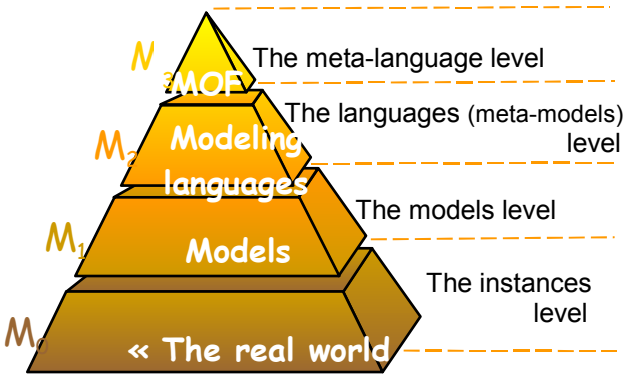


Figure1: The four-layer meta-model architecture.

As TOPCASED relies on the Eclipse platform, the M3 meta-modeling language used is Ecore provided by the EMF ("*Eclipse modeling Framework*") project [4] which is strongly related to Essential MOF 2.0 as specified by the OMG [15] . Several M2 modeling language editors have been developed (ECORE, UML2, SAM, AADL) and others will follow (SYSML, SPEM, ...). The M1 level then corresponds to specific system models (an UML use case, activity or class diagram) and the M0 level to instances of the models (a real execution of the system).

4. Communication and Basic services

According to MDA, models are treated as first-class elements in software development. MDA application requires a wide range of model operations such as model edition, model storage, model manipulation, code generation and model transformation.

A lot of tools are now available commercially or as open source and provide various model operations. It has been widely observed that a single tool is not sufficient to realise a complete MDA software production. This is because the MDA software production requires a wide range of model operations to be used in different software development activities (e.g. analysis phase, design phase, test phase, implementation phase).

Therefore, the fact that model operations are not supported by the same tool must not prevent models to be processed by all the operations required in the MDA software production. For this reason, when users need to use two tools conjointly, they must be able to send an output model produced by an operation of tool T1 as an input to an operation of tool T2. The term "*operation connection*" is used to denote the action of linking an operation's output to another operation's input.

Model Bus is an architecture defined by X. Blanc at LIP6 [6] in the IST ModelWare project which allows model operations to be connected. TOPCASED has currently defined and implemented a simple yet effective model bus and will later on be also connected to X. Blanc model bus.

Several basic services are required in the TOPCASED platform mainly for the deployment and management of the various Eclipse plugins through the Eclipse platform and the model bus.

Based on the OSGi Eclipse mechanisms the TOPCASED bus allows to retrieve all the services registered on the bus. TOPCASED services deployed as plugins can then be tested using a basic console, which allows to run locally registered services providing parameters values. Services can be retrieved using a basic ontology, so that developer does not need to know the exact interface. Furthermore, the user can chain functional treatments which need intermediate output results.

TOPCASED facilities can be split in two groups :

- deployment and management plugins,
- communication plugins.

First objectives in the TOPCASED project has lead us to assign most of the efforts on the plugins deployment and development project management. TOPCASED offers today plugins that ease the deployment :

- for an eclipse adapted platform from a local or network full installed Eclipse reference. This feature allows to build a specific Eclipse environment centred on the user's selection of targeted plugins – mandatory plugins to selected ones are automatically retrieved and packaged according to user requirements and Eclipse constraints.
- for a specific project through the build of a specific Eclipse feature from already done set of deployable jar.
- by generating a master configuration file in order to run Eclipse with a referenced configuration and share the same preferences.

and the management of a project :

- allowing to build project templates from an Eclipse project (Java, ...).
- allowing to reuse existing project templates to create new projects following the given architecture.

Communication between TOPCASED and external tools may occur in both ways. On one hand, from TOPCASED to external tools, it proposes to generate part of an adaptator which registers the new functionality as a new TOPCASED service. It can then be invoked through a wizard that can help the developer in providing parameters and generate the structure of the new plugin. Other mechanisms will also be implemented in the future. The developer can then implement the part that is specific to the tool, and takes benefits of all the registered services.

The sequence diagram in figure 2 explains how the communication between TOPCASED and an external tool has been considered.

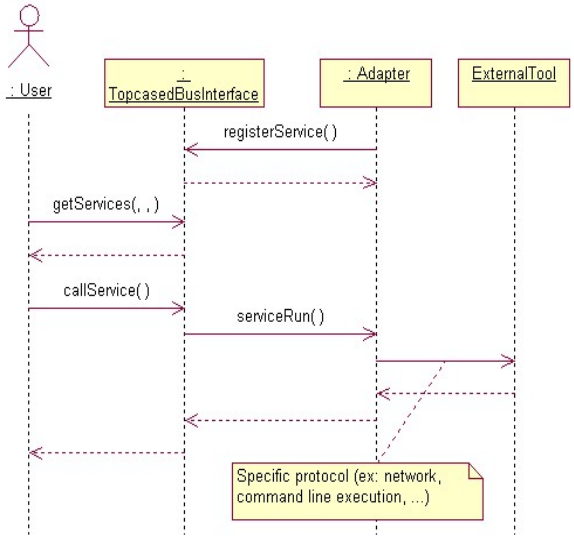


Figure 2: Communication from TOPCASED to an external tool.

On the other hand, from an external tool to TOPCASED, it offers a HTTP Web server, whose port is customizable and which allows restrictions over the authorized connections through preferences page. This Web server is launched at the TOPCASED start and wait for SOAP requests. Available web services can be accessed through a getServices request. To allow to retrieve available web services, TOPCASED bus offers the same mechanism as for its local services, providing extension for web services registering. Model transformations offered in TOPCASED to import models from previously used formalism are currently available through this mechanism.

The sequence diagram in figure 3 exposes how the communication has been thought.

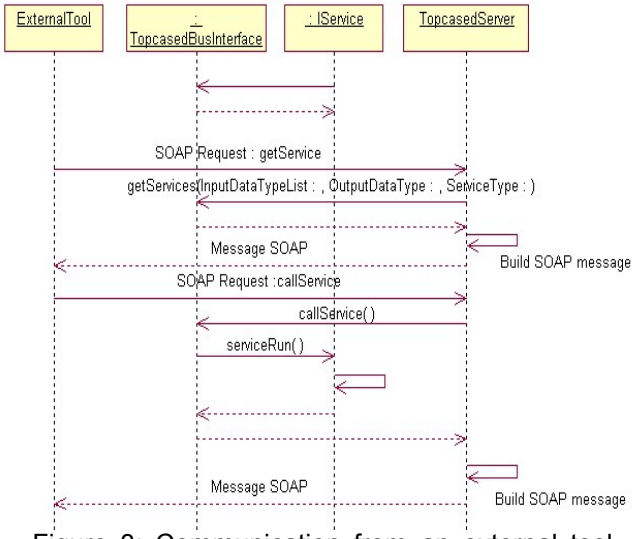


Figure 3: Communication from an external tool to TOPCASED.

5. Use of model transformation

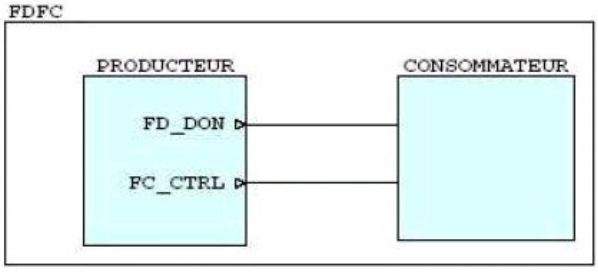
MDE advocates the use of model transformation to relate two different models of the same system. Model transformations basically allows to translate a model in a given modelling language to another model in another language.

TOPCASED currently relies on the ATL model transformation Eclipse plugin defined and implemented in the INRIA Triskell project which is part of Eclipse Generative Model Framework project. At this stage of the project, model transformations have been mainly used in order to access models defined using other editors than TOPCASED one's.

To illustrate the approach, let's rely on the import of SILDEX models ([18] a synchronous language based graphical modelling formalism developed by TNI and currently in use at AIRBUS) in the TOPCASED SAM modelling language.

A meta-model for SILDEX defining all the concepts that can be handled in TOPCASED SAM formalism has been defined. The SILDEX example below illustrate the communication of two sub-systems named “Producteur” and “Consommateur” via data flows and control flows connected on ports (not visible on the sub system “Consommateur”).

Figure 4: An example of communication between two sub-systems



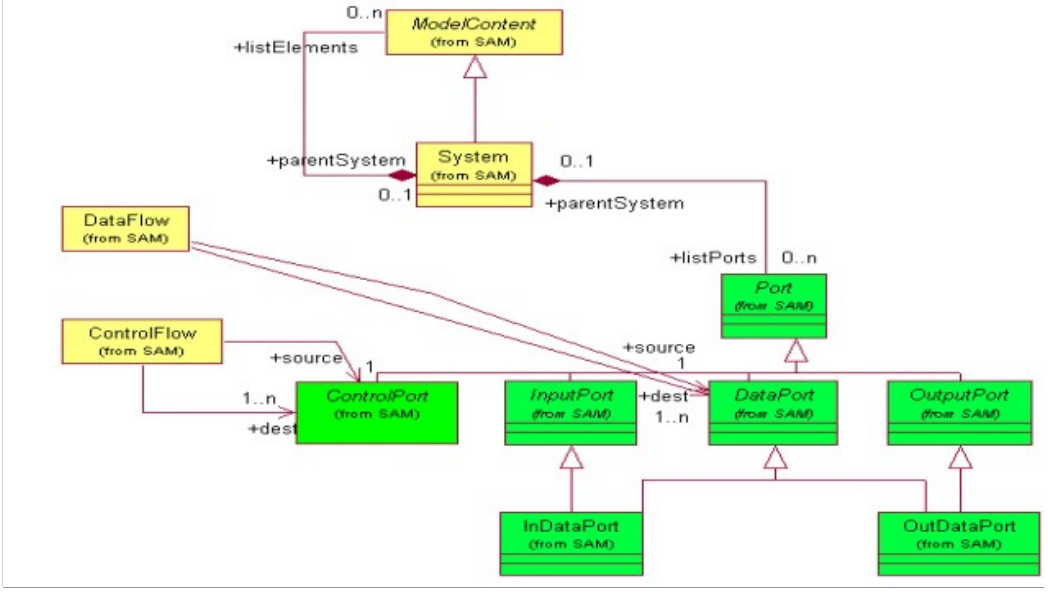


Figure 5: A simplified view of the SAM meta-model

This example illustrates the following notions: System, Data flows prefixed by FD_, Control flows prefixed by FC_, Ports that we specialised depending on their nature and their direction. To precise some relations, a system can contain sub systems, and a flow is linked by ports. The confrontation of the analysis of other examples allowed to extend the SAM meta-model whose simplified view is proposed in figure 5.

After the creation of the source and target meta models, in our example SILDEX and SAM, the TOPCASED process to make a transformation can be breakdown as follow : first, the mapping of the concepts, then the writing of the transformation rules (in our case with ATL/OCL) and the automatic transformation by rule execution (also with ATL), at last, manual check of the result through the reading of the files obtained by the editors (in our example TOPCASED SAM Editor).

Today the TOPCASED service to transform SILDEX models to SAM Models is operational and used by AIRBUS France.

More information on this work can be found in [19].

6. Generating model editors

As many different modelling languages are required, the production of model editors will be a key task for TOPCASED. Most of the currently available editors rely on hard-wired technologies with very little code reuse between editors. Model driven engineering, and specifically generative programming, can be applied in order to ease the development of these editors.

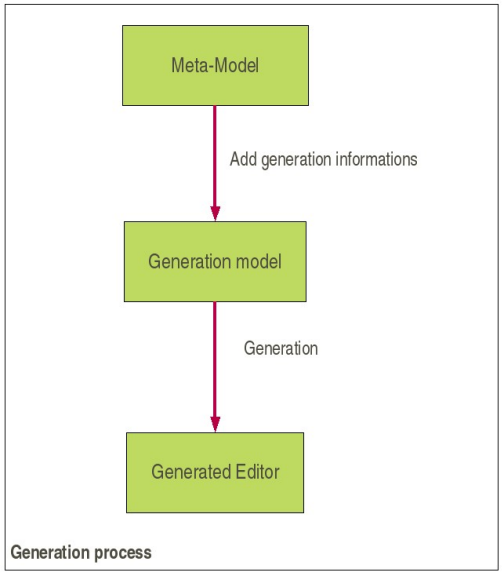


Figure 6: Generation process

TOPCASED has specified and implemented a simple yet effective MDE based editor generator for the Eclipse EMF/GEF platform. The editors are generated according to the modelling language description (its meta-model), and several models specifying the graphical primitive and interaction used in order to edit each part of the modelling language. This approach is multi-staged and based on the Eclipse JET (Java Emitter Template) for code generation. Each stage takes the meta-model as parameter and a model describing the various aspects of the editor in order to generate Java classes for the handling of the models, a hierarchical editor and then a graphical editor. This approach has been proposed as a contribution to Eclipse GMF ("*Graphical Modelling Framework*") [20].

The main requirement for a model editor is to be compliant with the model constraints – defined in the meta-model – and with the usual graphical notation.

A lot of information about the way to edit a model are already in the meta-model and can be used to generate a part of the editor. Other information – like graphical presentation, available diagrams... - are not available in the model. Another model is then used to store the additional parameters needed for the generation.

The process to create a new model editor can be split into several steps :

- the meta-model definition – using the Ecore modelling language
- the generation model definition describing the editor behaviour and the graphical informations
- the execution of the generation action provided by TOPCASED. The result of this execution is a functional graphical editor compliant with the input meta-model
- the last step is optional. It consists of a Java customization of the generated code. During this step the graphical representations can be customized to fit with the requirements of the meta-model specifications.

An example of this process for the Ecore meta-model is given in figure 7.

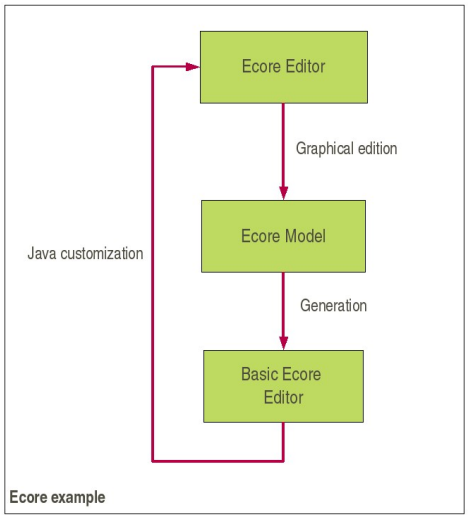


Figure 7: Ecore process example

The graphical editor needs to store graphical informations about the current edited model (position of objects, colors...) but this kind of properties are not defined in the domain meta-model. To solve this problem TOPCASED uses the OMG standard, XMI-DI (XMI – Diagram Interchange).

A simplified view of the XMI-DI model is presented in figure 8.

The XMI-DI model wraps the domain model and adds all the missing informations needed to display the model. Using this “graphical” model, we can exchange the diagrams with others modelling tools and export it to external formats (RSM, Together, SVG...).

TOPCASED is based on the Eclipse JET engine (Java Emitter Templates) for code generation. With this generation engine, the generated code can be customized and then re-generated without losing already defined customizations.

Using TOPCASED and EMF you can also generate Java classes for the handling of the models within Eclipse, a hierarchical editor, documentation reports (HTML, PDF), helper classes to handle context menu...

This approach has been proposed as a contribution to Eclipse GMF (Graphical Modelling Framework).

7. Project infrastructure and perpectives

TOPCASED has been started through the CNRT-AE (Centre National de Recherche Technologique Aéronautique et Espace, [1]) in 2004, and is now a project of the Aerospace Valley Regional Competitivity Pole.

It gathers today some major industrial partners, like EADS Airbus, EADS Astrium, Atos-Origin, CS, Siemens-VDO and Thales Aerospace; some SME like Adacore, Anyware Technologies, Micouin consulting, Sinters, and Tectosages; and some laboratories and schools : ENSEEIHT (repository of the collaborative development gforge platform for TOPCASED), ENSIETA, ESEO, ESSAIM, FéRIA-IRIT, FéRIA-LAAS, FéRIA-ONERA, INRIA Rhone-alpes, INRIA-IRISA, the Federal University of Santa Catarina (UFSC, Brazil), the Paul Sabatier University (UPS).

This partnership is open. To become a new member and to participate to the strategic decisions, you will need to accept the TOPCASED membership charter, and your proposal will have to be validated by the steering committee. Of course, our tools remain publicly available.

The initial TOPCASED infrastructure developments have been funded by Airbus, and have been specified and implemented by Airbus, Anyware Technologies, CS and FéRIA.

Only a small part of the full project has been addressed until now, mainly work packages WP2 – Model editors tasks :

- SP2.1 - State of the art on modelling methods and languages
- SP2.2 – Meta-modelling language definition
- SP2.3 – Model editors specifications
- SP2.4 – Editor for the meta-modelling language
- SP2.5 – Editor for UML2 (currently use case and class diagrams)
- SP2.6 – Editor for AADL/COTRE.

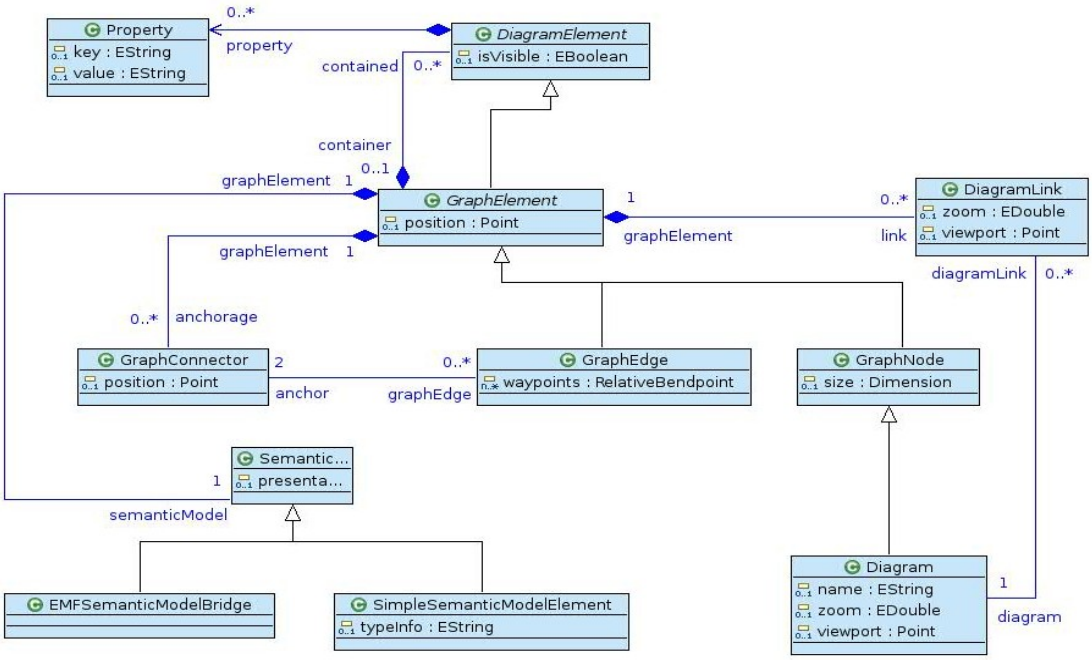


Figure 8: A simplified view of the XMI-DI

This work led to the development of the first versions of the TOPCASED toolkits which are available through the project website (http://www.topcased.org) or the development server (http://gforge.enseeiht.fr).

Some work still need to be done on the improvements of the editors, on the development of new editors and on the semantics side of meta-models in order to allow the use of formal model validation and simulation tools. A Ph.D. has started at FéRIA-IRIT has started on the expression of the semantics of development processes at the meta-model level. A preliminary integration of OCL for the validation of models will be soon available.

Many other points will be studied and developed during this project:

- WP1 – Process and life-cycle for critical systems, which will provide a common process for the development of such systems taking into account the constraints and experiences of the various partners. All the concepts used in TOPCASED will be defined through a common ontology. The use of the MDA Y approach in system engineering will also be assessed. The WP1 tasks have already started.
- WP3 – Model verification and simulation, which will allow to integrate easily at the model level validation approaches based both on formal tools and simulations in the toolkit. A Ph.D. has started between FéRIA-LAAS and Airbus to work on pre-emptive software models verification.
- WP4 – Implementation tools. This work package aims to select, and if required to improve and adapt to the qualification constraints the numerous implementation tools (GCC, binutils, GDB, CDT, JDT, etc.) already available in open source.
- WP5 – Model transformations which will allow to interconnect the various tools and produce both code and documentation using the various models. Some tasks have been started at the end of 2005.
- WP6 – Tools interoperability and Data management. The first result of this work package is the current bus used by the TOPCASED tools. A work is also in progress to start to integrate Stood (a HOOD software design tool). This work package is also in charge of transversal tools like anomalies management, requirement management, and version control systems.
- WP7 – Infrastructure. Three points will be studied in the WP7 : the juridical point of view (what open source licenses are acceptable for the components we use, what open source licenses are acceptable for the components we release, software patents issues, insurance, copyright, etc.), the business model (who and in what conditions will maintain and develop the tools, how to guarantee the perpetuity of the skills, etc.), and the technical means (collaborative development server, etc.). some work have already been done on these 3 points.
- WP8 – Safety. This work package has to specify on the one hand specific tools to ensure safety of the critical systems, and on the other hand to define some requirements on the other

TOPCASED tools to ensure that they will have a positive impact on the final safety level of the products.

TOPCASED is today in touch with several other projects, like the Society of Automotive Engineers (SAE) Architecture Analysis and Design Language (AADL) standardization committee and the IST project ASSERT for AADL editors and analysis tools, the IST project ModelWare for the definition and the development of the ModelBus, the ATLAS group at INRIA, which is working on the ATLAS Transformation Language (ATL) used for our first implementations of models transformations, and the TopModL initiative, which aims to develop meta-modelling tools.

8. Acknowledgement

The authors acknowledge the contribution to this work of F. Migeon and X. Thirioux and all the developers and testers from Airbus, AnyWare and C/S involved in the project.

9. References

[1] CNRT: "*CNRT-AE: Centre National de Recherche Technologique Aéronautique et Espace.*", URL: http://www.cnrtae.com/ (in french), 2005.

[2] M. Dahchour, A. Pirotte, E. Zimányi. "*Definition and application of metaclasses.*", In Proceedings of the 12th Int. Conference on Database and Expert Systems Applications, DEXA'01, Munich, Germany, LNCS Vol. 2113, Springer-Verlag, p. 32-41, Sept. 2001.

[3] Eclipse consortium: "*Eclipse: an extensible development platform and application frameworks for building software.*", URL: http://www.eclipse.org/, 2005.

[4] EMF project: "*EMF: Eclipse Modelling Framework.*", URL: http://www.eclipse.org/emf/, 2005.

[5] X. Blanc, M.-P. Gervais, P. Sriplakich, "*Model Bus : Towards the interoperability of modelling tools*", MDAFA'04, Linköping, 2004.

[6] Assert project: "*Assert: Automated proof based System and Software Engineering for Real-Time.*", URL: http://www.assert-online.net/, 2005.

[7] ModelWare project: "*MODELWARE: MODELling solution for softWARE systems.*", European research project (IST), URL: http://www.modelware-ist.org,2005.

[8] P.-A. Muller, C. Dumoulin, F. Fondement, M. Hassenforder. "*The TopModL initiative.*", 3rd Workshop in Software Model Engineering , 7th International Conference on the UML (WiSME@UML 2004), Lisbon, Portugal, Oct. 2004.

[9] B. Berthomieu, P.O Ribet, F. Vernadt, JL Bernartt, J.M Farines,J.P. Bodeveix, M. Filali, G. Padiou, P. Farail, P.Gaufillet, P. Dissaux, J.L. Lambert: "*Towards the verification of real-time systems in avionics: the Cotre approach*", 8th International Workshop on Formal Methods for Industrial Critical Systems (FMICS'03), Trondheim (Norway), June 2005.

[10] B. Selic. "*The Pragmatics of Model-Driven Development,*", IEEE Software, Vol. 20(5), Sept. 2003.

[11] OMG. "*The Common Object Request Broker: Architecture and Specification.*", Revision 2.0, OMG document formal/97-02-25, Jul. 1995 (updated Jul. 1995, until Revision 2.3 June 1999). URL Revision 2.0: http://www.omg.org/cgi-bin/apps/doc?formal/97-02-25.pdf . URL Revision 2.3: http://www.omg.org/cgi-bin/apps/doc?formal/98-12-01.pdf

[12] OMG. "*Unified Modeling Language Specification.*", Version 1.5, OMG document formal/03-03-01, Mar. 2003. Available at: http://www.omg.org/cgi-bin/apps/doc?formal/03-03-01.pdf

[13] A. Kleppe, J. Warmer, W. Bast. "*MDA Explained: The Model Driven Architecture Practice and Promise.*", Addison Wesley, 2003.

[14] D. Frankel. "*Model Driven Architecture: Applying MDA to Enterprise Computing.*", Wiley Press, 2003.

[15] OMG. "*Meta Object Facility (MOF) Specification.*", Version 1.3, OMG document formal/00-04-03, Mar. 2000. URL: http://www.omg.org/cgi-bin/apps/doc?formal/00-04-03.pdf . URL Version 1.4: http://www.omg.org/cgi-bin/apps/doc?formal/02-04-03.pdf

[16] ISO, Information Technology. "*Information Resource Dictionary System (IRDS) Framework.*", standard ISO/IEC 10027, 1990 (and ISO/IEC 10728, 1993).

[17] EIA. "*Framework for Modeling and Extensibility.*", Extract of Interim Standard, EIA/IS-107, Electronics Industries Association, CDIF Technical Committee, Jan. 1994.

[18] Tni Europe: "*Sildex: a formal approach to real-time applications development.*", URL: http://www.tni-world.com/rtbuilder.asp ,2005.

[19] A. Canals, C. Le Camus, M. Fleau, G. Jolly, V. Bonafous, P. Bazavan: " *An operational use of ATL: integration of the model transformation in the TOPCASED project*", ICSSEA'2005 Conf., 2005.

[20] GMF project: "G*MF: the Graphical Modelling Framework.*", URL: http://www.eclipse.org/gmf/, 2005.

10. Glossary

AADL: Architecture Analysis and Design Language
EMF: Eclipse Modelling Framework
EMFT: EMF Tools
GEF: Graphical Editor Framework (part of Eclipse)
GMF: Graphical Modeller Framework (part of Eclipse)
GMT: Generative Modelling Framework part of Eclipse)
HOOD: Hierarchical Object Oriented Design
MDA: Model Driven Architecture (part of OMG)
MDE: Model Driven Engineering
MOF: Meta-Object Facility (part of OMG)
OCL: Object Constraint Language (part of OMG)
OMG: Object Management Group
TOPCASED: Toolkit in Open source for Critical Aeronautic SystEms Design
UML: Unified Modelling Language (part of OMG)
XMI: XML Metadata Interchange(part of OMG)
XML: Extensible Markup Language (from W3 Consortium)