A multi-model process for managing project complexity

A. Boulle, Marie-Line Valentin, Anthony Inard

A multi-model process for managing project complexity

A. Patrice BOULLE[1], B. Marie-Line Valentin[2], C. Anthony Inard[3]

1: SYSALYS, 7 rue du grenier à neige 91370 Verrières le Buisson

2: AIRBUS France, 316 route de Bayonne 31060 Toulouse Cedex 9

3: EUROCONTROL, Centre des Bois des Bordes - BP15 - 91222 Brétigny-sur-Orge Cedex

Abstract: Developments of large software systems have to face an exponential increase in volume and complexity of the requisite knowledge, tighter time constraints and a continual growing number of stakeholders. Current solutions are based on breaking down the system or software components and equipments in smaller pieces in order to be able to handle them, but the side effect of this approach is to dilute the synthetic vision. This paper presents a model centric process supported by tools. This process is based on a multi-model approach for managing synthetic views and making project information available to all stakeholders, with an appropriate view and presentation.

Keywords: meta-model, multi-models, model transformation, UML

1. Introduction

Today, companies developing large software or system engineering projects (aeronautics, space, defense, automotive…) have to deal with an exponential increase in volume and complexity of the requisite knowledge, tighter time constraints, and a continuously growing number of participants.

Companies answer by a methodological reinforcement based on analytic breakdown, which strongly separates system components but dilutes the synthetic vision. This synthetic vision relies most often on synthesis capabilities and personal performance of Systems Engineers.

Tools often propose a centralized data management based on document structure. This approach encounters several difficulties:

- Data centralization does not suppress source multiplicity but increases duplication and then, inconsistency risks.
- Data synchronization between database and sources are difficult and often impossible to achieve. Information is often incomplete or insufficiently verified.
- Impacts of changes are difficult to estimate when the analyst needs to compile multiple sources of information.

This paper presents a model centric process supported by tools, which is implemented in different application domains such as Air Traffic Control (EUROCONTROL), Aeronautics (Airbus) or Automatic Railway System. This process is based on a multi-model approach for managing synthetic views and making project information available to all stakeholders, with an appropriate view and presentation.

The objectives of the process are the following:

- Urge project members to structure and formalize their knowledge and their system view by the means of connected models (Requirements, Architecture, Component, Deployment, Interface / Communication and Equipment / Execution Platform),
- Support teamwork by sharing and synchronizing relevant information,
- Produce adapted models that contain linked information necessary for a specific task,
- Manage dependencies and consistency between the different models, especially when their development lifecycles are different,
- Establish and evaluate impacts and costs linked to changes,
- Produce an up-to-date status of the project developments.

The process is supported by a prototype tool named MME for Multi-model Editor developed by SYSALYS. The concepts used by each type of stakeholder are defined through meta-models. MME provides import, export and filtering capabilities, which support merging, linking and “gluing” information coming from various documents or models produced by different tools.

In this article, for each issue identified in a multi-model approach, we describe the process to handle the issue and illustrate it with the support of the MME tool.

2. Multi sources of information and Meta-model definition

A model driven process aims at representing in models all the project information and links which exist between them [1].

This information is contained in documents, which most often have different formats. For example, a use case and its actors can be defined in a UML model but conditions, steps, alternatives and traceability links towards the requirements can remain outside the model, in an external document although they constitute also essential data for the project.

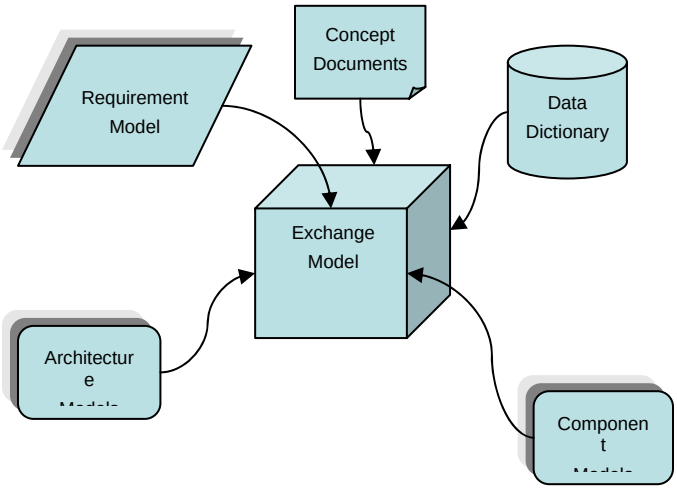

Figure 1: Exchange Model Context

If the reference is a UML model or a database, the structure and, to a certain extent, the semantics of information is defined with precision; that is not always the case if the reference is a text or a proprietary tool file.

The first thing to do is to draw up the list of the documents which contain the project data, to describe for each one the information it contains and to clarify the links between information defined in the different documents. Information, which is repeated in several documents, must be highlighted, in order to determine where it is defined (origin of the definition) and where it is referred to or made more precise. This activity results in the elaboration of the meta-model.

The data and their relationships modelling in a meta-model is an essential activity for the mastering of the models. At this stage, it is not necessary to clarify all the data and the internal links but it is essential to focus on the data shared by several documents and the links between the data located in different documents.

For example, the functional requirements have to be traced in the other models and thus have to be extracted and referred to in the other models of the project. On the other hand, the additional data which is necessary to be added in order to be able to simulate and to validate the model at a given phase of the development (for example, during system design) should not be exported to the next development phase (in the same example, to the software specification).

The border between “private” data to a given activity or a given user profile is not easy to determine, and that’s why the meta-model has to be updated throughout the project life cycle, each time new documents, new links… have to be taken into account. The review of the meta-model allows to highlight implicit data, links which are not obvious or redundancies / ambiguities in the definitions.

The analysis of the documents allows the detection of lacks, ambiguities and redundancies. It also makes it possible to propose modifications of the textual documents contents and format, in order to improve the presentation and to facilitate information extraction.

The MME tool provides a meta-model editor, which allows in a simple way the definition of the project meta-model. Each data is defined with its properties and its links. The compilation of the meta-model generates the body of a MME editor associated with the meta-model and also the related documentation.

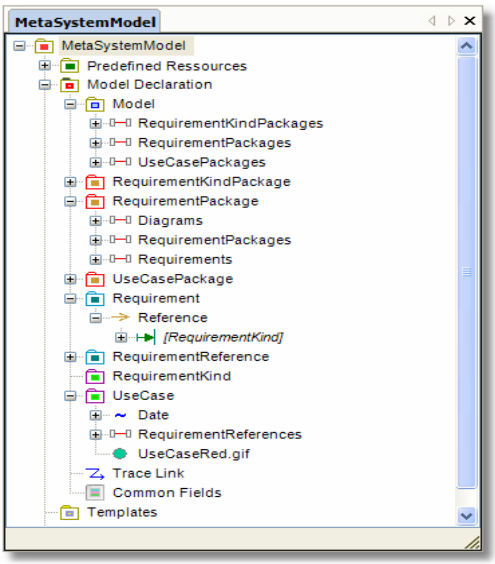

Figure 2: Meta Model

The Meta Model Editor creates an instance of a model editor with which it is possible to edit a model corresponding to the meta-model.

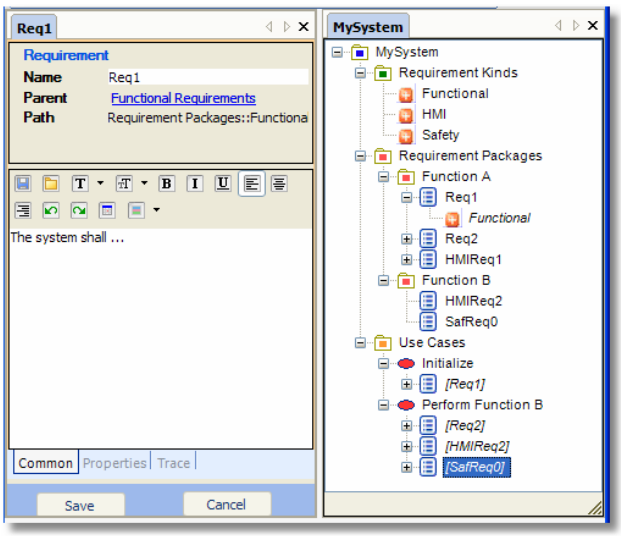

Figure 3: Model editor corresponding to the "MetaSystem" meta-model

3. Importing and Exporting data

The model provides a picture of the essential data of the project and the links between these data. The data have to be extracted from the documents or the files in which they are defined or referred to, in order to present them in a view, which complies to the meta-model.

To build and update the model, it is necessary to implement mechanisms of extraction and filtering from the various documents or files. These documents or files have various formats: the files produced by tools such as UML editors follow an XMI representation which allows a quite easy extraction and a simple filtering; other tools with a proprietary format provide an API which makes it possible to export relevant information, with more or less significant development.

The importation of essential data contained in the documents is a critical phase of a model driven process. The implementation of the import function may require complex developments and generally requires the development of a corresponding export function in order to be able to re-synchronize the documents with updated information.

The resulting model can be considered as a high level model or an exchange model, which contains the essential data with their links.

MME provides tools which allow to import data starting from the documents developed with standard text editors (word, excel…) and editors providing an XML output (UML/XMI). It is also possible with MME to create extraction scripts, which use the tools API.

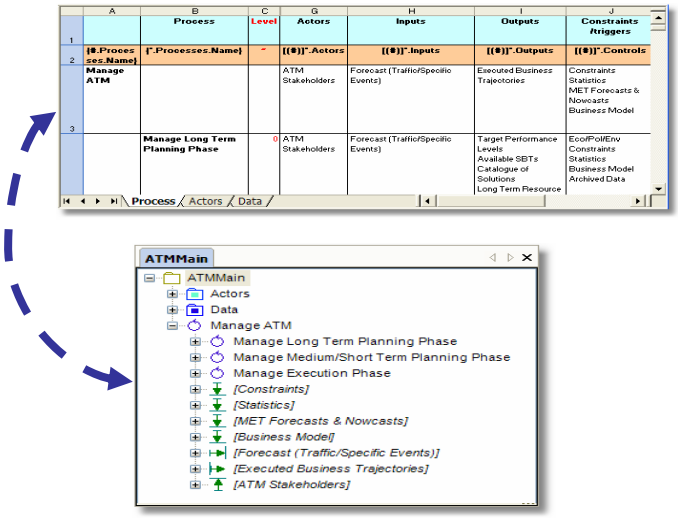

Figure 4:Importation and exportation of a model using a spreadsheet (figure provided by courtesy of EUROCONTROL)

The exchange model can be modified directly in MME (by adding, removing or modifying items of model if the corresponding meta-model permits it). Then the export functions provided by MME allow the propagation of modifications into the source documents. The UML/XMI export function makes it possible to create a model UML starting from the exchange model. For example, starting from the extraction of the relevant data from the system design model contained in the exchange model, it is possible to initiate a validation model by inserting the necessary data in it [2].

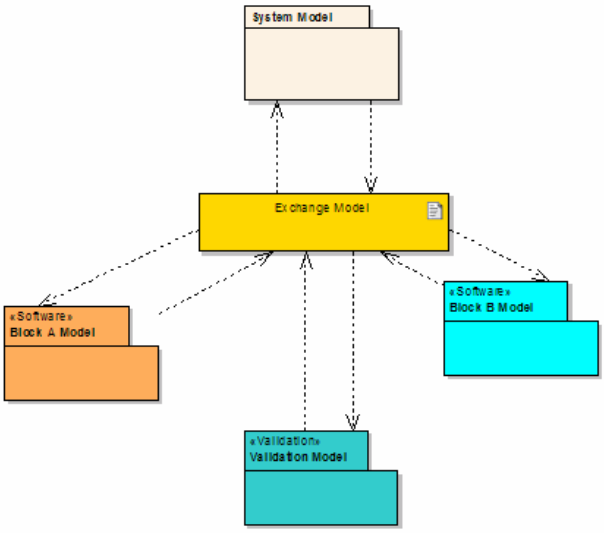

Figure 5: Creation and Synchronisation of models through the Exchange Model

The use of import/export functions makes it possible to create and update the model of the project, and thus to be able to use alternative editors for the model information. A graphic modelling language such as UML is complex and requires a good level of training. It can be preferable to provide the participants who are not familiar with UML, a view of the model, which is appropriate to them, using for example a text editor or a calculation sheet. In the same way, to create or update a model UML, it can be interesting for the participants to publish their contributions in pre-formatted documents (table or sheet). The creation and the update of the model and the UML diagrams are performed by specific importation tools starting from the meta-model.

The import/export functions really make the model polymorph and able to be presented, at least partially, in various formats adapted to the skill and the activity of each participant.

4. Manage model complexity

Gradually during the project development and whatever the selected development cycle is (incremental, iterative, etc), a growing number of data coming from different sources are integrated in the exchange model. The model must take into account information from new stakeholders (integration, quality, certification, validation, user support…) On the basis of a reference model, it should be possible to extract the most adapted views for a given type of user, for a development phase, for a delivery or for an activity. The objective is to show only relevant information, by reorganising them if necessary but while guaranteeing the total consistency of the model.

A monolithic exchange model in which all information of the project is centralised is useful to provide a global view of the project, but it is not adapted to perform a specific activity (validation, verification of the traceability…) It is necessary to separate the model in consistent sub-models easiest to manage, and ensure that the modifications performed in the sub-models are propagated towards the reference model.

MME allows the edition of several models in the same session. MME provides mechanisms, which make it possible to separate a model in a set of models. The links and the shared data are defined in the meta-model, so that the data can be defined in a model but referred in other models.

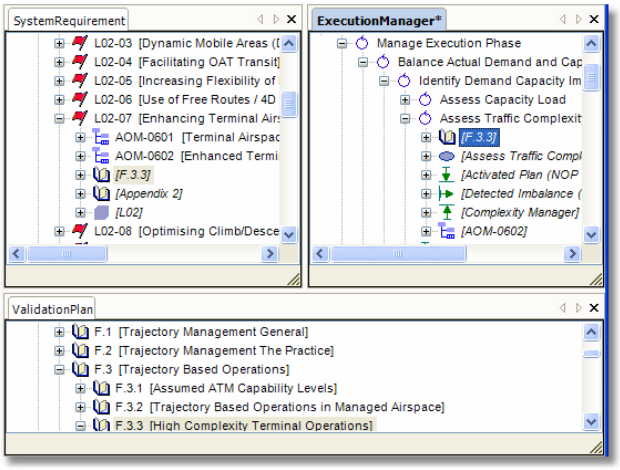

Figure 6: Links between models (figure provided by courtesy of EUROCONTROL)

For example, a system requirement is defined in the system design model and can be referred to in test cases, in change requests or in the software component which implements it. Each model can bring precision on a shared data.

MME provides filtering functions, which make it possible to create a subset starting from a reference model according to various criteria (direct selection or advanced search). It is for example possible to extract from the system design model, the use cases, the requirements and the interfaces associated with a software function, in order to create an autonomous and self-sufficient software specification model.

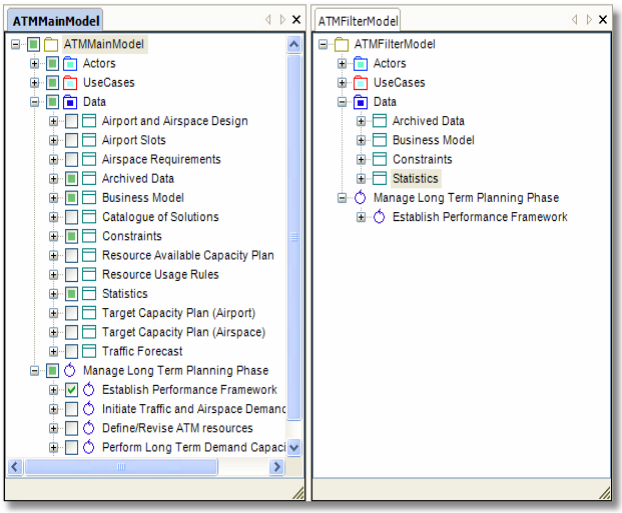

Figure 7: Filtering model (figure provided by courtesy of EUROCONTROL)

Thus the software team responsible for this function development gets a sub-model initialised with the only necessary information. In the same way, starting from the exchange model, it is possible to extract the use cases and the system requirements in order to initialise the validation model.

A UML/XMI export tool in MME allows the creation of a UML 2 model with the adequate profiles. Vice versa, a UML/XMI import tool allows the importation in the exchange model of the modifications done in a sub-model.

The separation of the reference model in an assembly of sub-models adapted to a specific activity makes it possible to control complexity, to specify in the meta-model the useful information for each activity and to reveal the shared data.

5. Traceability management

The traceability management is an essential activity of a model driven process. The definition of the links, which exist between the major data of the project, is performed in the meta-model. The traceability can be initialised automatically by the import tools but has to be completed explicitly by the various teams. For example, the traceability between a functional requirement and a more specific software requirement, a software use case or a software architecture component has to be established explicitly.

It is important to control the traceability links by specifying them in the meta-model, in order to avoid an anarchistic proliferation of the links. But it should be also possible to create “free” links between data, which are not linked in a meta-model, for the purpose of annotation as an example.

The users should be able to establish and extract easily the traceability links, for example:

- To retrieve the system requirements which are not covered by a software element or a validation test,
- To assess the impact analysis of a modification (impacted use case, interfaces, components or test cases)
- To obtain relevant metrics on a modification cost.

It should be noted that during the iterative cycle, it is necessary to synchronise the traceability links and to indicate the obsolete links. A component of the architecture model can refer to a requirement of the software model. In this case, the user can create the link by a simple drag&drop.

MME allows to link data which are not linked in their meta-model by means of undifferentiated links. These links are very useful to manage annotation models associated with a model.

MME provides various tools in order to display or to follow the links. It also provides queries to generate traceability matrixes and manage duplicated data.

The traceability management helps to clarify the links existing between documents of different formats, to be able to analyse the impact of a modification by going all over the chain of links.

Since the traceability information is defined at the level of the exchange model, it does not pollute the documents themselves.

6. Collaborative modelling

Project model elements are shared by different participants and in different activities. Each participant may have his/her own view on shared data. Shared data may be duplicated in different models or documents. Each activity may have its own lifecycle.

It is mandatory to manage and to synchronize the shared data modifications. A change control process needs to be implemented in order to master modifications of shared data avoiding concurrent access.

For example, the system design model defines the software components architecture, the responsibilities of each software component, the interfaces and the interaction between components. Software components may be realised by different teams. The validation model is built from the requirements and the use cases of the system design model.

The system design model is connected to its sub-models (software components models and validation model) but the sub-models are not connected. If a software team needs to modify a shared data or a responsibility, the modification has to be propagated to the system design model and, possibly, to the validation model. To avoid anarchic modifications, each model should be defined with an “exchange area”. When a shared data needs to be modified or to be added, it is moved in the “exchange area”, so the modification is traced and is local to the sub-model.

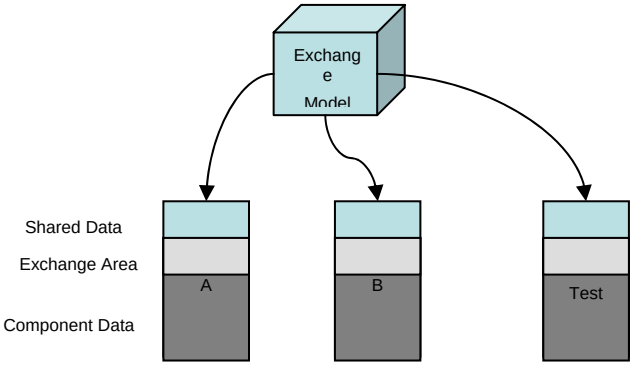

Figure 8: Sub-models structure

When the system model and the sub-models need to be synchronized, the tool analyses the exchange area of the sub-models and proposes a set of correction. Note that the system team may take into account new requirements. The modifications are propagated to the sub-models. The synchronisation process may be iterative, each team can analyse the impact of the corrections using traceability links and estimate modification cost.

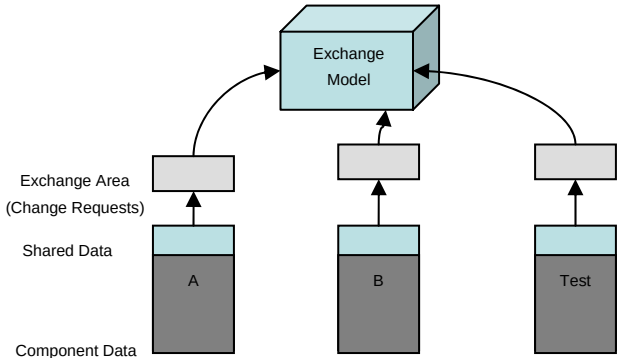

Figure 9: Extraction and Analysis of the sub-model exchange areas.

When the corrections are approved, the sub-models are updated in two phases:

- The modifications are propagated to the sub-models, new data are added, modified data are updated and marked as modified and removed data are marked but are not deleted. Data that depend from updated or deleted data are also marked using traceability in order to make easier the impact analysis. This process is realised by MME
- The model is updated by the responsible team using the update marks. Obsolete data and marks are manually deleted.

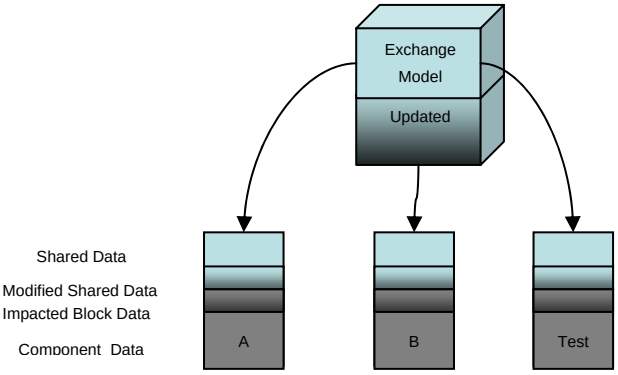

Figure 10: Propagation of the modifications to the sub-models

Note that the process may be recursive; a software component may be split in sub-components defined with their own model.

The Synchronisation process is supported by MME and is realized by comparing the initial model with the updated model and by generating synchronization instructions.

7. Evolutions of the model

During the first iterations, the exchange model contains the data that concerns the analysis, the architecture and the realisation of the project. Data associated to the integration, the validation, the deployment or the maintenance are gradually added to the exchange model, because the project focus moves. It is not appropriate to anticipate exchange model evolutions while the actual needs are clearly defined, because the meta-model might contain inappropriate data. Moreover, during the project lifecycle, the exchange model has to take into account new concepts, new stakeholders and new documents with specific interfaces. The size of the exchange model increases and it is necessary to restructure it in order to control model complexity.

It is necessary to update frequently the meta-model in order to remove obsolete data, to adapt existing data or to add new data and links.

Each time the project meta-model evolves, it is necessary to adapt the associated models using model transformations. In particular, a model transformation is realised when it is necessary to:

- Modify the structure of a model,
- Extend an existing model with new data and new internal links or new links with other model
- Filter a model because it is necessary to remove obsolete data

- Merge models, for example two software components are merged because they share common responsibilities
- Split a model in several models in order to master its complexity, for example if a software component is split in two components, the interfaces of the new components need to be defined and have to be propagated to the clients of the split components.

MME provides simple and easy to use mechanisms for model transformation.

- an export function allows to save the data contained in structured tables or spreadsheets, using the current version of the meta-model,
- an import function creates a model from the tables or the spreadsheets using the new version of the meta-model.

Complementary scripts may be necessary to modify the tables. Model transformation provided by MME should be improved in future versions [2].

Model evolution and adaption are critical issues, because, in a model centric approach, the exchange model has to evolve in order to take into account new data and new project participants. Model transformation is necessary when developments of a project are reused in a different context.

8. Conclusion

A first feedback of operational implementations shows that the multi-model process allows a better communication between teams: each participant works with his/her own tools or languages but can share a subset of the information he/she manages with other teams by the means of integration models. The tool provides an efficient support to modifications propagation among the different views. Above all, this approach allows to build multiple and consistent views of the system and thus to recover the control of the overall system.

MME is the result of a long experience in modelling and is currently used to develop large models.

The next step will be to industrialize the prototype tool MME in order to provide a robust and durable support to the elaborated process. The envisioned way is to develop MME in the frame of the open source engineering framework TopCased [5].

9. References

[1] Erwan Breton Jean Bézivin: "*Model-Driven Process Engineering*" Int. Computer Software and Applications Conf. (COMPSAC'2001)", Chicago, Illinois

[2] Jean-Marc Jezequel: *"A MDA approach to model and implement transformations",* Language Engineering for Model-Driven Software Development, number 04101 in Dagstuhl Seminar Proceedings. Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI), Schloss Dagstuhl, Germany, 2005.

[3] Jackson, A.; Geissel, M.; Dorbes: *"Supporting the collective process of controller working position development for ATM" Digital Avionics Systems, 2001. DASC. The 20th Conference Volume 2, Issue , Oct 2001 Page(s):7E3/1 - 7E3/12 vol.2*

[4] Ubayashi, Naoyasu; Sano, Shinji; Otsubo, Genya: *"A Reflective Aspect-Oriented Model Editor Based on Metamodel Extension", Modeling in Software Engineering, 2007. MISE apos;07: ICSE Workshop 2007. International Workshop on Volume 20-26 May 2007*

[5] Marc Pantel, ACADIE team, OLC team, and TOPCASED team *" The TOPCASED project - a Toolkit in OPen source for Critical Applications and SystEms DesignDocument Actions "*

10. Glossary

API: Application Programming Interface
ATM: Air Traffic Management
MME: Multi-Model Editor
UML: Unified Modeling Language
XMI: XML Metadata Interchange
XML: eXtensible Markup Language