A Model-Driven Approach For Information System Migration

Raymonde Le Delliou[1], Nicolas Ploquin[2], Mariano Belaunde[3], Reda Bendraou[4], Louis Féraud[5]
1 Electricité de France, 2 SOFT-MAINT, 3 France Télécom, 4 LIP6, 5 IRIT
1 juliette.le-delliou@edf.fr, 2 nploquin@sodifrance.fr, 3 mariano.belaunde@rd.francetelecom.com, 4 Reda.Bendraou@lip6.fr, 5 Louis.Feraud@irit.fr

Abstract

In 2002 and 2003, the TRAMs project, led by a consortium of French companies and universities, experimented the application of emerging model engineering techniques to Information System (IS) migration. The main objectives were to define a methodology and to specify an open and modular migration framework to solve IS migration complexity on the basis of models and meta-models, as well as model transformation. A concrete result of this project was the development of a demonstrator that performed the migration of a large COBOL legacy application of an insurance company to use a JAVA and HTML based interface. In this migration use case, we used a common intermediate meta-model based on ISO/RM-ODP (Reference Model of Open Distributed Processing) extended with the UML Action Semantics formalism.

Keywords—model driven architecture, modeling, meta-modeling, model transformation, Information System migration, legacy code.

1. INTRODUCTION

To deal with increasing competition, companies have to face constant strategic, organizational or technical changes which imply repetitive modifications of their Information Systems (IS). The massive introduction of internet technologies is one example. Though IS migration solutions exist on the market, all these solution are ad-doc solutions that are difficult to reuse and most of them rely on proprietary tools.

This paper will describe the experiments made in a cooperative French project called TRAMs to solve some of the challenges of IS modernization, by using extensively *model-oriented* engineering techniques. In particular there is an attempt to integrate RM-ODP with Action Semantics to enable fine-grained representation of the applications an enterprise may wish to migrate.

TRAMs, was launched in 2002 and was partially funded by the French government in the context of RNTL national research program. The consortium comprises:

- two French industrial groups, France Télécom and Electricité De France, having to migrate large legacy applications,
- two universities, LIP6 Paris 6 and IRIT Paul Sabatier Toulouse, known for their works on modeling, meta-modeling, languages and model transformation,
- SOFT-MAINT, a company specialized in large-scale IS migration projects.

One of the primary objectives of TRAMs's was the definition of a methodology and the provision of a generic architecture to help mastering IS migrations. In particular our ambition was to ensure that it is possible to reuse the good practices on migration technology in multiple and repetitive migration projects. The more efficiently an enterprise organizes knowledge reuse for their Information System (IS) evolution, the less it would cost at the end to integrate the new technologies, even if the initial investment may be high. One of the important issues also addressed by the project was the problem of maintenance after a migration occurs. Is maintenance facilitated when models are first class artifacts?

The approach taken by TRAMs to reduce IS migration complexity was to decompose a migration as a sequence of transformations that apply on models. A model represents an abstraction of an input, an output or an intermediate result. Some of these transformations may involve a lot of "manual" human operations – such as reverse engineering– while others may be totally automated. The inputs and outputs of IS migration were re-formulated and formalized in terms of meta-models in order to apply model engineering techniques, like model-to-model transformation or code generation. In short, transforming an IS comes down to transforming models of the IS.

TRAMs makes use of various OMG modeling standards, such as MOF and UML for model representation [MOF] [UML], XMI for tool exchange [XMI], SPEM for migration process and process patterns description [SPEM] also In addition it used a combination of the ISO/RM-ODP standard [ISO95] [ISO2002] and UML's Action Semantics to define an intermediate common enterprise model capable of expressing low-level computations. The advantage of using these standards, instead of proprietary formalisms, is easy to understand if a company aims to take advantage of present or future products in the market place. However, our methodology and architecture does not enforce the usage of these standards as long as other formalisms are able to play the same role.

In the context of this work, we will assume the following definition of a model: a model is a description or a specification of a system and its environment for some certain purpose [MDAG]. In line with actual meta-modeling principles, a model will always be defined in the terms of a meta-model. For instance, a complete RM-ODP specification can be formalized as a collection of one or more models, each of them being defined in relation to specific meta-models, such as a meta-model for the enterprise view and another for the computational view.

2. METHODOLOGY AND FRAMEWORK OVERVIEW

In order to define a general methodology for information system migrations, TRAMs project has:

- Firstly, provided a typology of migration projects according to the kind of information that is impacted, For instance, we have migrations that imply changes on the business process, and/or in the applications interfaces and data. Some migrations are on technology oriented while other are fundamentally driven by a business change.
- Then, identified and implemented various process patterns, which may or may not include the usage of common intermediate formalism, which potentially can be reused in various migration projects.

An important outcome of the project was the classification of migration activities that are part of the *preparation* from those that are part of the *execution* of the migration.

In parallel to this work on methodology, the TRAMs project has specified an architecture – the so called TRAMs framework – that allows managing successive migrations based on the usage of model repositories and model transformers.

To define a migration framework one has to think about the relevant components needed:

- to prepare the migration,
- to execute the migration.

Preparing a Migration implies the ability to represent and store a description of the migration process to be performed. In the project, we used the SPEM formalism and notation and a MOF-based repository capable to store SPEM models.

Within the preparation phase, after designing the migration process model, one has to think how to implement each process activity. This was done by enriching the process description with the so-called "instrumentation model" which indicates what software components are used in each activity, or what kind of manual actions need to be done for the activity to be fulfilled. Storing a model of the implementation (the instrumentation model) ensures that information is not lost. This information can be re-used when a new migration has to be performed.

Migration preparation involves also the definition and/or the identification of the meta-models to represent the inputs or the outputs of the IS migration. Finally, each transformation component needs to be specified and implemented. In the case of a model-to-model transformation it is possible to use an executable specification language. Typically, a generic transformation engine will execute transformation specified as a list of rules. Section IV presents some of model transformation techniques experimented within this project.

The kinds of tools that are used during the execution of the migration are typically:

- Specialized reverse engineering tools – such and parsers and pattern analyzers – to scan the available legacy code and to discover any relevant high-level structure,
- CASE tools, with graphical capabilities, such as UML tools,

- Model repositories to store and publish the input, output or intermediate models,
- Model-to-model transformers and code generators.

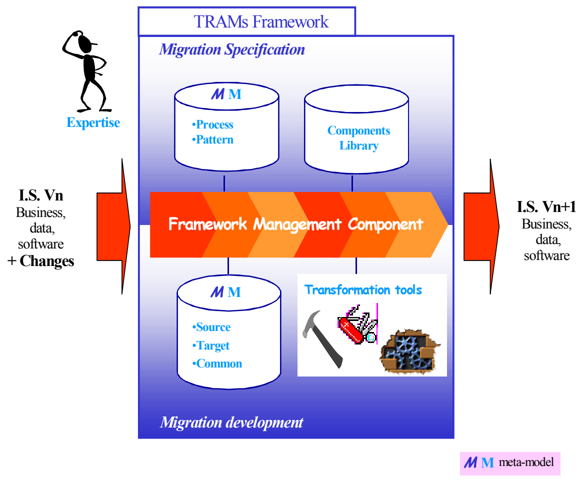

Figure 1: TRAMs Framework

The Figure 1 depicts the TRAMs framework architecture. On the top are showed the tools that are used for preparing the migration and in the bottom the tools used when executing the transformation. In the middle, a management component centralizes all information needed by a user to control and monitor the migration process.

To summarize, the TRAMs framework is **generic** and **open.** It does not impose any process or tool: each company can define its own migration process, use their own proprietary formalisms and connect their own tools. However, in order to take plain advantage of model-oriented engineering techniques it is important to use as much as possible data representations that are based on meta-models.

Meta-modeling is the key to achieve tool fine-grained inter-operability, as opposite of interface-based inter-operability that permits in most cases only coarse-grained inter-operability. For instance, any tool that supports SPEM models storage can play the role of the process repository as long as it supports import/export functions. In addition, any model transformer that knows the type of the input and output models and that implements the transformation specification rules can play the role of a transformer.

An initial implementation of the TRAMs framework principles was achieved by the end of 2002. This initial demonstrator performed the migration of the GUI capabilities of a large COBOL-based insurance application into two distinct targets: one based on HTML and Java script and a second one based on Java applets technology. The final demonstrator, presented in 2003 demonstrated the migration of business computational parts of the application. For that purpose an intermediate meta-model based on RM-ODP and Action semantics was used (see section III).

3. MODELING THE MIGRATION PROCESS

An example of a typical pattern for a migration project is depicted in Figure 2. This pattern results from the experience of the industrial partners involved in TRAMs..

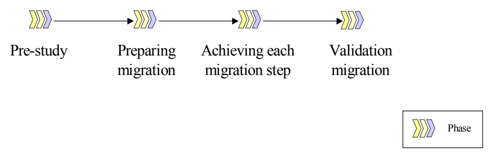

Figure 2 : TRAMs process-type

The pre-study is a phase in which the relevance for making a migration is done. Sometimes it is better to get rid of an obsolete application simply by re-written it completely from scratch. Sometimes, however, this is un-realistic, simply because no one is capable to understand the business rules that are hidden in an obsolete application!

In the "Preparing migration" phase, one has to:

- Identify the migration nature to measure migration complexity. Is it a business or technologic evolution? Does it imply changing the data storage, the user interfaces, the network elements, and so on.
- Identify – and if needed specify - the meta-models to be used during the migration. In the TRAMs 2003 final demonstrator, we used a COBOL meta-model in conjunction with a BMS meta-model to represent the source data. The BMS – *Basic Mapping Support* - is the transactional GUI system used in the insurance application. In addition, we used a meta-model to represent RM-ODP augmented with UML action semantics concepts.
- Model the migration process in SPEM, and try to reuse any pre-existing process pattern (if applicable). By modeling the process one may decide on the numbers of models that are to be managed separately – for instance the GUI aspect

may be put in a separated model in order not to pollute pure business data models. At this stage also, one may decide to use an intermediate meta-model that the company may want to use for various successive migrations.

The Figure 3 depicts the pattern that was used in the final demonstrator for the execution of the migration. . First, we have a reverse-engineering phase, then the input models are transformed – most automatically - in terms of the "neutral" intermediate meta-model. Then this intermediate representation is re-worked using heuristics and manual annotations into a more conceptual model. Finally this re-worked representation is used as the input for generating the artifacts needed by the target platform.

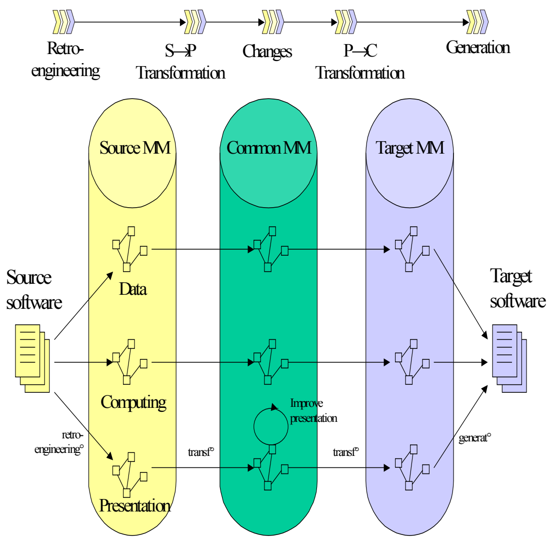

Figure 3: Process pattern for the migration of a COBOL Interface to Java/HTML

When a company needs to perform multiple and repetitive migrations, a common intermediate meta-model helps to:

- Decrease significantly the number of model transformers to be developed (for all the migrations to be done),
- Decrease times and costs of the information system evolution during further migrations. Thus, it may be possible to reuse the models stored in previous migrations as well as to reuse the transformers).

On the other hand a dedicated source-to-target solution may be much simpler to develop, simply because we can concentrate directly on the mapping rules that are needed in the specific migration case.

To summarize, the determining decision-making factors are:

- The number of similar migrations expected to be performed (or if it is a one-shot),
- The durability of the target applications, if they are to be maintained and the length of maintenance period,
- The availability of a transformer. In such a case, we can easily understand that we will tend to favor the option that reuse the available transformer rather than the option that requires developing new ones.

4. A COMMON INTERMEDIATE MIGRATION MODEL BASED ON RM-ODP AND ACTION SEMANTICS

The common intermediate meta-model used in the final TRAMs demonstrator is a collection of models based on ISO/RM-ODP (Reference Model of Open Distributed Processing) and on OMG's UML Action Semantics.

RM-ODP supplies the proper concepts for distributed computer system specifications. RM-ODP is based on an object approach. The system is described from five complementary viewpoints [IEEE-1471] [PUT], covering as well business aspects as the most technical aspects.

Identifying those viewpoints allows system specification to express at the same time but distinctly: the business the IS supports (Enterprise Viewpoint), the way it is modeled in the computer system regarding information and functions (Information Viewpoint, computational Viewpoint, Engineering Viewpoint) and the technical choices of the computer system mapping user requirements (Engineering Viewpoint, Technology Viewpoint).

The key points of RM-ODP are the sufficient completeness of its concepts and structuring rules and the relevance of its abstraction levels. In this way, the software architecture of the system to be built can be well specified using this set of concepts [BGL99] [BGL01] [BL01] [ODAC] [DASIBAO].

However, these abstractions are quite high level ones and do not allow to express the very detailed information that can be found in a software code. As an example, RM-ODP does not allow to specify actions connected with operations of a class nor does it allow specifying with enough detail their underlying semantics. Migration applications clearly bring out the need for formalizing coding instructions, while remaining standard and platform-independent. Action Semantics turned out to be the right candidate for this criterion.

We then chose to integrate Action Semantics meta-model with RM-ODP meta-model.

We identified shared or common concepts from each standard, and displayed a logical continuity from one standard towards the other, i.e. from the more abstract concept towards the more detailed concept.

As an example, we will focus on the computational viewpoint, which is the central element of our two meta-models integration. Junction points were identified to achieve the coupling of this RM-ODP viewpoint with Action Semantics [SERP04].

If we consider the RM-ODP concepts of Action and Operation, according to the ODP standard, an Object has a Behavior, which is defined by a set of Actions and an Action is "something which happens".

In Action Semantics, the Action concept is a first class one, and is actually the fundamental concept of the Actions package. An Action is defined, in Action Semantics, as "a behavioral specification that acts on inputs to produce outputs". Even if both definitions do not strictly express the same semantic, it seems obvious that the ODP definition is more abstract and includes the one given by Action Semantics standard.

On the other hand, in ODP, an Interface is composed of Interactions, all of the same type (*Signal*, *Operation* or *Flow*). An Operation has a signature defined by a name as well as parameters. In UML, an *Operation* (defined in the Core package) corresponds to the specification of a *Method*. It has a name and zero or n parameters inherited respectively from the *ModelElement* and *BehavioralFeature*, abstract classes. A *Method* is implemented by a *Procedure* (package Actions/Action Foundation). Then we can easily map the ODP *Operation* concept to the UML *Operation*.

On Figure 4, the "Operation" rectangle represents both the UML and the ODP Operation concepts. This unique rectangle represents our first junction point and enables us to link the two meta-models. The second junction point is represented by the "Action" rectangle, which also links the two meta-models.

The final meta-model integrating ODP Computational viewpoint meta-model and UML Action Semantics meta-model (Figure 4) is now complete and covers both the software architecture and the coding instructions allowing a behavior specification, thus opening the way towards a complete code generation from models.

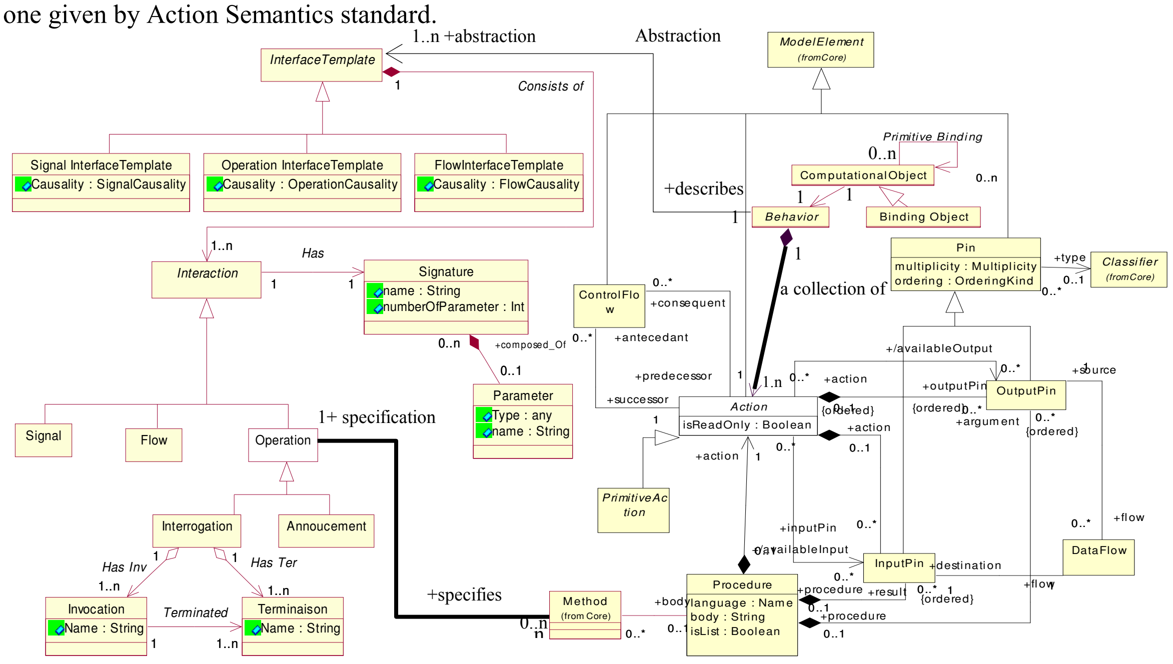

Figure 4: An extract of the final meta-model coupling RM-ODP Computational viewpoint and UML Action Semantics

5. Using model transformation techniques for translating Cobol programs into Java

The transformation process relies on model transformation techniques. TRAMs partners are involved in the OMG initiative for standardizing model transformation techniques known as MOF Q/V/T Request for Proposal. In this context, some of them already developed their own transformation engine.

TRAMs project thus has experimented three types of transformers, which are all based on rules expressing correspondence between concepts of two meta-models or within a meta-model:

- Model In Action (MIA), based on a SOFT-MAINT tool: the language is fully declarative and is based on predicate logic, ,
- TRL – formely called MTRANs - , based on a France Télécom tool: the rules have a declarative signature and an imperative body [BEL],
- Attribute Grammars: the rules are mathematic expressions in SSL [KAST] [KNUTH].

The rules are executed by transformation engines in compliance with the structure shown in Figure 5.

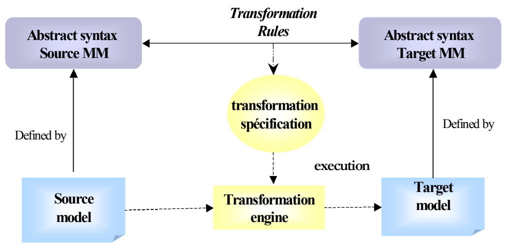

Figure 5: Model transformation process

All three approaches allow composing transformations, to trace concept evolution during transformation, to reuse the transformer or at least part of it. Only Attribute Grammars allow inferring modifications automatically, whereas MIA and TRL are the ones that can offer a user friendly concrete notation.

As shown in section II, a migration is typically divided in three sub-processes. As an example, we present here the transformation of computing information from COBOL into Java which was part of the 2003 demonstrator (Figure 6).

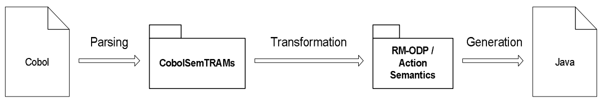

Figure 6: Transformation sub-process for computing information

This transformation is composed of several steps. The first consists in parsing source code from Cobol programs. This operation also consists in selecting relevant computing information. We extract control flow models by use of slicing techniques. Result models are based on a platform specific meta-model for Cobol source code, CobolSemTRAMs. This meta-model may be considered as an action meta-model and define concepts such as CobolProgram, CobolVariableData, CobolProcedure, If, While, Affectation, Comparison, etc.

The following step transform Cobol-oriented models into platform independent models based on the RM-ODP/Action Semantics meta-model presented in §III. Transformation rules are expressed by mapping COBOL meta-model concepts to Action Semantics meta-model ones.

Few mapping rules:

- CobolProgram → Class
- CobolVariableData → Attribute
- CobolProcedure → Method
- CobolStatementBlock → GroupAction
- If → ConditionalAction
- While → LoopAction
- Affectation → AddAttributeValueAction
- ArithmeticExpression → ApplyFunctionAction
- Comparison – ApplyFunctionAction

Action Semantics concepts are quite similar to Java actions concepts. Therefore, we decided to also consider the RM-ODP/Action Semantics meta-model as a target meta-model for this transformation sub-process. The last step thus consists in generating the Java source code from intermediate models. Each Cobol program is translated as a Java class.

The resulting program is merged with Java code generated by other transformation sub-processes. Final programs may be built and contain all required functionalities extracted from Cobol programs. However, the generated Java code stays excessively "Cobol-oriented". Transformation rules may be improved to get better Java programs.

6. Conclusion and further works

The TRAMs project has demonstrated how to take advantage of the emerging model-oriented techniques to facilitate successive migrations within a large company. As we mentioned in the introductory section, the cost of building a model-aware migration framework may be high, but return of investment can be quickly be positive if the company has to perform various successive migrations to take advantage of the new technologies.

The TRAMs project final results were mainly a methodology and architectural principles that helps to apply the best practices for migration projects – like well-known migration patterns. The experiments achieved during this project have allowed SOFT-MAINT to improve their migration tools and to wide its offer towards other migration markets on which the company already has know-how stored in models (FORTRAN for example). We should note that TRAMs feedback in model transformation has also influenced the OpenQVT response (the so called "French submission") to the OMG's Q/V/T RFP.

This work has also made a fruitful attempt to combine the high-level architectural concepts of RM-ODP with low-level computational constructs bring by the Action Semantics standard. This integration has permitted us to use directly RM-ODP as an intermediate language for the representation of the applications that are part of the information system of an enterprise, and which could be subject to maintenance and evolution.

We believe that there is still a lot of research work that needs to be done in the field of information system migration. In particular, we need to study and classify the strategies that can improve the analysis of the code that is reversed as a model. The difficult part of a migration is still the ability to produce high-level models from low-level code.

GLOSSARY

BMS:	Basic Mapping Support
EDOC :	Enterprise Distributed Object Computing
ISO :	International Organization for Standardization
MDA :	Model Driven Architecture
MOF :	Meta Object Facility
OMG :	Object Management Group
PIM	Platform Independent Model
PSM	Platform Specific Model
Q/V/T :	Query/View/Transformation
RFP :	Request For Proposal
RM-ODP :	Reference Model of Open Distributed Processing
SPEM :	Software Process Engineering Meta-model
UML :	Unified Modeling Language
XMI :	XML Metadata Interchange
XSLT :	eXtensible Stylesheet Language Transformation
W3C :	World Wide Web Consortium

REFERENCES

[BEL] M. Belaunde, M. Peltier: From EDOC components to CCM components: a precise mapping specification (2002). *In Proc. ETAPS 2002*, Grenoble, France. LNCS 2306 Springer (2002)

[BGL99] X. Blanc, M.P. Gervais and R. Le Delliou, "Using the UML Language to Express the ODP Enterprise Concepts", in Proceedings of the 3rd International Enterprise Distributing Object Computing Conference (EDOC'99), IEEE Press (Ed), Mannheim, Germany, September 1999

[BGL01] X. Blanc, M-P. Gervais, R. Le Delliou "On the Construction of Distributed RM-ODP specifications" Proceeding of the Third IFIP International Working Conference on Distributed Applications and Interoperable Systems (DAIS 2001).

[BL01] X. Blanc, R. Le Delliou, "Information System architecture with RM-ODP: an on-the-field experience" Proceeding of the Open Distributed Processing: Enterprise, Computation, Knowledge, Engineering and Realisation (WOODPECKER 2001). pp27-37. June 2001.

[DASIBAO] A. Picault, P. Bedu, J. Le Delliou, J. Perrin, B. Traverson «Specifying an Information System architecture with DASIBAO, a standard based method", Proceeding of the International Conference on Enterprise Information Systems (ICEIS 2004)

[IEEE-1471] IEEE « Recommended practice for architectural description of software-intensive systems » IEEE Std 1471–2000.

[ISO95] ISO/IEC IS 10746-x, ITU-T Rec.X90x Open Distributed Processing-Reference Model Part x, 1995.

[ISO2002] ISO/IEC IS 15414 Open Distributed Processing Reference Model –Enterprise Language, may 2002

[KAST] U. Kastens: Ordered Attributed Grammars. *Acta Informatica* (1980), 13(3): 229-256

[KNUTH] D. Knuth: Semantics of context free languages. *Mathematical Systems theory* (1968)

[MDA] « Model Driven Architecture – Architecture board ORMSC » – document number ormsc/2001-07-01 – OMG-2001

[MDAG] « MDA Guide » – document number ab/2001-01-03 – OMG-2003

[MOF] OMG. "Meta-Object Facility (MOF) Specification v1.4". TC. Document formal/02-04-03 OMG. April 2002. http://www.omg.org

[ODAC] M.P. Gervais, ODAC : An Agent-Oriented Methodology Based on ODP Journal of Autonomous Agents and Multi-Agent Systems, Kluwer Publishers (Jan. 2002)

[PUT] J-R. Putman, « Architecting with RM-ODP », Prentice-Hal, 2001

[SPEM] Software Process Engineering Metamodel, Draft Adopted Specification, November 2001

[SERP04] R. Bendraou, S. Bouzitouna and M. P. Gervais, From MDA Platform-Specific Model to Code Generation: Coupling of RM-ODP and UML Action Semantics Standards, to appear in Proceedings of the International Conference on Software Engineering Research and Practice (SERP'04), Las Vegas, USA, June 2004

[UML] : "OMG Unified Modeling Language Specification", Object Management Group, March 2003, OMG TC Document UML1.5 (Action Semantics) formal/03-03-01, www.omg.org

[XMI] OMG "XML Metadata Interchange (XMI) v1.1". TC Document ad/99-10-02 OMG. 1999. http://www.omg.org

[XML] Extended Markup Language Version 1.0, W3C Recommandation
http://www.w3.org/TR/1998/REC-xml-19980210.pdf