Trust in MDE Components: the DOMINO Experiment

Benoit Baudry, Pierre Bazex, Jean-Charles Dalbin, Philippe Dhaussy, Hubert Dubois, Christian Percebois, Erwann Poupart, Laurent Sabatier

Trust in MDE Components: the DOMINO Experiment

Benoît Baudry[1], Pierre Bazex[2], Jean-Charles Dalbin[3], Philippe Dhaussy[4], Hubert Dubois[5], Christian Percebois[2], Erwann Poupart[6], Laurent Sabatier[7]

(1) IRISA, Campus universitaire de Beaulieu, 35042 Rennes, France
(2) IRIT, Université de Toulouse, 118 route de Narbonne, 31062 Toulouse Cedex 9, France
(3) Airbus France, 316 route de Bayonne, 31060 Toulouse Cedex 3, France
(4) ENSIETA, 2 rue François Verny, 29806 Brest Cedex 9, France
(5) CEA, LIST, Point Courrier 94, Gif-sur-Yvette, F-91191, France
(6) CNES, Toulouse, 31401 Toulouse Cedex 4, France
(7) Sodifrance, 15 chemin de la Crabe, 31300 Toulouse, France

bbaudry@irisa.fr, bazex@irit.fr, jean-charles.dalbin@airbus.com, Philippe.Dhaussy@ensieta.fr, hubert.dubois@cea.fr, percebois@irit.fr, erwann.poupart@cnes.fr, LSabatier@sodifrance.fr

ABSTRACT
A large number of modeling activities can be automatic or computer assisted. This automation ensures a more rapid and robust software development. However, engineers must ensure that the models have the properties required for the application. In order to tend towards this requirement, the DOMINO project (DOMaINs and methodological prOcess) proposes to use the so-called trustworthy Model-Driven Engineering (MDE) components and aims to provide a methodology for the validation and qualification of such components.

Categories and Subject Descriptors
D.2 SOFTWARE ENGINEERING
D.2.1 Requirements/Specifications, D.2.2 Design Tools and Techniques, D.2.3 Coding Tools and Techniques, D.2.4 Software/Program Verification, D.2.5 Testing and Debugging, D.2.6 Programming Environments

General Terms
Design, Reliability, Verification, Languages, Experimentation.

Keywords
Model-Driven Engineering Trust, Component, Requirement, Domain-specific languages, OCL contracts, Proofs, Mutation analysis, Transformation tests, Transformation traceability.

1. INTRODUCTION
Software development is based on a complete set of methods and tools for design, integration and verification. In the past, the role of models was limited to documents, but now, these abstract and simplified representations of a system become directly involved in the development.

This observation has led to the creation of a new paradigm in software engineering, Model-Driven Engineering (MDE) [6]. It places models at the core of software development by making them explicit assets that can be manipulated by programs. The resulting development processes can therefore be considered as a succession of models, at different levels of abstraction, from the definition of requirements to the development of its actual code for operation.

The DOMINO approach explicitly aims to increase the level of trust [7] in software systems developed with a model-driven approach. DOMINO proposes a set of concepts and techniques to domain experts so that they can define and establish their own model-based process to leverage models and model transformations and build trustable systems. We present two complementary aspects: the validation of models produced at the different steps of the process and the development of trustworthy components that completely or partially automate a development step.

The proposed methodology to establish trust in MDE components is mainly based on the consistency between three characteristics: specification, implementation and verification. We propose various solutions to increase trust in the transformations, such as keeping a relationship between the models of the application and their verification, in order to test or prove them. The process tends to be as continuous as possible offering model follow-up functions throughout the different steps of an MDE process.

Along these processes, experts use specific models related to their field. In particular, for embedded systems, software engineers have defined specific modeling and programming languages to better integrate the critical limits of operation safety. Introducing the characteristics of such languages into abstract models enables specialists to anticipate, in very early development stages, the solutions that are the most appropriate for their applications. As part of the MDE approach DOMINO transfers paradigms that are well established in the theory of programming languages. Experts

S&D4RCES 2010 September 14, 2010, Vienna, Austria.

in the field can then adapt their development processes making them more representative.

Two case studies were used to experiment with the methodologies and the techniques proposed by DOMINO: the specification, development and implementation of operational procedures in the ATV cargo craft Jules Verne (CNES) and a synchronisation function of two flight command units (Airbus). In both cases it was possible to set up an MDE process integrating the activities of transformation and verification of the models. In addition to using these models, the DOMINO technology contributes to the definition of a reasonable balance, for each of the activities of the process, between human expertise and the tool that replace that expertise.

2. RELATED ISSUES

Model-Driven Engineering (MDE) improves the capitalization of design know-how, the reuse - on an abstract level - of development artefacts and the control of complexity by means of the unifying framework that the models create. However, the lack of robust tools for model manipulation and management, of well-defined domain specific languages and associated technologies, the issues related to separation of concerns in models and the lack of validation techniques of models and model transformations are critical barriers to a wide industrial adoption of MDE. Thus, current processes use conventional software engineering processes and tools and suffer from interpretation and implementation errors. In most cases, verification occurs after the development phase. As a consequence, verification, correction and maintenance require a large human effort, whereas MDE tends to automate critical development steps in order to increase quality by construction, thus decreasing the effort needed at the end of the development.

These activities should be reconsidered and more attention should be paid to the new possibilities in modeling and meta-modeling offered by MDE. Efforts should now be focused on the requirements modeling and on ensuring the continuity in the development process. In turn, the process can be implemented by means of model transformations and intermediate verifications.

It is in this context that DOMINO aims to increase the quality in software development by offering rigorous methods and associated tools to evaluate and improve trust in basic components. Below, we define trustworthiness as a set of guarantees for the component. Engineers can therefore make the component either totally or partially responsible for an activity in a model-based process. More specifically, our study involves an MDE development process. It approaches the questions of model quality improvement and the trust in the transformations of models that perform development stages automatically.

3. TECHNICAL APPROACH

Trust in a software system is supported both by the validation of the models built along the development cycle and the use of trustworthy MDE components that automate some development steps. We propose modeling and verification techniques for model validation. We also propose a model for MDE components based on the integration of three characteristics: its specification, its implementation and the techniques of V&V (Verification and Validation).

3.1 Validation of models

This part of the study aims to ensure the relevance of the models produced during the development process. Validation comes down to seeking a certain form of completeness with respect to the system being modeled and thus to link the models to expected or perceived reality. The experiments on the two DOMINO case studies (CNES and Airbus) more specifically involved checking the requirements and the modeling of so-called domain-specific languages, as illustrated by Figure 1.

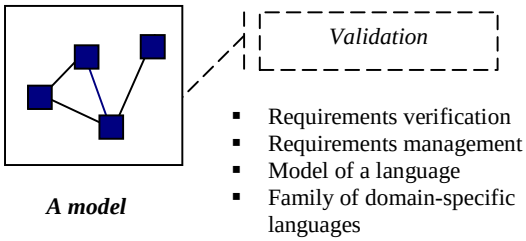

Figure 1. DOMINO's techniques to validate a model

3.1.1 Verification of requirements

A well known challenge in the formal methods domain is to improve their integration with practical engineering methods. In the context of embedded systems, model checking requires first to model the system to be validated, then to formalize the properties to be satisfied, and finally to describe the behaviour of the environment. This last point which we name as the proof context is often neglected. It could, however, be of great importance in order to reduce the complexity of the proof. The question is then how to formalize such a proof context.

In DOMINO, we experimented a language, named CDL (Context Description Language) [8] [9], for describing a system environment using actors and sequence diagrams, together with the properties to be checked. The properties are specified with textual patterns and attached to specific regions in the context. The idea behind context is that the requirements one want to verify are often linked to specific use cases so that it's not necessary to explore all possible scenarios on the system. Contrary to classical model checking methods where the system is explored in its entirety, the CDL language aims at reducing the system behaviour before its effective verification by interfacing with an existing model checker. The context description thus contributes to reduce the complexity of the system bypassing the state explosion. CDL is designed so that formal artefacts required by existing model checkers could be automatically generated from it. This generation is currently implemented in our prototype tool named OBP (Observer Based Prover) [8] [9].

3.1.2 Requirements management

The models must be validated with respect to the requirements. A requirements meta-model has been proposed to define the requirements and manage their traceability [10]. The meta-model takes the form of a dedicated profile called DARWIN based on SysML and includes tooling for requirements management in the Papyrus environment [25]. In addition to the annotation and the traceability links, the accent is placed on the coherence of the requirements model, the solution model, and the requirements V&V. The DARWIN profile has been experimented in the MDT Papyrus [25].

3.1.3 Model of language

DOMINO contributes to the integration of Domain-Specific Languages (DSL) into MDE developments. We propose an intermediate modeling step to adapt design models before code

generation, by establishing, in terms of models, a clear and unambiguous definition of the grammar of the target programming language [12] [13]. The integration of models based on the grammar of the languages enables software development on a precise definition of the links between the models and the programs. Two levels of assistance are proposed: one, which is original, in the BNF concerning the grammar of a language and another, which is more conventional, in the language model concerning the program models. It then becomes possible, at these two levels, to add domain specific properties.

3.1.4 Family of domain-specific languages

Any given professional field usually has a family of DSL. Developers are supposed to speak the same single language. In practice however, each group of specialists rewrites the concept of that sector of activities in his own domain-specific language. In order to ensure the interoperability of the domain-specific components, we consider the formal semantics of each of the languages. This formal specification, that covers both the syntax and the semantics of each DSL, is considered as an object in the category of algebraic specifications. Using results from category theory, we obtain semi-automatically, the definition of a language that can unify the family, as well as translators for the initial languages [1] [2] [19] [20]. The same theoretical results ensure that it is possible to transpose and to automatically prove a property from one DSL towards the unified language. This approach was validated for the operational procedure languages used in the space industry.

3.2 Trust in MDE components

An important goal in DOMINO is to control the development of an MDE component that automates a step of the process in order to measure its level of trust. Generally speaking, the notion of trust has many facets. It takes various forms depending on the level of abstraction considered and is enforced either by construction, or by redundancy. For model transformation for instance, trust can be enforced by construction through the use of transformation specific formalisms that can reduce the risk of errors. On the other hand, redundancy aims to keep track of requirements in relation with the implementation in order to prove and test the transformation.

We propose a component model represented by the triangle in Figure 2 that has three vertices: its specification, its implementation and its associated assets for V&V. DOMINO technologies aim to improve the specification and V&V protocols for MDE components and contribute to the overall consistency of the components' facets.

The process for building trust in a MDE component consists in three steps. The process starts with an initial component that already has its three facets ready. The goal is then to improve each facet and to check the global consistency between facets. It can be noted that this triangular structure model can be used at each of the stages of the engineering process.

For instance, considering a model transformation encapsulated into a MDE component, we are able to estimate the consistency between contracts, implementation and tests using mutation analysis as the main qualification technique. The process to improve the trustability of MDE components consists improving the test set by analyzing their efficiency using a mutation analysis, improving the implementation, thanks to the previously evaluated test set, and finally improving the contracts by measuring their accuracy as embedded oracles [17].

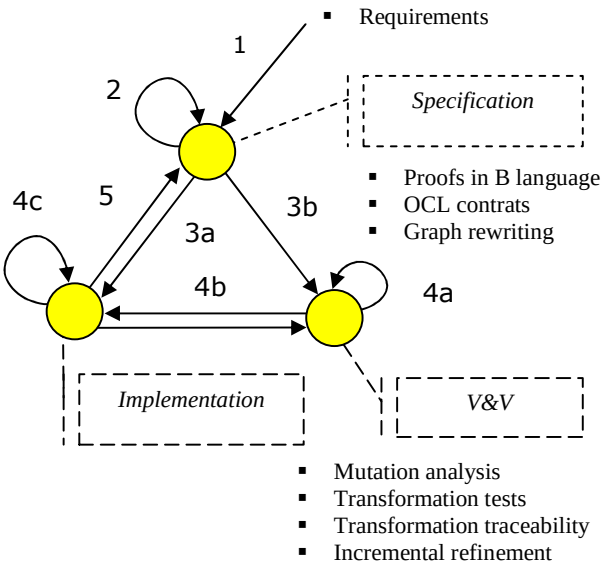

Figure 2. A trustworthy MDE component

For each facet of the triangle view of the MDE component, techniques have been defined and proposed by DOMINO, and were tested on the CNES and Airbus case studies. They apply distinct levels of formalization – from semi-formal, such as model transformation testing, to the formal level with the proof in B of the transformation and contracts. We then evoke more prospective studies concerning the rewriting of graphs and the refinement of models more akin to development proved by construction.

3.2.1 Proof in B

We experimented the B language to formalize the meta-models and the transformation rules from SOLM to O2PL for the CNES case study [11]. The formalism helps to express the meta-models and the source and target models by data elements (sets, constants and variables) and by the predicates defining the properties of these elements (B invariants). Each transformation rule is modeled as an abstract B operation which defines a precondition and a substitution. The B precondition formalizes the condition for applying the transformation rule, while the substitution part formalizes the target elements. The transformation specified in B enables the use of the B-proof assistant to analyze and prove the transformation formalization consistency with respect to the meta-models and transformation invariants. This proof in B is done on the specification of the transformation.

3.2.2 OCL contracts

These contracts are useful to test model transformations as they represent an executable version of the specification and enable errors to be detected at runtime [18]. It is therefore possible to dynamically ensure that the component accepts the models that are processed and, in turn, produces correct models. With DOMINO, the contracts concern the syntax of the meta-models and source and target models of the transformation as well as its behavioral semantics. They can be written in OCL-Kermeta by using the aspects mechanism to dissociate the structural from the behavioral parts of the transformation, or in pOCL (procedural Object Constraint Language) [15], an extension of OCL that supports simultaneous manipulation by several models and allows inter-model correspondences.

3.2.3 Mutation analysis

Originally used for the qualification of a set of test models, we use mutation analysis to measure the level of trust of the transformation component [16]. The technique is based on the creation of erroneous versions of the transformation to check how the test models behave. The erroneous versions, called mutants, are confronted with the contracts that implement an executable form of the specification embedded in the component. We adapt this technique for model transformation with new fault models that capture the errors specifically found in transformations. Such faults are related to the navigation and the filtering of the source and target models and on the creation of the target model.

3.2.4 Transformation tests

Testing a model transformation consists in running the transformation with test data, and checking that the model that is produced is acceptable with respect to the transformation specifications (expressed in natural language, in terms of rules or in terms of contracts). In the DOMINO project, we focused on the automatic generation of test models, taking into account two essential issues in test data generation: these data must satisfy a large number of constraints coming from heterogeneous sources of knowledge – cover requirements, conform to the meta-model, satisfy preconditions for the transformation – and these data are structurally complex-graphs of objects [23]. The solution proposed in DOMINO is based on the expression of all constraints in a common formalism and on techniques for automatic constraint solving using SAT solvers [23] [24].

3.2.5 Traceability of the transformations

The ETraceTool platform of DOMINO captures traces of imperative model transformations [3] [4] [5]. Our objective is to track transformational events (update, delete and create) during imperative transformations and organize them in a trace model. The transformation events are intercepted in a non intrusive way by means of aspects-oriented techniques and represented as a trace model which conforms to nested and arcs labeled traces meta-model. Then we have a nested traces graph that is isomorphic to the method calls. In a MDE way, if we tool all along a refinement chain with our traceability tool and repercussion transformation, we can obtain requirement traceability in exhibiting design choices on properties refined during the process. A model is generated for each execution of a traced transformation. The platform has been used in the refinement of properties defined in Context Description Language (CDL). A new transformation obtained by co-evolution thus allowed the automatic generation of the refined properties from the transformation of an abstract context and its CDL properties [4].

3.2.6 Graph rewriting

It is a unified approach for the category of attributed graphs depending on inductive types to define the structure of graphs and the associated attributes. In our framework which uses the double pushout approach to rewrite graphs, the arrows for attribute parts are reversed with respect to the arrows of the structural parts. This reversal permits us to have a pseudo-pullback (pushout in dual category) to organize the computations with attributes [22]. In order to tend towards a program for operational model transformation, we consider two paths: proof-oriented implementation (Coq and other proof assistants) or implementation in a functional language (Caml, Ocaml and Haskell) [26].

3.2.7 Incremental refinement

We are working on definition of an incremental approach for the construction of complete, valid and deterministic behavioral specifications based on transition systems. The model would be constructed from a succession of models obtained through a series of transformations (addition or deletion of elements, reduction of the indeterminism, etc.) with associated verification procedures. The computability of the conformance relation on transition systems relies on the composition of two operators: the reduction relation (*red*) and the merge function of acceptance graphs (*Merge*) associated with transition systems under comparison. We demonstrated by a theorem the conformance relation computability (*conf*) through a reduction relation applied on the merging of acceptance graphs [14]. This result was illustrated through a case study whose analysis is performed by a Java prototype we have developed. We plan to study the applicability of this approach to UML behavioral models, in particular to state machines and sequence diagrams.

4. RESULTS

One of the DOMINO contributions for the industrial partners CNES and Airbus is the improvement of an MDE process integrating the activities of model transformation and validation.

DOMINO provides technologies that aim at building trust in automatic transformations. This should reduce human efforts and the amount of error-prone, tedious manual activities of the development process. Moreover, they should reduce the amount of test and manual checks on MDE components while maintaining the necessary level of trust.

Expected MDE benefits are twofold: MDE provides the ability to build more automated design processes and know-how capitalization; it also provides an opportunity to capture well-formed (IEEE 1220: unambiguous, testable or measurable, etc.) and formal requirements. When requirements are modeled as formal properties, the subsequent design and validation phases are more effective, and trust may be provided by DOMINO formal technologies. Another MDE benefit is the ability to use, early in the requirement engineering process, high level concepts, easier to use for humans compared to those coming from lower level executable domain specific languages [20]. DOMINO's DSL technologies are elements that may help to achieve this goal.

We validated some techniques for building trust in MDE components. The process iterates on the activities related to the three vertices. At each step, one can choose the technology that is the best suited to manage the current activity for the development of the component. Improving a summit with respect to the two others improves the overall consistency of a component. Thus, for instance, in the context of V&V by testing, fixing an error in the implementation leads to the creation of a new set of mutants. This forces to iterate on the generation of test models and the definition of contracts. Likewise, the improvement of the contracts involves testing the implementation again, since the contracts can be used as the oracle of any test case.

The case study proposed by Airbus suggested the MDE approach to develop control functions implemented in tools used for the development of the fly-by-wire software. The specification of these controls was described in UML / OCL. OCL invariants, pre- and post-conditions were applied to UML objects associated to ICD files describing the input-output signals of the embedded software. This experimentation allowed to define unambiguous

requirements and to implement them directly on the USE framework. On top of these results which give trust in the capabilities of this technology to save costs and delays in industrial context, we found that UML / OCL could be very interesting for define users' needs, as a common language between embedded software team and tool development team. In this way, we experimented and checked it with the users and we saw that UML / OCL could allow them to define their needs and to check them very soon in the development cycle. Following this project, UML / OCL is now used, within an operational project, for the development of a verification tool dedicated to the control of signals for the bus 1553 used by a fly-by-wire software of the Airbus' forthcoming A350.

CNES participated through a case study on the reliable design of operational procedures and associated operations [21]. The first objective was to improve offline interoperability with the possibility to build import/export tools for any scripting procedure language using meta-modeling technology. The second was to improve efficiency for the production, validation and execution of scripting procedures using operational specifications. For instance, we successfully applied our ETraceTool platform on a transformation based on two main master models as input: an activity diagram which represents a procedure to apply to perform a specific task, and various technical statements used to express all the possible low-level satellite manipulation commands [4]. The target model is a grammarware model of a procedural language for satellite manipulation. We obtained traces from this transformation. Properties are expressed on interaction diagram elements and we aim to code them in the target language.

CNES didn't only experiment model transformations but also verification and validation of requirements. To be able to capitalize on expertise for formal checking, it seems important to structure the approach and the data handled during the proof. For that purpose, we identified MDE components, called proof units, referencing all the data, models, meta-models, etc. necessary to the verification. Definition of such MDE components can take part in a better methodological framework, for a better capitalization of software validation activities, and afterwards a better integration of validation techniques in model development processes. This verification technique has demonstrated its effectiveness trough aeronautic (with Airbus) and spatial (with CNES) case studies [9].

Proof units address behavioral requirements and can be seen as an advanced test unit which explores many execution paths. With regard to the DOMINO triangle representation, they can be used starting from the beginning of the specification phase with the high level requirements to design it an iterative way up to the lower level requirements. This experimentation has been continued internally at CNES using space domain knowledge to constrain the context expressed in CDL for the power subsystem of PICARD spacecraft. CNES could check then without state explosion that the battery can stay sufficiently charged despite some worst scenarios during eclipse period and including one failure of a sun array drive mechanism.

Tables 1 and 2 report the technical achievements for the CNES and Airbus case studies about the points mentioned above. The items in italics refer to academic studies related to case studies. The first table relates to the validation of models and the second one is related trustworthy components.

Table 1. DOMINO's achievements for the validation of models

▪ *A meta-model generator based on BNF rules applied to the meta-modeling of O2PL language*
▪ Proof of a behavioral property about the *Guidance Navigation and Control (GNC)* system
▪ *Definition of a language common to two dialects inspired from the Pluto standard*
▪ A meta-model generator based on BNF rules applied to the meta-modeling of C language
▪ Proof of the properties of the model of "COM/MON synchronization"
▪ Management of the "COM/MON synchronization" requirements
▪ OCL verification of the models at the input of the code generators

Table 2. DOMINO's achievements for trustworthy components

▪ Proof in B of the transformation of SOLM into O2PL
▪ pOCL verification of the transformation Activity Diagram into O2PL
▪ Automatic generator of test data
▪ *Co-evolution of business and properties models*

From the academic point of view, the study provided a better understanding and formalization of the notion of trust. The different propositions have been consolidated by the definition of a common framework of interaction between specification, implementation and verification of the component. As a multi-faceted concept, trust involves different formalisms which are complementary: requirements and traceability of the transformations, proof units and traceability, modeling formalism and the creation of a unifying language for a family of DSL.

5. DISCUSSION AND PERSPECTIVES

5.1 Diagnosis and analysis

The techniques developed by DOMINO are meant to design correctly and find defect in models or MDE components. However, if these techniques allow detecting anomalies early, they currently provide little assistance to find the source of the error and to fix it. Providing relevant and understandable feedback to the user, based on the analysis provided by DOMINO techniques, is an important perspective in order to make these techniques applicable in an industrial context. For example, while studying the applicability of OCL or proof units, the lack of efficient feedback has been identified has a major limitation by Airbus. Diagnosis information has to be provided as charts, text or logs, which are technological spaces very different from the modeling space. This shift between spaces is a major challenge for relevant diagnosis and efficient assistance to diagnosis.

5.2 Adaptable components

The studies carried out during the DOMINO project also demonstrate the necessity to identify and quantify the trust that can be attributed to a MDE component, especially a model transformation that automates parts of software development. If this is put into practice, software engineers would therefore have access to libraries of MDE components and would choose the component(s) that best fit the process activities. On the other hand, it is necessary to take specific contexts into account when using a component as a part of a specific MDE process. This

raises the challenge of the variability of components and the way they are used. This perspective is also about how to compose components in compliance with the global reference process. The use of such components can only be envisaged if the investment devoted to the requirement formalization actually leads to more trust and less tedious and error-prone development tasks. This will leave time for the unambiguous management of activities and interference-free interpretation and will lead to an efficient and robust model-based development.

5.3 Multi-domain collaborative development

To ease the effective use of DOMINO technologies in an industrial context, it is necessary to take into account collaborative engineering with experts coming from different domains. Embedded systems such as space systems are often systems of systems that would benefit from multi-views capability for design, validation and also operation phases. However, multi-domain collaborative development of complex system models is not currently supported by model-based environments. Working on different but related models can be critical when considering dynamicity of models and domain-specific teams. In order to address complex inter-relationships and complex evolution cycles of the whole process, we need to address consistency checks between viewpoints at some specific synchronization periods of the development.

6. ACKNOWLEDGMENTS

The DOMINO project was supported by the French National Research Agency (ANR) from March 2007 to June 2009.

7. REFERENCES

[1] A. Abou Dib, I. Ober, L. Féraud, C. Percebois. Towards a Rigorous Framework for dealing with Domain Specific Language Families. International Conference on Sciences of Electronics, Technologies of Information and Telecommunications (SETIT 2008), Damascus, Syria, April 2008. *First Best Paper Award.*

[2] A. Abou Dib, I. Ober, L. Féraud, C. Percebois. Towards interoperability in component based development with a family of DSLs. European Conference on Software Architecture (ECSA 2008), Chypre, September 2008.

[3] B. Amar, H. Leblanc, B. Coulette. A Traceability Engine Dedicated to Model Transformation for Software Engineering. ECMDA Traceability Workshop, Berlin, June 2008.

[4] B. Amar, H. Leblanc, Ph. Dhaussy, B. Coulette. Trace Transformation Reuse to Guide Co-evolution of Models. 5th International Conference on Software and Data Technologies, Athens, Greece, July 2010.

[5] B. Amar, H. Leblanc, B. Coulette, C. Nebut. Using Aspect-Oriented Programming to Trace Imperative Transformations. 14th International IEEE EDOC Conference (EDOC 2010), IEEE Computer Society Press, 2010 (to appear).

[6] J. Bézivin, J. On the Unification Power of Models. Software and System Modeling. SoSym 4(2):17-188, 2005.

[7] L. J. Camp. Design for Trust, Trust, Reputation and Security: Theories and Practice. Ed. Rino Falcone, Springer-Verlang (Berlin) 2003.

[8] Ph. Dhaussy, J. Auvray, S. De Belloy, F. Boniol, E. Landel. Using context descriptions and property definition patterns for software formal verification. Workshop MoDeVVa'08, 9 April 2008 (hosted by ICST 2008), Lillehammer, Norway.

[9] Ph. Dhaussy, P.-Y. Pillain, S. Creff, A. Raji, Y. Le Traon, B. Baudry. Evaluating Context Descriptions and Property Definition Patterns for Software Formal Validation. 12th IEEE/ACM International Conference on Model Driven Engineering Languages and Systems (MoDELS'2009), 2009.

[10] H. Dubois, M.-A. Peraldi-Frati, F. Lakhal. A Model for Requirements Traceability in a Heterogeneous Model-Based Design Process: Application to Automotive Embedded Systems. 15th International Conference on Engineering of Complex Computer Systems, pp. 233-244, St. Anne's College, University of Oxford, 22-26 March 2010, UK.

[11] H. Le Dang, H. Dubois. Proving Model Transformations. In Proceedings of the 4th IEEE International Symposium on Theoretical Aspects of Software Engineering (TASE 2010), August 2010, Taipei, Taïwan.

[12] T.-T. Le Thi, P. Bazex, T. Millan. Modeling of Languages' Grammars in UML/OCL: Applying to a Model Driven Software Development Process. International Conference on Theories and Applications of Computer Science (ICTACS 2009), Nha Trang, Vietnam, February 2009.

[13] T.-T. Le Thi. Modeling of Programming Languages in UML/OCL and Application in a MDE Process. IADIS International Conference Applied Computing 2009, Rome, Italy, November 2009.

[14] H.-V. Luong, T. Lambolais, A.-L. Courbis. Implementation of the Conformance Relation for Incremental Development of Behavioural Models. 11th International Conference on Model Driven Engineering Languages and Systems (MoDELS'2008), 28 September - 3 October 2008.

[15] T. Millan, L. Sabatier, T.-T. Le Thi, P. Bazex, C. Percebois. An OCL extension for checking and transforming UML Models. International Conference on Software Engineering, Parallel and Distributed Systems (SEPADS'09), Cambridge, United Kingdom, February 2009.

[16] J.-M. Mottu, B. Baudry, Y. Le Traon. Mutation Analysis Testing for Model Transformations, ECMDA'06 (European Conference on Model Driven Architecture), Bilbao, Spain, 2006.

[17] J.-M. Mottu, B. Baudry, Y. Le Traon. Reusable MDA Components: A Testing-for-Trust Approach. 9th International Conference on Model Driven Engineering Languages and Systems (MoDELS'2006), October 2006.

[18] J.-M. Mottu, B. Baudry, Y. Le Traon. Model transformation testing: oracle issue. Proceedings of MoDeVVa workshop in association with ICST'08, 2008, Lillehamer, Norway.

[19] I. Ober, A. Abou Dib. Using ASM to achieve executability within a family of DSL. International Conference on ASM, B and Z, London, September 2008.

[20] I. Ober, L. Féraud, C. Percebois. Dealing with variability within a family of domain-specific languages: comparative analysis of different techniques. Innovations in Systems and Software Engineering, Springer, Vol. 6 Number 1, p. 21-28, January 2010.

[21] E. Poupart, G. Jolly, C. Percebois, P. Bazex, P. Palanque, S. Basnyat, P. Rabault, L. Sabatier, A. Walrawens. Offline interoperability, cost reduction and reliability for operational procedure using meta-modeling technology. International Conference on Space Operations (SpaceOps 2008), 2008.

[22] M. Rebout, L. Féraud, S. Soloviev. A Unified Categorical Approach for Attributed Graph Rewriting. 3rd International Computer Science Symposium in Russia 2008, Moscow, June 9-14th, 2008.

[23] S. Sen, B. Baudry, J.-M. Mottu. On Combining Multi-formalism Knowledge to Select Models for Model Transformation Testing. Proceedings of ICST'08 (International Conference on Software Testing Verification and Validation), 2008, Lillehamer, Norway.

[24] S. Sen, B. Baudry, J.-M. Mottu. Automatic Model Generation Strategies for Model Transformation Testing. International Conference on Model Transformation 2009 (ICMT'09).

[25] The Papyrus tool: http://www.eclipse.org/modeling/mdt/

[26] H.N. Tran, C. Percebois, A. Abou Dib, L. Féraud, S. Soloviev. Attribute Computations in the DPoPb Graph Transformation Engine. 4th International Workshop on Graph Based Tools (GraBaTs 2010), University of Twente, Enschede, The Netherlands, September 2010.