A Design Pattern meta model for Systems Engineering

François Pfister, Vincent Chapurlat, Marianne Huchard, Clémentine Nebut

A design pattern meta model for Systems Engineering

François Pfister*, Vincent Chapurlat*
Marianne Huchard and Clémentine Nebut****

**LGI2P, Ecole des Mines d'Alès, site de Nîmes, Parc Scientifique Georges Besse, 30000 Nîmes, France*
(e-mail:{forename.lastname}@mines-ales.fr)
***LIRMM, CNRS – Université Montpellier 2, 161 rue Ada, 34095 Montpellier Cedex 5, France*
(e-mail: {lastname}@lirmm.fr)

Abstract: STEP (ISO 10303) and PLib (ISO 13584) standards define meta-models and data exchange formats for industrial component models. Many digital component catalogs used by the CAD/CAM tools are based on these standards. In the same way SysML (System Modeling Language) from OMG is a specification language for designing, analyzing and verifying complex technical systems in Systems Engineering domain. Models defined with SysML can be transformed to conform to STEP and PLib standards. Within models, particular component arrangements can be frequently recognized, corresponding to specific features or functionalities. They are now known as reusable patterns or design patterns. These one provide a potential mean to enrich the model semantics, and, as such, to enhance generation of automatic model matching. Therefore they allow engineers to study and improve interoperability of systems under design. This paper presents a first approach of a model alignment method based on design patterns, which can be considered here as an extension to existing modeling tools.

Keywords: design-pattern, systems engineering, model matching, model alignment, model driven engineering

1. INTRODUCTION

Systems Engineering (SI) proposes, in an approach called Model Based Systems Engineering (MBSE) (Estefan, 2008), to adopt the principles of MDE (Model Driven Engineering). This bringd the experience of a decade effort and the promise of domain models sustainability compared to the variability of actual tools and technologies. In addition, Systems Engineering, like software engineering, uses several modelling languages then meta-metamodels: the succession of these conceptual foundations was an obstacle to business models re-usability. Allow the conservation of the latter despite technological evolutions and found them above a stable conceptual pyramid like OMG's, this is the intention of MBSE. Systems may be composed of sub-systems which often are heterogeneous, themselves composed of software, hardware and components, interfaced with human actors or other systems. Therefore, it is necessary to ensure, during the conception phase, interoperability between each connected sub-system, in spite of the final system native heterogeneity or complexity. In accordance with (Pingaud 2009), "*interoperability can be defined as an aptitude for two foreign systems to interact in order to establish collective, harmonious, and finalized behaviours, without the need of modifying deeply their own structure and behaviour.*" Systems efficiency, in terms of their missions, depends on their ability to establish and maintain connections with surrounding systems. Several types of interoperability barriers are emerging (technical, informational, functional, and semantic) (ATHENA, 2003). Our target interests are here the semantic and functional levels. To resolve these issues, alignment (as the result of a model matching activity) of models representing these systems is a prerequisite. One key to interoperability is based on the mapping of concepts and processes for each system to interoperate. Matching different but compatible systems (by aligning their models) can be done manually by domain experts, but also in a semi-automatic way. Indeed, current models contain hundreds or thousands of entities, and efficiency constraints have led to the emergence of tools to assist alignment. In this paper, we will try to show how design patterns can help model alignment. We propose to make a model pre-handling, in order to find pattern correspondences for each model, as described further. Such a pre-handling consists of a decomposition which will be useful to facilitate model alignment. We also will consider all the consequences of using design patterns in Systems Engineering.

We describe the pattern semantics in section 2; in section 3 we propose a pattern metamodel for Systems Engineering. In section 4, we expose current pattern implementation within metamodels. In section 5; we tell how to mobilize patterns to improve system interoperability and model alignment, and we'll conclude in section 6.

2. DESIGN PATTERN SEMANTICS

Design patterns are one of the approaches used to abstract individual constructions within an overall architecture. This idea, originally proposed by Christopher Alexander (Alexander et al., 1977), has been widely used in software

engineering (Gamma et al., 1994). We propose to use of Design Patterns in Systems Engineering, not in response to technical implementation problems as does Gamma, but to structure the functional and physical architecture. Pattern mobilization occurs, in Systems Engineering, during the requirements definition phase, while describing the problem solved by the pattern application, as well as in the functional and the physical architecture design phase, while describing the solution carried by the pattern. A design pattern is a way to represent invariant knowledge and experience in design by practitioners. It can help human actors to identify and solve problems by drawing or imitating such knowledge and experience. The objectives are: to gain performance (comprehensiveness, relevance), reliability (proven solutions, justified and contextual argued), to gain economic value (time savings) and, finally, to facilitate collaborative work by sharing pattern repositories. These objectives can be achieved by leveraging and integrating such knowledge, good practices and lessons learned, and by formalizing them for reuse. So design patterns are a means to formalize and create standard solutions repositories in response to known and frequently encountered issues in a particular field. A pattern is a simple and small artifact, rarely isolated and therefore correlated with other ones. A pattern is described by an abstract model to be imitated by actual models. It defines the collaboration of some system components or some system functions to contribute to a given mission. A pattern is not a creativity method (by definition, it exists only if the solution it proposes is well known and frequently used in the field and, therefore, is not innovative). In the same manner, it is not a reusable component. It is destined to be imitated and adapted to a particular context.

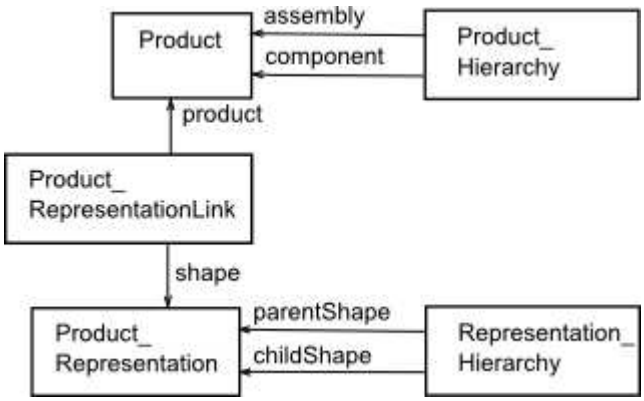

Fig. 1. Loose Coupling Assembly Pattern

We will distinguish several types of patterns: the first are idiomatic patterns, which describe a low level, structuring elements of a model, governing associations between these elements, defining aggregation and containment strategies. They are structures at low level without affecting model overall architecture. The GOF patterns (Gamma et al., 1994) could be classified as such, in the software engineering domain. Similarly, in Systems Engineering, physical components are sometimes expressed with ISO10303 AP214 or AP210 STEP models (physical components in the fields of mechanics and electronics). If one examines the STEP product representation (Fig. 1), we find that the composition pattern is far from the Composite Pattern described by (Gamma et al., 1994). Instead, we can detect a loose coupling pattern based on a bipartite graph. One graph represents the structure of the product, and the other represents it's geometry. In such a model, it will be possible to stack, as layers, many sub-models involved in various aspects of the component representation (functional model, performance model, geometric model ...) Patterns at this level, participate in the concern of modelling technology, and have no effect on the nature of the end systems. It is necessary to detect those patterns in the physical architecture that our algorithms must walk through, but these patterns are not involved in the macroscopic system semantics we are targeting. Patterns that we seek capture the various domain concerns about the system when in the operational phase of its lifecycle. So, the example that we propose in Section 5 describes a case of applying a regulation pattern within the functional and physical architecture of motor vehicles.

3. A PATTERN METAMODEL FOR SYSTEMS ENGINEERING

This point constitutes the first part of our contribution. Many proposals for formalizing patterns exist. These include for example the P-Sigma formalism (Conte et al., 2002), and also (Gzara 2000) contribution who proposes the application of design patterns for product data management, that is, to structure a model of physical system components. Many efforts have been undertaken by the promoters of MBSE to integrate design patterns into the models developed by Systems Engineering (Cloutier, 2007). The AFIS (Association Française d'Ingénierie Système - French Association of Systems Engineering), the French chapter of INCOSE, has been mandated to further the implementation of design patterns in System Design. Our participation in the Technical Committee MBSE within AFIS gives us the opportunity to begin a formal reflection to represent design patterns on the basis of SysML or its implementation by the ISO 10303 - STEP AP233.

We rely on the Cloutier proposal (Cloutier, 2007) but we consider more formally the pattern metamodel within a (candidate) System metamodel (Fig. 2). The design pattern for Systems Engineering (SystemPattern) has participants who can be physical components, or functions in the case of functional patterns. The fragment of an actual model impacted by a pattern imitates the latter if the actual model's components or functions play the role of the pattern participants, and also if the model's dynamic behavior mimics the dynamic behavior described by the pattern.

Fig. 3 shows a pattern associated with other ones (related, requested or mandatory, equivalent, anti-pattern). It can have multiple aliases (aka, also-known-as), a set of keywords allows its localization, and has a rationale. A pattern is legitimated by citing known application cases (known-uses). Its participants are components or functions, one components being associated with one another through interfaces that convey informational or physical flows (Item on Fig. 2).

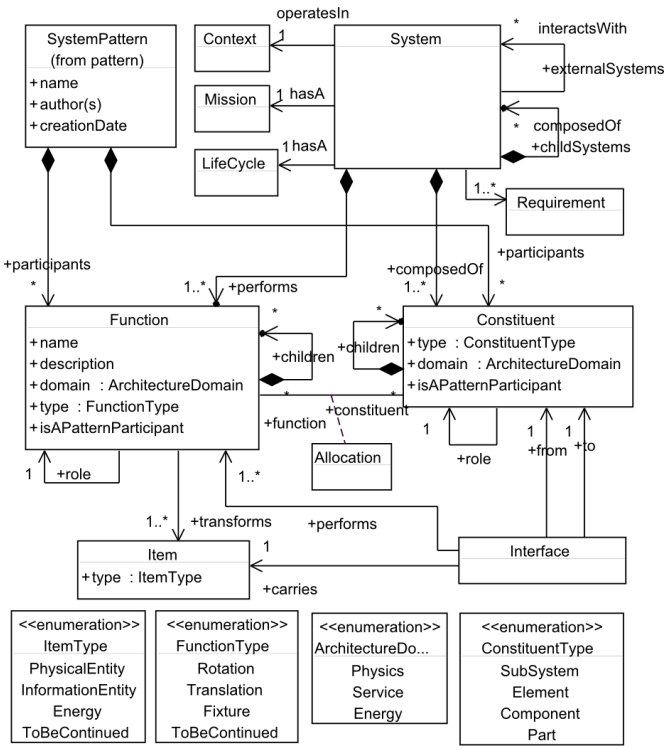

Fig. 2. Systems Engineering patterns within a System metamodel (overall vision).

According to Fig. 4, a pattern embeds a controlled vocabulary depending on the domain where it belongs. The application context of the pattern is described, so are the set of constraints and contradictions that the design pattern solves (forces). The impact of the pattern implementation is evaluated, and finally, the definition is supplemented by a model of the problem solved by the pattern, and a model of the solution to be imitated by the actual model. These models can be described with different languages, each language offering different views (static, dynamic, functional, behavioral).

4. PATTERN IMPLEMENTATION WITHIN METAMODELS

According to the Model-Based/Model-Driven initiative (Soley 2000), a pattern should be represented using a formal language, handled through a design tool and be implied in model transformations. Patterns (and models on which they apply), are expressed with languages such as SysML (OMG), which is an UML profile, or STEP (ISO), based on the EXPRESS schema (Schenk, 1994), and also OWL (W3C). Within models, design patterns apply a crystallization process (Baudry, 2003) (Jézéquel et al., 2005). Impacted entities fit together in a configuration to meet specific roles defined in the pattern. Design patterns are defined in UML or SysML as parameterized collaborations. They specify a set of classes and objects that have specific roles and interactions. Mechanisms such as inheritance, delegation, and implementation are used to give rise to collaborations that will be captured by use cases as well as through interaction diagrams. (Jacobson, 1997) (Sunyé 2000). Applying a pattern returns to generate or to correct a part of the model by applying a prototype (Barcia 2006).

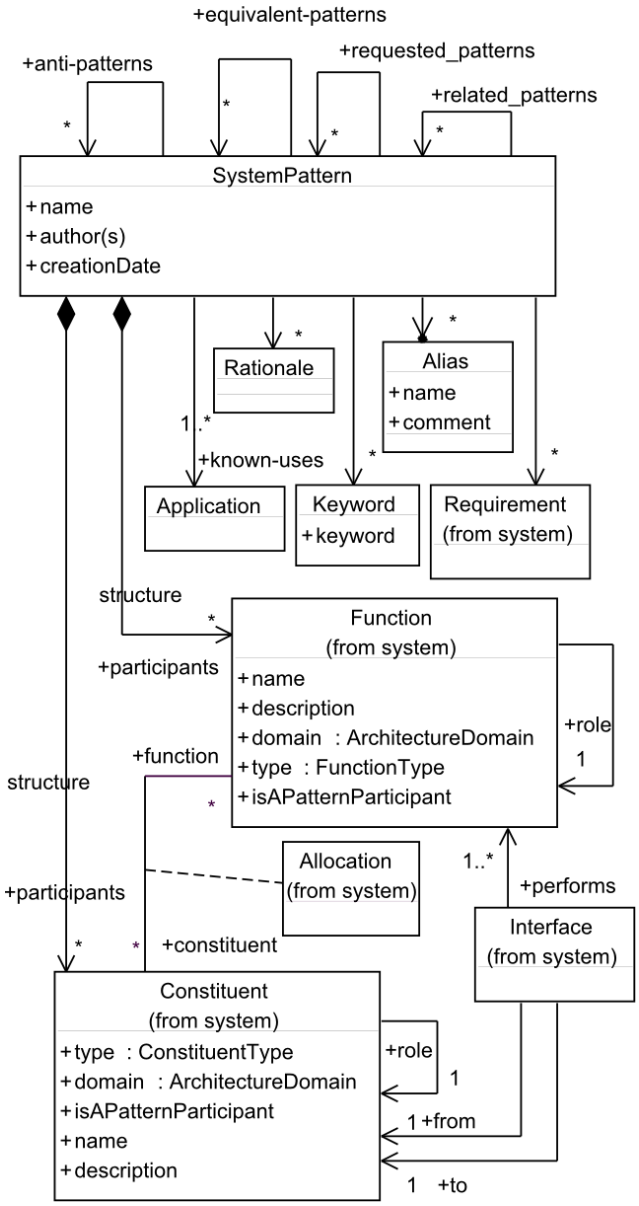

Fig. 3. System Pattern meta-model (A).

5. PATTERNS MAY IMPROVE MODEL ALIGNMENT

This constitutes the second point of our contribution. Model transformation, according to the MDA paradigm, is based on a transformation model instance of a meta-model as QVT (Bézivin, 2001). This transformation model formalizes a mapping of correspondences between the source model elements and the target ones. When match rules are not trivial, the transformation model formalizes more complex imperative rules. The transformation model can be constructed manually by an expert in the domain in which lie the two models to align. Actually, these models may contain hundreds or thousands of items, so it would be useful to rely on a tool to assist the expert. Our proposal is to use the contribution of design patterns and related tools already

integrated modeling platforms to improve the automated discovery of mapping alignment, in combination with existing techniques (Falleri et al. 2010) (Euzenat et al., 2004).

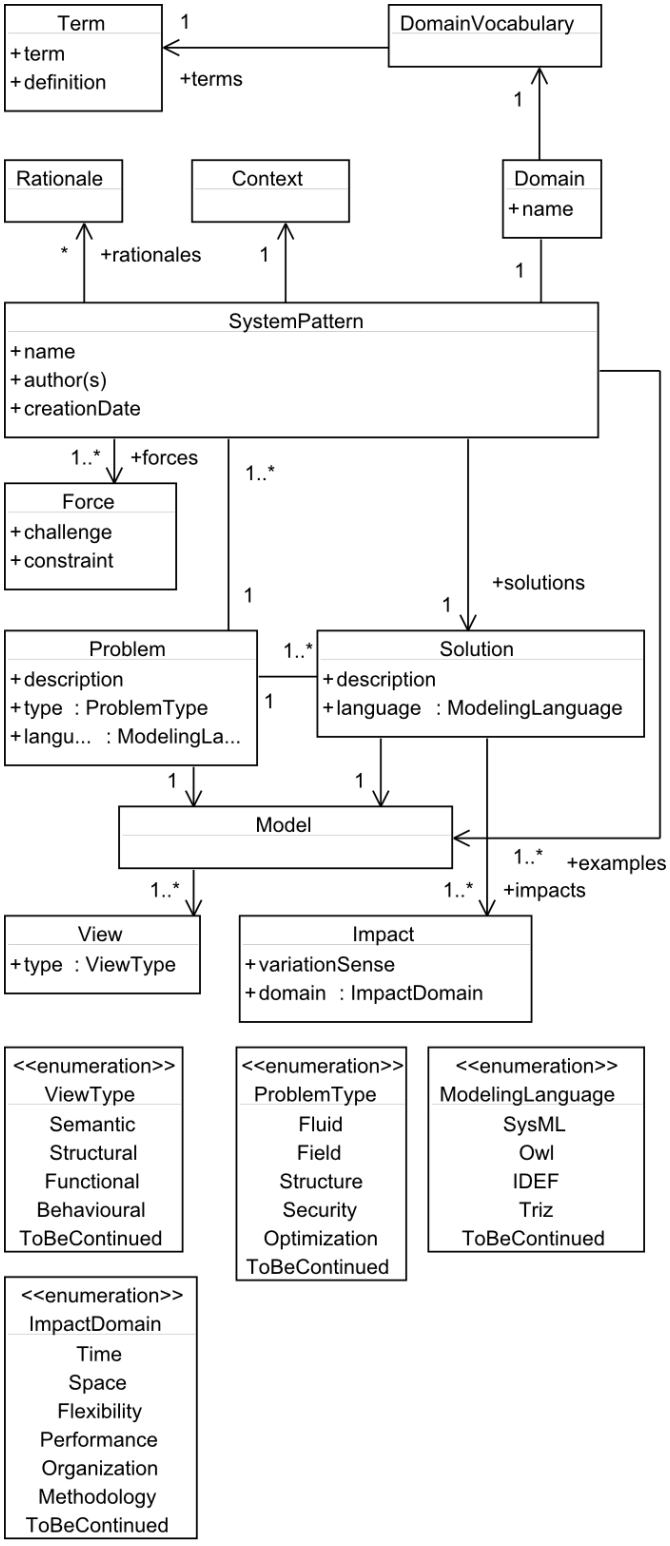

Fig. 4. System Pattern meta-model (B).

The currentl model components are reconfigured to mimic the pattern. Notably, in the case of physical or mechanical oriented systems, particular constraints (structure, geometry, surface aspect, non-functional requirements) can be allocated on the current components. These fit together so that they assume the roles of the parameters defined for collaborations that represent the pattern in the model. These roles represent associations between current model components and the pattern participants. If the pattern is involved in the functional architecture, the elements are impacted functions. In addition to their structural impact, the pattern also specifies dynamic constraints in prescribing sequential collaboration protocols between components or behaviors described with state diagrams. In the case where patterns are not expressed in the model by the existence of such roles, some tools can detect when buried (Tonella et al., 1999) (Arevalo & al. 2004) (Gueheneuc, 2008).

The example shown Fig. 5 describes a cruise-control pattern, applied to an electric car, on the one hand, and on a conventional thermic car on the other. Each model mimics this pattern to define its own cruise-control subsystem. Thus, this case represents the need to interoperate two automotive systems helped by aligning their models. The pattern is described by its organizational structure (functional and dynamic views have been omitted). The pattern participants (throttle, brake, transmission control unit ...) which are the component components of a solution-type cruise control, are associated with the roles played by the component components of the two actual models that imitate the pattern. A classical alignment model based on the Similarity Flooding (Melnik et al., 2002), which is a method of propagation of similarities in a labeled graph to determine a match, will be used to generate a mapping between two models. This algorithm will be effectively complemented by the analyzing the roles connecting the pattern to both models.

6. CONCLUSIONS

We proposed to complete the process initiated by INCOSE, to formalize a Pattern for Systems Engineering metamodel. This metamodel takes advantage of lessons learned in software engineering, but adapts the principle to System design.

If a model keeps trace of patterns it imitates, the latter will contribute to document it and to promote interoperability of the represented system, with surrounding systems. The system components are often physical components, modeled with languages used to represent products throughout their life cycle, such as STEP AP203 and AP214 (Mechanical Engineering) and AP210 (Electronic Engineering). In continuation of our work, we focus to implement design patterns within this family of languages, and study the consequences in terms of interoperability in these specific areas. Finally, from a methodological point of view, we will propose to help designers to describe domain patterns and to pass from problems to solutions by implementing the metamodel described above. Such a methodology will lead us to contribute to existing Systems Engineering methods, with the proposition of a pattern application functional model, expressed in SysML activity diagrams.

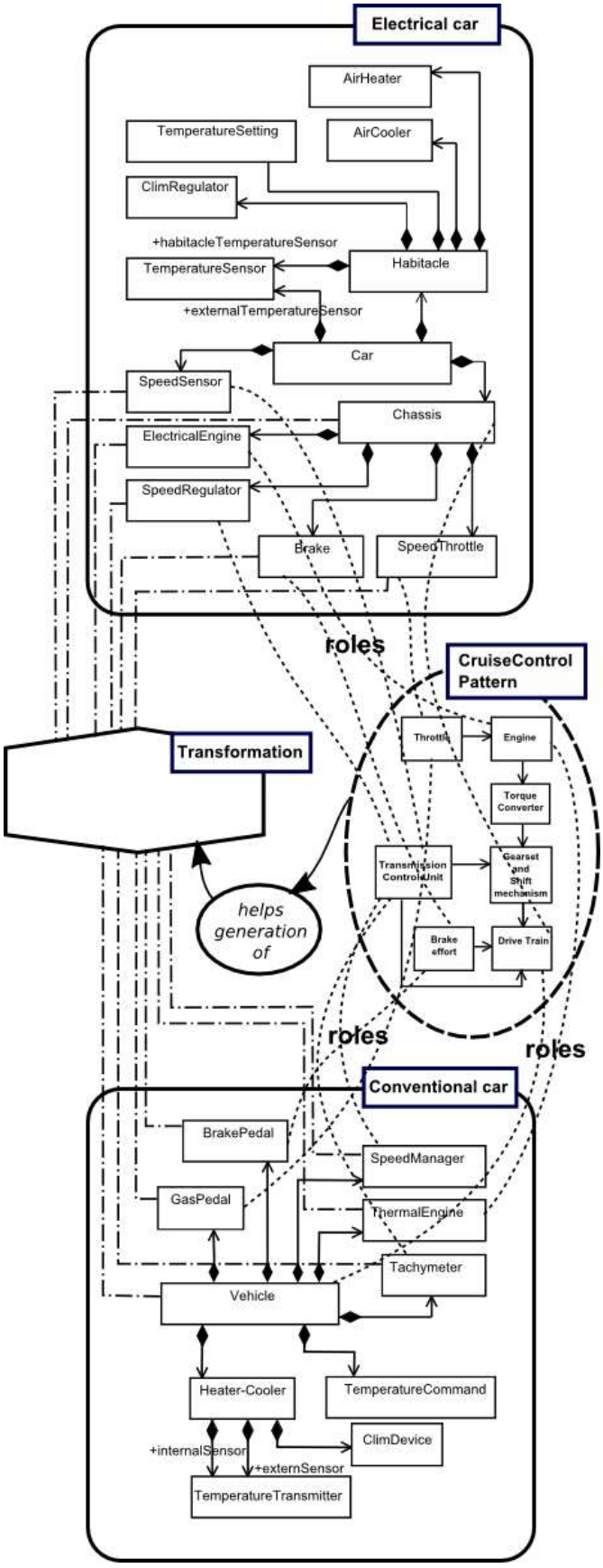

Fig. 5. Role based model alignment.

REFERENCES

Alexander, C. & al., 1977. A pattern language : towns, buildings, construction, Oxford University Press.

Arevalo, G., Buchli, F. & Nierstrasz, O., 2004. Detecting implicit collaboration patterns. In 11th Working Conference on Reverse Engineering. Delft, Netherlands.

ATHENA, 2003. Advanced Technologies for Interoperability of Heterogeneous Enterprise Networks and their Applications,

Barcia, R. & Gerken, C., 2006. Get started with model-driven development using the Design Pattern Toolkit. IBM WebSphere Developer Technical Journal.

Baudry, B., 2003. Assemblage testable and validation de composants. Renens, France: Rennes 1.

Bézivin, J. & Gerbé, O., 2001. Towards a Precise Definition of the OMG/MDA Framework. In 16th IEEE international conference on Automated software engineering. San Diego, U.S.A.

Cloutier, R. & Verma, D., 2007. Applying the concepts of patterns to systems architecture. In Systems Engineering. Wiley, p. 138-154.

Conte, A. & al., 2002. A tool and a formalism to design and apply patterns. In International conference on object-oriented information systems. Conference on Object-Oriented Information Systems OOIS. Montpellier, France, p. 135-146.

Estefan, J., 2008. Survey of Model-Based Systems Engineering (MBSE) Methodologies, Seattle, WA - U.S.A.: INCOSE.

Euzenat, J. & Valtchev, P., 2004. Similarity-based ontology alignment in OWL-Lite. In European Conference on Artificial Intelligence. Valencia, Spain.

Falleri, J. & al., 2010. Metamodel Matching for Automatic Model Transformation Generation. Model Driven Engineering Languages and Systems, 5301, 326-340.

Fenves, S. & al., 2005. Product Information Exchange: Practices and Standards. Transactions of the ASME Journal of Computing and Information Science in Engineering.

Friendenthal, S., Moore, A. & Steiner, R., 2009. A Practical Guide to SysML: The Systems Modeling Language, Morgan Kaufmann.

Gamma, E. & al., 1994. Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley.

Gueheneuc, Y. & Antoniol, G., 2008. DeMIMA: A Multilayered Approach for Design Pattern Identification. IEEE Transactions on Software Engineering.

Gzara, L., Rieu, D. & Tollenaere, M., 2000. An Approach for Building Product Models by Reuse of Patterns. In 7th ISPE International Conference On Concurrent Engineering CE2000. Lyon, France.

Jacobson, I., Griss, M. & Jonsson, P., 1997. Software Reuse: Architecture, Process and Organization for Business Success, Addison-Wesley Professional.

Jézéquel, J., Plouzeau, N. & Le Traon, Y., 2005. Développement de logiciel à objets avec UML, Université de Rennes 1.

Kemmerer, S., 2009. Manufacturing Interoperability Program, a Synopsis, N.I.S.T.

Melnik, S., Garcia-Molina, H. & Rahm, E., 2002. Similarity flooding: A versatile graph matching algorithm. In 18th International Conference on Data Engineering. San Jose, CA, U.S.A.

Pingaud, H., 2009. Rationalité du développement de l'interopérabilité dans les organisations. In Management des Techniques Organisationnelles. Nîmes, France.

Schenk, D. & Wilson, R., 1994. Information Modeling the EXPRESS Way, Oxford University Press.

Soley, R., 2000. Model Driven Architecture, OMG

Sunyé, G., Le Guennec, A. & Jézéquel, J., 2000. Design Patterns Application in UML. In ECOOP. Sophia Antipolis, France: Springer.

Tonella, P. & Antoniol, G., 1999. Object Oriented Design Pattern Inference. In IEEE International Conference on Software Maintenance. Oxford, England, UK.