

Using Model Transformation and Architectural Frameworks to Support the Software Development Process: the FIDJI Approach¹

Nicolas Guelfi, Gilles Perrouin
Software Engineering Competence Center
Faculty of Science, Technology and Communication
University of Luxembourg, Campus Kirchberg
L-1359 Luxembourg-Kirchberg, Luxembourg
{nicolas.guelfi, gilles.perrouin}@uni.lu

Abstract

Framework-based development is a well-known software engineering practice which encourages reuse and thus reduces development costs. To improve maintainability and reuse, frameworks need to be modeled quite abstractly and provide specialization mechanisms to refine these models into concrete implementation. Hence they are candidates to address Model Driven Engineering (MDE) and Model Driven Architecture (MDA) challenges. Currently, few development methodologies for MDE integrate such frameworks in their processes. In this paper, we present a development approach called FIDJI, which bases its process on architectural framework specialization using model transformation and its associated CASE tool support.

Keywords: Architectural Framework, Model Transformation, Software Development Methodology

1 Introduction

Software architecture has occupied the front of the academic software engineering scene for more than 10 years (while the main concepts have been defined since the 60s). Real interest of industry for software architecture came only recently. Why this sudden interest? Software systems evolved from monolithic centralized architectures to distributed ones raising issues in the design and development of such systems. In [1] Rick Kazman explains why architecture is crucial to software-intensive systems: it promotes communication among stakeholders, captures early aspects of the design and finally by its abstraction level has the ability to be reused elsewhere.

The contribution of this paper shows how architectural frameworks and model transformation when integrated in a coherent, model-driven and architecture-centric design methodology help to pursue these goals. Section 2 states the problem, Section 3 examines the background FIDJI relies on, Section 4 and 5 describe the FIDJI methodology and its supporting model transformation approach, Section 6 presents FIDJI's dedicated tool support and Section 7 compares FIDJI with related approaches.

¹ This work is supported by the Luxembourg Ministry of Higher Education and Research under the title I project n°MEN/IST/01/001.

2 The problem

In one hand, frameworks are defined as “a set of classes that embodies an abstract design for solutions to a family of related problems, and supports reuse at a larger granularity than classes” [2]. On the other hand, a software *product line* is “a set of software-intensive systems sharing a common, managed set of features that satisfy the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way” [3].

It appears then clearly that frameworks are excellent candidates to implement product lines. However framework reuse is a hard problem. To address this issue, we have investigated coupling UML-based modeling and model transformation techniques with frameworks. We have called the resulting entity an *architectural framework*. It comprises a model-based description of the architectural design of the framework, the framework’s code, model refinement mechanisms (transformations) able to specialize the framework and a set of methodological guidelines driving the user through framework instantiation. Fitting in this context, we detail our approach in the next sections. We are also developing an e-business auctioneering product-line called LuxDeal. Its application to the mobile world can be found in [4].

3 Background

3.1 MDE/MDA

MDE advocates the shifting from code-oriented paradigms to model-oriented ones [5]. Adopting a model-based approach mainly deals with 1) model description 2) model exchange, and 3) model evolution throughout the lifecycle. In a standardization effort called Model Driven Architecture (MDA) [6], the Object Management Group, Inc. (OMG), is working on standard solutions to these issues. In effect, the MDA suite of standards includes 1) the Unified Modeling Language (UML) [7] that acts as a multi-purpose modeling language, 2) the XML Metadata Interchange (XMI) [8] format that promises an increased interoperability between CASE-tools, and 3) the MOF 2.0 Query / Views / Transformations standardization effort focuses on ways of managing model evolutions, both technically [9] and conceptually [6].

3.2 Model transformation

Model transformation is a central issue to MDE approaches [10]. There are two main approaches to specify model transformations: textual and visual. Textual approaches include programming languages, OCL extensions [11] and UML action semantics [12]. Visual languages rely on graph transformation theory: a transformation rule consists of a graph to match, commonly referred to as Left Hand Side (LHS), and a replacement graph, commonly referred to as Right Hand Side (RHS). If a match is found for the LHS graph, then the rule is fired, which results in the matched sub-graph of the graph under transformation being replaced by the RHS graph. A more complete overview of model transformation approaches can be found in [13].

4 The FIDJI approach

In this section we present the main steps of the FIDJI methodology.

4.1 Methodology

The FIDJI approach covers analysis to implementation disciplines² (with a focus on architecture to implementation ones) as shown on Figure 1.

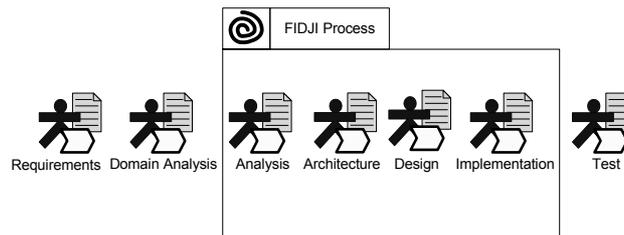


Figure 1 - FIDJI Coverage

The activities of the FIDJI process are organized around the specialization of the architectural framework via model transformations.

4.2 Analysis modeling

The role of the analysis model is to provide an ideally unambiguous definition of the system requirements. We use UML for creating this model. For the time being, this model comprises *use-cases*, organized in coherent sets. Each set models a system functional area. According to the applications needs, other UML modeling elements, such as class or sequence diagrams may be used during this phase. This will be addressed in future FIDJI extensions as well as non-functional requirements which could be defined as contracts [15].

4.3 Design modeling based on an architectural framework

Three intermediary activities, i.e. developing the user experience diagram, choosing the appropriate transformations and architectural framework, precede the creation of the design model and cover architecture and design disciplines.

User experience modeling [16] consists of exploiting the information provided in the analysis model to define the user interface content and its navigation structure. This activity is supported by specific model transformations that take as input the analysis model and generate a skeleton of the user experience diagram.

From this point, transformations that should be applied depend on the architectural framework and the analysis model (that may consider a family of similar software).

² In this paragraph we use the OMG's Software Process Engineering Metamodel (SPEM 1.0) for process descriptions [14].

The choice of an architectural framework is essential since it allows improving and accelerating the development process through architecture reuse. The concrete design step consists of then producing a UML model that instantiates the architectural framework profile. This instantiation takes into account the application requirements specified in the analysis model.

In the current state of our experimentations on this topic through the JAFAR architectural framework a design model comprises key abstractions classes and enterprise component interfaces. Further, since the JAFAR metamodel is being defined, the corresponding knowledge is encoded in the transformations rules.

4.4 Implementation

The design model produced in the last disciplines is platform-independent (PIM). The current step consists of choosing and applying the transformations that convert the design model first into a platform specific model (PSM), followed by its corresponding code. The ability to choose between several platforms depends upon the transformations provided by the framework and the code generation capability of the model transformation tool.

Even if transformations can produce most part of the implementation model, some additions have to be made “by hand”. The FIDJI process assists the developer by indicating the parts of the code that need to be completed.

4.5 Deployment

Deployment of the application relies on the platform deployment tool. However deployment model can be built and platform specific deployment artifacts can be constructed via transformations. Thus, model transformation plays a key role in the FIDJI process. This topic is further developed in the following section.

5 FIDJI’s model transformation

5.1 Overview

Implementing the FIDJI approach described above requires defining generic and complex transformations. To address this need, we have designed a new approach called Visual Model Transformation (VMT) that is presented hereafter. VMT is currently being developed at the conceptual level and will be supported in the next release of the FIDJI CASE tool.

VMT is a graph-based and mainly visual approach for describing UML model transformations. Within VMT, the specification of a transformation rule, or for short a *rule specification*, consists of two schemas, *matching schema* and *result schema*.

The matching schema of a rule specification declaratively describes the kind of model elements (and their properties) that can be matched by the transformation on a given source model. Thus, the matching schema plays the role of a LHS in graph grammars techniques. Model elements are defined

graphically using the UML notation. To each model element is associated a properties box which allows to set precisely properties and additional conditions (using OCL) under which they can be considered by the transformation.

The result schema of a rule specification defines the target model that should be constructed, based on the elements of the source model bound by the matching schema. Such a schema comprises also a set of properties boxes. In this case, each properties box corresponds to a set of properties or statements defining the precise attributes of each model element or set of model elements. Statements can be specified in Java or as OCL post-conditions.

Matching and result schemas form together a *transformation rule* diagram that completely specify one single rule. Within a transformation rules diagram, each model element is identified by its id in order to allow reusability and the tracing of elements between matching and result schemas.

Moreover, creating more complex model transformations by composing simpler ones is a good practice from the reuse and reliability perspective. VMT accommodates rule composition by means of “rule ordering”, i.e. defining the required ordering of simple rule applications. The UML activity diagrams are used to compose rule orderings. The resulting diagram is called a *rule-ordering diagram*. It allows building a sequence of rules as well as their branching and iterations. It further allows mapping outputs of a rule to inputs of another rule.

As opposed to Story Driven Modeling [17], in which the complete transformation resides within one type of activity diagrams, we think that having a clear separation of concerns between rules specification and their ordering enhances readability and understanding of the transformations.

VMT mixes visual, declarative and imperative languages in transformations rules, and organizes transformations with activity diagrams in a procedural fashion.

5.2 Example

As an example, we use the transformation “Entity Bean” which is proposed by the FIDJI CASE tool. This transformation consists in generating from an entity from the design model (shown on Figure 2) its corresponding EJB entity bean implementation according to the *value object* pattern [18]. This pattern proposes the creation of a value object encapsulating the business data. The data is delivered to the client once in a single method call which reduces network overhead in J2EE applications.

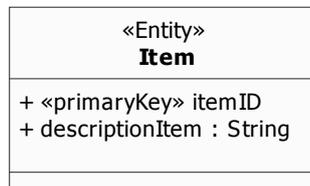


Figure 2. Item design entity

Matching and result schemas (separated by the dashed line) of the transformation are shown on Figure 3. The left hand side shows the matching schema of the “Entity Bean” transformation. Due to space considerations it is not possible to show all the boxes on the figures here. This schema allows for searching all classes having the “Entity” stereotype, that can have any names, owning at least two public attributes and that are part of the design model... The properties that can be defined are

depending upon the model element being considered with respect to the UML metamodel. One possible match for this schema is shown on Figure 2.

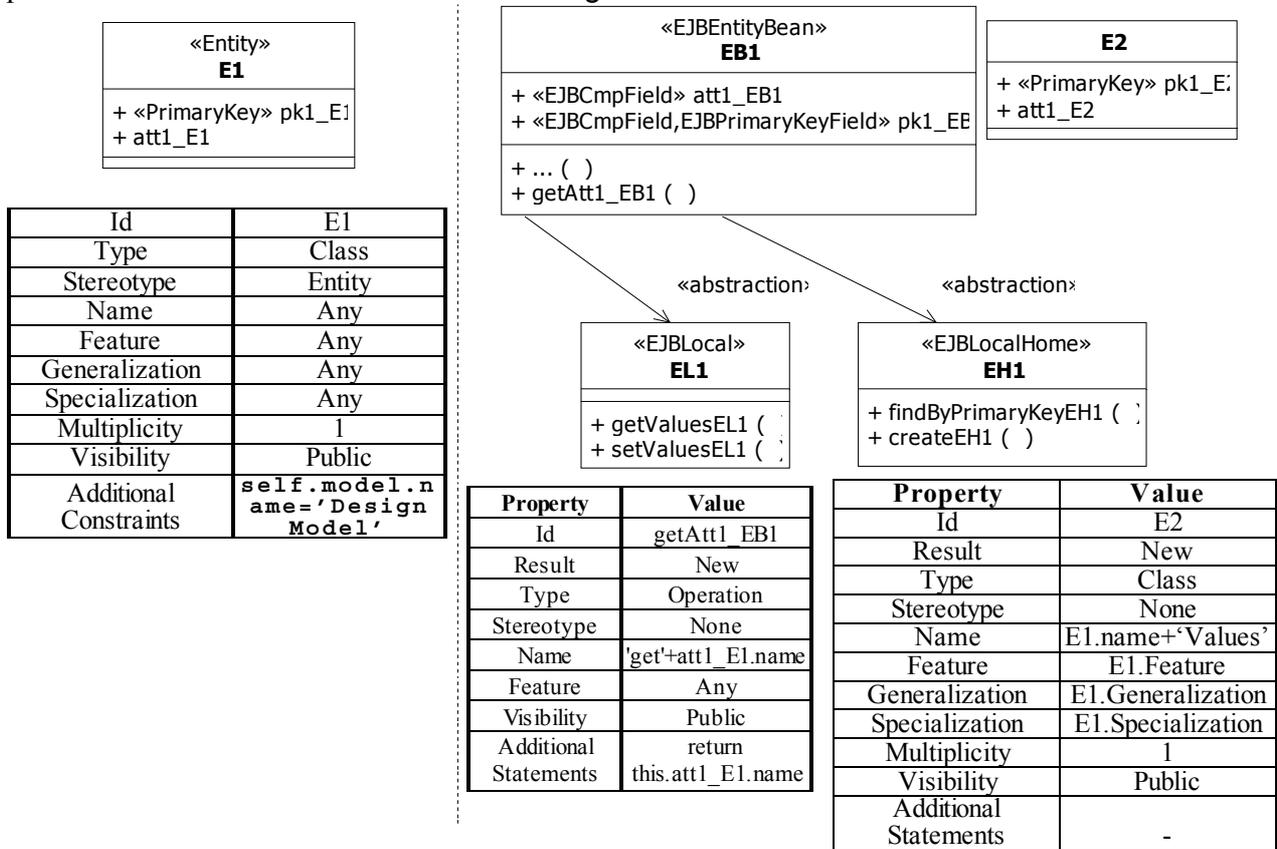


Figure 3. Matching & Result schemas of the “Entity Bean” transformation

The right hand side of Figure 3 shows the result schema of the “Entity Bean” transformation. This schema show how the transformation result is built: e.g. the id `getAtt1_EB1` refers to a “getter” which is new (not present in the source model) and whose name is formed upon `att1_EB1` name. It is also possible to provide its implementation in the additional statements field. `getValuesE1` and `setValuesEL1` are used to gain access to the value object E2. EB1, EL1, EH1 are built according to the entity bean transformation. Once again it is not possible to show all the boxes associated to each model element existing on the result schema.

6 FIDJI Case Tool

6.1 Overview

The FIDJI CASE tool is composed of the following parts: our architectural framework, JAFAR [19], our model transformer tool, MEDAL [20], and the UML CASE tool IBM/Rational XDE (JAVA version) [21] which is based on the Eclipse IDE .

6.2 JAFAR

JAFAR Platform, or for short JAFAR, stands for J2EE Architectural FrAmewoRk [19], It is an architectural framework for building J2EE [22] web applications. In effect, JAFAR extends the J2EE architecture with a set of design patterns [18] to provide an extensible architecture and ready-to-use recurrent services for building web applications.

JAFAR provides a component architecture which defines a number of component types that can be used to develop an application or to extend the framework. Each component type has a particular role in the overall architecture. The components are managed by a component registry. Applications that are based on JAFAR consist of two component types:

- Technical services: implement the application's business logic. Each service has an interface that describes the services it provides. The services are also responsible for managing the application's data. Each entity is managed by exactly one service.
- Dispatchers: The dispatchers implement the use case realizations' control logic. Client requests are forwarded to dispatchers to be handled.

The requests and services invocations are validated before being handled by the dispatchers and the technical services respectively. The request validation uses the framework's form validation mechanism with rules that the application developer has provided to validate the form data that the requests contain. The service invocation validator checks whether the current application user has the right to invoke that service. If a request or a service invocation is found to be invalid then an error is generated.

The dispatchers and technical services are grouped into units. Each unit realizes one functional area, consisting of a set of related use cases of the application. The framework itself is also built upon this component architecture, except for the framework core, which contains the basic services that support JAFAR's architecture. The resulting architecture is shown in Figure 4.

JAFAR's core provides the following functionalities:

- The component registry, which allows gaining access to other components
- The primary key generator, which allows generating unique primary keys in distributed applications in an efficient way
- A logging service

JAFAR is composed of the following units:

- Pluggable Presentation unit, which implements a facade that is able to handle requests coming from different kinds of user interfaces (e.g. web, swing)
- User Management unit: offers the user interfaces and the technical services that are required to manage users and groups. Permissions can be assigned to user groups.
- A messaging units offers a simple API for sending messages (e.g. e-mails) to application users
- Batch job unit that offers a means to define tasks that are executed asynchronously

JAFAR is compliant with J2EE 1.4. Its implementation uses the following features of the J2EE platform: the Enterprise Java Beans (EJB), Servlets, and Java Server Pages (JSP).

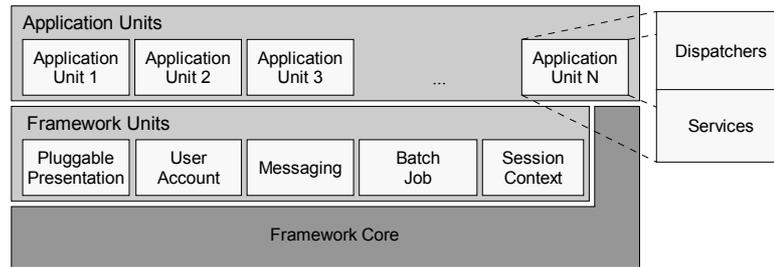


Figure 4. JAFAR application architecture

6.3 MEDAL

As stated in Section 4, our approach is to derive applications using architectural frameworks. MEDAL is used to support the application developer by providing transformations that assist the user in specializing the framework.

The FIDJI CASE tool uses our UML Generic Model Transformer tool (MEDAL) [20] to accelerate the exploitation of JAFAR platform by offering wizards and transformations that create prototype implementations of applications, starting from UML diagrams. It provides several transformations that take UML model elements as inputs and it refines them to other models either UML models or implementation models (code). These transformations guide the application designer while modeling his application. A complete technical description can be found in [13]. This first experiment with model transformation helped us to define the VMT approach presented in Section 5 that will be implemented in the future generation of the MEDAL tool (MEDAL being the generic name for our model transformation tool, we make follow the name of the transformation approach to avoid confusion: e.g. MEDAL-VMT).

7 Related work

In [23] Sztipanovits et al presents an approach called Model Integrated Computing (MIC). The approach is composed of the following steps:

- Design a Domain Specific Modeling Language (DSML): this step allows engineers to create a language that is tailored to the specific needs of application domain. One has also to create the tools that can interpret instances of this language (i.e. models of the application).
- This language is then used to construct domain models of the application.
- The transformation tool interprets domain models to build executable models for a specific target platform.

The whole approach is supported by a toolset; GME [24] (Generic Modeling Environment) for the design of DSMLs and their use (GME can turn itself to a DSML editor according to the specification of the DSML) and GRE [25] (Graph Rewriting Engine) which performs model transformations.

In FIDJI, modeling languages are based on the UML standard and its extension mechanisms. We also aim at providing reference architectures that can be instantiated via model transformations. Concerning model transformations, VMT supported by MEDAL and GRE share a similar approach as both are being based on graph transformation theory.

8 Conclusion

In this paper we have presented our approach to the engineering of software using architectural frameworks and model transformation. We have shown that, given an architectural framework, the application developer can, using the FIDJI approach, design the architecture of the application and obtain its implementation, these activities being supported using the VMT approach we have designed. The FIDJI approach is supported by a CASE tool based on the IBM/Rational XDE development environment.

The main benefit of our approach is that it forces developers on how they reach designs from the requirements while transformations assist them in refining their designs and obtaining most part of the implementation. Developers' work can be seen as writing transformations instead of writing application specific code. However the FIDJI approach needs a relatively stable domain to obtain an architectural framework to maximize the benefits.

Further work on the FIDJI approach includes the description of JAFAR's metamodel in the form of an UML profile (using the upcoming UML 2.0 specification) to improve the flexibility and automation level of the approach. We plan to assess performance of JAFAR and integrate our Visual Model Transformation approach in the FIDJI CASE tool.

The FIDJI approach is currently being transferred in the field of embedded systems development and in the field of secure and complex nested transactional systems. In those two domains, we found interesting architectural frameworks to develop and we have also identified a precise set of transformations. Thus it will also help us in providing a library of architectural frameworks and transformations which will be available in our development platform.

9 Acknowledgments

We would like to thank the other members of the SE2C team for their valuable comments, especially Paul Sterges, Paris Avgeriou and Reza Razavi.

10 References

- [1] R. Kazman, "Software Architecture", *Handbook of Software Engineering and Knowledge Engineering*, World Scientific Publishing, pp. 47-68, 2001
- [2] R. Johnson, B. Foote, "Designing Reusable Classes", *Journal of Object-Oriented Programming* 1(2), pp.22-35, 1988
- [3] P. Clements, L. Northrop, *Software Product Lines: Practices and Patterns*, SEI Series in Software Engineering, Addison Wesley, Reading, MA, USA, 2001
- [4] N. Guelfi, C. Pruski, B. Ries, "A Study of Mobile Internet Technologies for Secure e-commerce Applications Development", *Techniques and Applications for Mobile Commerce (TAMOCO) part of Multi-Konferenz Wirtschaftsinformatik 2004*, Essen, Germany, p.194-2004
- [5] S.J. Mellor, M. J.Balcer, *Executable UML A foundation for the Model-Driven Architecture*, Addison Wesley, 2002
- [6] OMG, "MDA Guide Version 1.0.1", 2003
- [7] OMG, "Unified Modeling Language Specification Version 1.5", 2003
- [8] OMG, "XML Metadata Interchange Specification Version 1.2", 2002

- [9] OMG, "MOF 2.0 Query / Views / Transformations RFP", n°ad/2002-04-10, Object Management Group, 2002
- [10] A. Gerber, M. Lawley, K. Raymond, J. Steel et al., "Transformation: The Missing Link of MDA", *1st International Conference on Graph Transformation (ICGT)*, Barcelona, Spain, pp.90-105, 2002
- [11] D. Pollet, D. Vojtisek, J.-M. Jézéquel, "OCL as a Core UML Transformation Language", *WITUML 2002*, Malaga, Spain, 2002
- [12] G. Sunyé, F. Pennaneac'h, W.-M. Ho, A.L. Guennec et al. , "Using UML Action Semantics for Executable Modeling and Beyond", *13th International Conference, CAiSE 2001*, Interlaken, Switzerland, Springer Verlag, LNCS 2068, pp.433-447, 2001
- [13] N. Guelfi, G. Perrouin, P. Sterges, B. Ries, S. Sendall, "MEDAL 1.0 Reference", Applied Computer Science Department Technical Report n°TR-CST-03-01, Luxembourg University of Applied Sciences, Luxembourg-Kirchberg, Luxembourg, 2003
- [14] OMG, "Software Process Engineering Metamodel Specification version 1.0", n°formal/02-11-14, OMG, 2002
- [15] N. Guelfi, "Flexible Consistency In Software Development Using Contracts and Refinements", *2nd International Workshop on Living With Inconsistency, part of the International Conference on Software Engineering - ICSE'01*, Toronto, Canada, 2001
- [16] P. Eeles, K. Houston, W. Kozaczynski, *Building J2EE Applications with the Rational Unified Process*, Addison-Wesley Object Technology, Addison Wesley, 2002
- [17] T. Fischer, J. Niere, L. Torunski, A. Zündorf, "Story Diagrams : A new Graph Rewrite Language based on the Unified Modeling Language and Java", *Theory and Application of Graph Transformations, 6th International Workshop, TAGT'98*, Paderborn, Germany, Springer Verlag, LNCS no 1764, pp.296-309, 1998
- [18] D. Alur, J. Crupi, D. Malks, *Core J2EE Patterns: Best Practices and Design Strategies*, 15th European Conference on Object-Oriented Programming, Prentice Hall PTR, Upper Saddle River, NJ, USA, 2001
- [19] N. Guelfi, B. Ries, "Using and Specializing a Pattern-Based E-business Framework: An Auction Case Study", *The 6th Annual IASTED International Conference on Software Engineering and Applications*, Cambridge, MA, USA, ACTA Press, pp.512-522, 2002
- [20] N. Guelfi, B. Ries, P. Sterges, "MEDAL: A CASE Tool Extension for Model-driven Software Engineering", *SwSTE'03 IEEE International Conference on Software - Science, Technology & Engineering*, Hertzeliyah, Israel, 2003
- [21] IBM Rational Software, "IBM Rational XDE - Extend your development experience", <http://www.rational.com/products/xde>, 2003
- [22] B. Shannon, "Java 2 Platform Enterprise Edition Specification, v1.4", Sun Microsystems, 2003
- [23] J. Sztipanovits, G. Karsai, "Model-Integrated Computing", *Computer* pp.110-1, 1997
- [24] A. Ledeczi, M. Maroti, A. Bakay, G. Karsai et al., "The Generic Modeling Environment", *Workshop on Intelligent Signal Processing*, Budapest, Hungary, 2001
- [25] A. Agrawal, T. Levendovszky, J. Sprinkle, F. Shi et al., "Generative Programming via Graph Transformations in the Model-Driven Architecture", *OOPSLA 2002 Workshop on Generative Techniques in the context of Model-Driven Architecture*, Seattle, WA, USA, 2002