Feasibility of the Implementation of a UML to XML Evolution Architecture Using Eclipse as Technological Ecosystem

Beatriz Pérez, Ángel Luis Rubio, Gloria Yanguas
Departamento de Matemáticas y Computación
Universidad de La Rioja
La Rioja, Spain
beatriz.perez@unirioja.es, arubio@unirioja.es, gloria.yanguas@alum.unirioja.es

***Abstract*—Unified Modeling Language (UML) and eXtensible Markup Language (XML) are two of the most commonly used languages in software engineering processes. One of the most critical of these processes is that of model evolution and maintenance. More specifically, when an XML schema is modified, the changes should be propagated to the corresponding XML documents, which must conform to the new, modified schema. A current trend in this context consists of propagating the changes from the conceptual level (UML in our case) to the other levels (XML Schemas and documents). This paper is devoted to the study of the feasibility of the implementation of a UML to XML evolution architecture using the Eclipse framework, by means of UML2 and XSD plug-ins. A conclusion drawn from our study is that the chosen plug-ins lack of technological capabilities to implement this architecture.**

Keywords-evolution architecture; UML class model; XML; Eclipse plug-ins

I. INTRODUCTION

The modification of existing systems and models, in order to be adapted to requirement changes or technical advances, while maintaining the consistency between the generated artifacts, is one of the most important challenges in model-based software engineering processes nowadays [1][2].

From its origins, eXtensible Markup Language (XML) has constituted one of the most commonly used forms of representation of information covering data and metadata processing, management and retrieval. Additionally, in this context, Unified Modeling Language (UML) is widely used in the early phases (analysis, design) of development process [3][4], while the design of XML schemas are a consequence of the decisions made in those stages [5]. Different works have highlighted the importance of minimizing the effort of updating an XML document conforming to modified XML schemas [1][6]. For this reason, several authors propose to propagate the changes from the conceptual level (UML in our case) to the others levels (XML Schemas and documents) [2][5][7][8]. This approach freed analyst to make low-level implementation decisions.

In order to provide with a solution in this context, our research group undertook the search for an automated tool out of specific technological requirements. Particularly, this solution has been tackled in two different steps. Firstly, a generic architecture, named Generic Evolution Architecture (GEA), has been defined for managing those tasks when a model-driven development is followed [5]. Particularly, we focused on the transformation of UML class models to XML schema (due to the great majority of the papers that deal with this kind of transformation [9]) and provided an evolution framework by means of which the XML schema and documents are updated conforming to the changes in the UML class model.

Secondly, we need a specific implementation of such architecture in order to obtain, as a long-term goal for our approach, the development of a software tool which implements GEA. In this line, a laboratory prototype as a proof-of-concept was already developed; implementing a subset of the evolution transformations with a textual user interface, but such prototype is far away from being a complete solution. Due to the complexity of our architecture, it makes no sense to consider developing it from scratch. Particularly, we need to carry out a first task exploring partial solutions currently available in order to find one which allows us to implement our architecture. One of these possible solutions consists of using Eclipse as technological space, since it is considered to be the most versatile, plural and configurable open source Integrated Development Environment (IDE) tool.

Taking this into account, the paper aims at presenting the results obtained from the study and analysis of several existing Eclipse plug-ins used within the model-based development context, for the implementation of GEA. More specifically, a conclusion drawn from such study is that the chosen plug-ins lack of technological capabilities to implement our architecture. This fact has made us to consider new lines of research to follow-up, considering more complex tools within the own Eclipse ecosystem, which are explored in this paper.

The rest of the paper is structured as follows. The main features of GEA are presented in the following section. Section 3 is devoted to describe briefly the Eclipse plug-ins

 ISBN: 978-1-61208-247-9

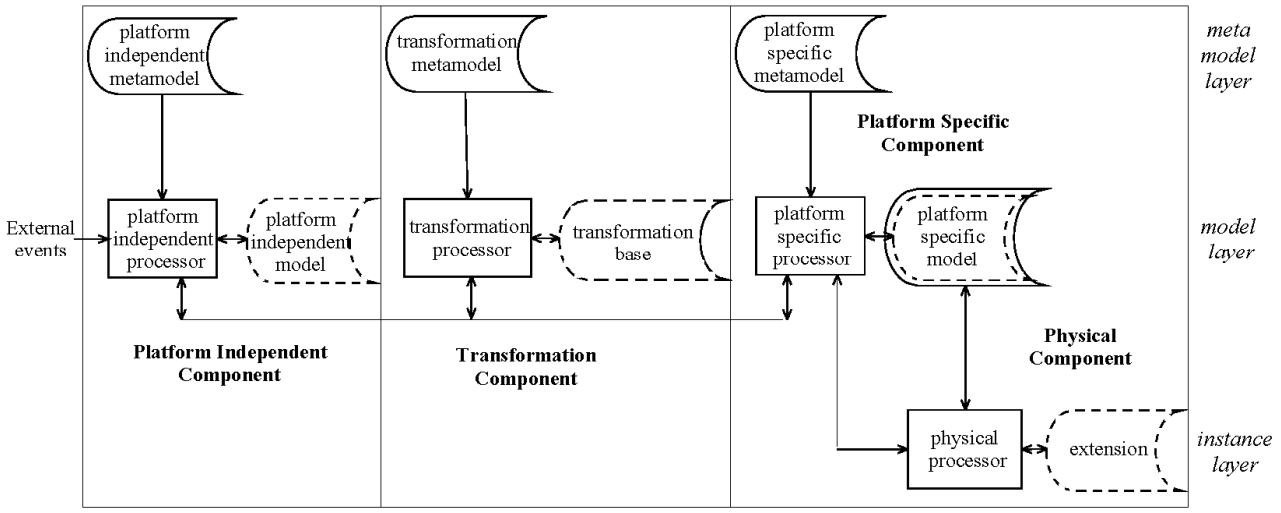

Figure 1. GEA: Generic Evolution Architecture (taken from [5]).

that have been used (UML2 and XSD). In Section 4, the proofs performed using the plug-ins, geared towards the development of GEA are explained in detail. Finally, conclusions and further work are presented in Section 5.

II. GENERIC EVOLUTION ARCHITECTURE

GEA, standing for Generic Evolution Architecture, is a generalization of a metamodel-based database evolution architecture called MeDEA and presented in [10]. As reflected in [5] "GEA keeps the characteristics of MeDEA stated in [10] and at the same time fits into a wider application context".

The main features of GEA, all of them deeply explained in [5], are enumerated in the remainder of this section.

(1) It follows the *Model Driven Architecture* (MDA) approach.
As it can be seen in Figure 1, GEA is structured around two dimensions. Vertically the different artifacts are divided into three abstraction levels which correspond with the M0, M1 and M2 layers of the 4-layer metamodel architecture pattern. Horizontally, the artifacts are identified with the developments phases established by MDA [11]. For the case of UML-XML, three specific metamodels (the Stereotyped UML Class metamodel, the UML-to-XML Transformation metamodel and the XML Schema metamodel) were proposed in [5] showing its graphical representation.

(2) The *transformation component* stores the links between the different elements of the platform independent component and the related elements of the platform specific component. It ensures the traceability of the transformation process.

(3) The *extension* to the Physical Component, propagating the evolution process from the platform specific model to the instances is another feature. Within the XML context, the XML documents are modified conforming to the evolved XML schema.

(4) *Evolution* is supported by the previous three features. Transformation and evolution process always start at the Platform Independent Component.

III. ECLIPSE

Eclipse [12] is an open source software project, which provides a highly integrated tool platform. One of the main characteristics of Eclipse is its extensibility, since it allows the user to develop plug–ins which are integrated into the core, defining a particular IDE. Eclipse has been described as "an IDE for anything, and nothing in particular [13]."

As described previously, the goal of this paper is to study the feasibility of using different plug-ins to implement the architecture explained in the previous section. Taking this into account, we have based on one of the top level projects, the Modeling Project [14], which mainly focuses on the evolution and promotion of model-based development technologies within the Eclipse community by providing a unified set of modeling frameworks, tooling, and standards implementations. More specifically, we have based on two of its subprojects: Eclipse Modeling Framework (EMF) and Model Development Tools (MDT).

On the one hand, the *EMF project* [15][16] is presented as a modeling framework and code generation facility for building tools and other applications based on a structured data model. EMF allows the user to define a model in any of three forms, Java Interfaces, UML diagrams or XML Schema, and later, generate the other forms from it, including even the corresponding implementation classes.

The purpose of the *MDT project* [17], on the other hand, is to provide an implementation of industry standard metamodels as well as tools for developing models based on those metamodels. For its interest in our work, two subprojects within this project have been considered: UML2 and XSD. *UML2* [18] is an EMF-based implementation of the UML 2.x metamodel for the Eclipse platform. Besides providing a usable implementation of the UML metamodel, it also includes a common XMI schema to facilitate interchange of models, test cases and validation rules. *XSD* [19] is a library that provides an Application Programming

 ISBN: 978-1-61208-247-9

Interface (API) for creating and manipulating W3C XML Schema and XML documents as well as an API for keeping documents conforming to their schemas as these are modified.

IV. TOWARDS AN IMPLEMENTATION OF GENERIC EVOLUTION ARCHITECTURE

This section is devoted to describe the most relevant aspects of the technological approach that have been undertaken to implement GEA. First, Eclipse Indigo (v 3.7.1) and the plug-ins UML2 Extender SDK (v 3.2.1) and XSD - XML Schema Definition SDK (v 2.7.1) have been installed in order to perform this implementation. Besides, the examples followed in [16] and commonly used in papers that deal with UML and XML [7][20] have been used as a benchmark (for instance, Simple Purchase Order, the Primer Purchase Order –PPO- and Extended Purchase Order –ExtendedPO-).

Before going into detail on the parallelism between these proofs and the architecture shown in Section 2, let us define some concepts related to EMF. "An *EMF model* is essentially the Class Diagram subset of UML [16]." "The model used to represent models in EMF is called Ecore. Ecore is itself an EMF model, and thus is its own metamodel [16]." Due to lack of space the graphical representation of the metamodels used in this section are not showed. Anyway, a comprehensive explanation on the Ecore metamodel can be found in [16], and it is stored in the file Ecore.ecore (in turn, contained in the file org.eclipse.emf.ecore_2.7.0.v20120127-1122.jar).
Essentially, the Ecore metamodel defines four types of objects:

EClass is used to represent a modeled class. It is identified by name and can have a number of attributes and references. A class can refer to a number of other classes as its supertypes.

EAttribute models attributes. It is identified by name and has a type.

EDataType models the type of an attribute. It is used to represent simple types whose details are not modeled as classes.

EReference is used to represent one end of an association between classes. It has a name, a boolean flag to indicate if it represents containment, a lower and upper bounds to specify multiplicity and a reference (target) type, which is an EClass. Besides, related classes and data types are grouped in *EPackage*, which is the root element of a serialized Ecore model.

In our implementation, both *the platform independent metamodel* and *the platform specific metamodel* are Ecore models. The first (uml.ecore) is included in the UML2 plug-in (org.eclipse.uml2.uml_3.2.100.v201108110105.jar). It consists of an `EPackage`, 247 `EClass`, 13 `EEnum` (which is a subclass of EDataType and it is used to model enumerated types) and 4 primitive types. On the other hand, the models generated into the Platform Specific Component will be conformed to the metamodel xsd.ecore provided in the XSD plug-in, within org.eclipse.xsd_2.7.1.v20120130-0943.jar. It is simpler than the previous metamodel, and it consists of an `EPackage`, 57 `EClass`, 20 `EEnum` and 5 primitive types.

The UML class model is transformed to an EMF model and afterwards the EMF model is transformed to an XML schema. The Eclipse framework defines and has total control of these *transformations*. In particular, the UML2 project defines a mapping from UML 2.0 to Ecore. A similar mapping, except subtle details, is described for UML version 1.4 in [16]. This mapping only concerns with the constructs of UML classes. Broadly speaking, a Package maps to an `EPackage`; a class is mapped to an `EClass`, `EEnum`, or an `EDataType`, depending on the class's stereotype; an attribute maps to an `EAttribute` and an operation maps to an `EOperation`, which models the behavioral features of an EClass. It is worth noting that an UML association maps to two `EReferences` and each of them has the other as its `eOpposite`. Taking these results into account, we have demonstrated that is not possible to map an association class. Furthermore, we have also proved that the UML model and its EMF counterpart are not automatically synchronized.

Detailed information about how the second transformation, from EMF model to XML schema, is performed can be obtained from [16]. At a high level, the mapping is as follows: an `EPackage` maps to a schema, an `Eclass` maps to a complex type, an `EDataType` maps to a simple type, an `EAttribute` and an `EReference` map to an attribute or element declaration.

With respect to the *instance layer*, the XSD plug-in provides the tools for creating XML documents from an XML schema and for validating them if the XML schema changes.

Taking this into account, we can conclude that the analysis and tests carried out on these tools confirm us that their expected capacities seem to be far away from those really supported, at least regarding models synchronization. Regarding *evolution*, although intuition points out to get similar results, we plan to follow a source-like approach, that is, propagating the changes from the UML class model (so this work must be considered 'in progress').

V. CONCLUSION AND FUTURE WORK

In this paper, the possibility of implementing a UML to XML Evolution Architecture by means of three of the plug-ins that seem to be the most appropriate ones (EMF, UML2 and XSD) have been studied.

These plug-ins have been tested founding several difficulties described below. To transform a UML class model to XML Schema is required an intermediate transformation to an EMF model. Besides, to update the EMF model conforming to the changes in the UML model is a very complex task for which it is necessary to be a specialized expert in adapters and notifiers.

It is worth noting that EMF only concerns itself with a small subset of UML. For instance, a UML class model that contains a class association cannot be mapped to an EMF model. Therefore, this process is valid only for specific UML class models. Finally, we want to note that all the

 ISBN: 978-1-61208-247-9

transformations are automatically carried out; to create and manage our own transformation rules is not possible.

For all these reasons, in despite of these plug-ins provide us with a lot of structures and procedures, we conclude that the development of our evolution architecture using only and directly these plug-ins is an exceedingly complex task, and it may not even be feasible.

There exist several possibilities for follow-up this implementation:

(1) To use the Ecore metamodel as the Platform Independent Metamodel. In this case, we give up the richness of UML since Ecore is a small subset of UML. We would like to advance that we have already obtained some preliminary results in this line, which lead us to think that our impressions about using this metamodel to implement our GEA architecture are justified.

(2) To create our own transformations by means of Atlas Transformation Language (ATL) and MofScript, both of them subprojects of the Eclipse Modeling Project. Thereby we would have under control the transformations of each element. Regarding this line of work, we have experience in using both tools (ATL and Mofscript) for the particular case of implementing a framework that automatically generates decision support systems for clinical guidelines [21]. This experience makes us to think that they are feasible solutions for our implementation problem, but we are aware that it is required a conceptual task to align our transformation approach with these tools.

(3) To explore the Hypermodel plug-in [22], in order to know if it could be used to implement our architecture. Hypermodel was designed and implemented by David Carlson and it is stated that generates XML schemas from any UML model. This fact leads us to think that this plug-in could be a good reference among other existing tools.

ACKNOWLEDGMENT

This work has been partially supported by Spanish Ministry of Science and Innovation project TIN2009-13584 and University of La Rioja, project API12/12.

The authors would like to thank the others authors of [5] for their contributions: Eladio Domínguez, Jorge Lloret, Áurea Rodríguez, María A. Zapata.

REFERENCES

[1] G. Guerrini, M. Mesiti and D. Rossi, "Impact of xml schema evolution on valid documents", in: WIDM '05: Proceedings of the 7th annual ACM international workshop on Web information and data management, ACM, NY, USA, 2005, pp. 39-44. http://doi.acm.org/10.1145/1097047.1097056.

[2] M. Klettke, "Conceptual XML schema evolution - The CoDEX approach for design and redesign", in M. Jarke, T. Seidl, C. Quix, D. Kensche, S. Conrad, E. Rahm, R. Klamma, H. Kosch, M. Granitzer, S. Apel, M. Rosenmüller, G. Saake, O. Spinczyk (Eds.), Workshop Proceedings Datenbanksysteme in Business, Technologie und Web (BTW 2007), Aachen, Germany, 2007, pp. 53-63.

[3] T. Krumbein and T. Kudrass, "Rule-based generation of XML schemas from UML class diagrams", in: Proceedings of the XML Days at Berlin, Workshop on Web Databases (WebDB), 2003, pp. 213-227.

[4] I. Kurtev, K. V. Berg and M. Aksit, "UML to XML-schema transformation: a case study in managing alternative model transformations in MDA", in: Proceedings of the Forum on specification and Design Languages (FDL'03), European Electronic Chips & Systems Design Initiative, Frankfurt, Germany, 2003.

[5] E. Domínguez, J. Lloret, B. Pérez, Á. Rodríguez, Á. L. Rubio and M. A. Zapata, "Evolution of XML schemas and documents from stereotyped UML class models: A traceable approach", Information and Software Technology. Vol. 53, no. 1, pp. 34-50, 2011.

[6] D. K. Kramer, "XEM: XML evolution management", Ph.D. thesis, Worcester Polytechnic Institute (2001).

[7] N. Routledge, L. Bird and A. Goodchild, "UML and XML schema", in: X. Zhou (Ed.), Thirteenth Australasian Database Conference (ADC2002), ACS, Melbourne, Australia, 2002.

[8] R. Elmasri, Q. Li, J. Fu, Y.-C. Wu, B. Hojabri and S. Ande, "Conceptual modeling for customized XML schemas", Data Knowl. Eng. 54 (1) (2005) 57-76.

[9] E. Domínguez, J. Lloret, B. Pérez, A. Rodríguez, A. L. Rubio and M. A. Zapata, "A survey of UML models to XML schemas transformations", in: Proceedings of the Web Information Systems Engineering (WISE) Conference, Vol. 4831 of Lecture Notes in Computer Science, Springer, 2007, pp. 184-195.

[10] E. Domínguez, J. Lloret, A. L. Rubio and M. A. Zapata, "MEDEA: A database evolution architecture with traceability", Data and Knowledge Engineering 65 (3) (2008) 419-441. doi:10.1016/j.datak.2007.12.001.

[11] J. Mukerji and J. Miller, "MDA guide version 1.0.1", available at http://www.omg.org/cgi-bin/doc?omg/03-06-01/ 19.11.2012 (June 2003).

[12] The Eclipse Foundation, "Eclipse - The Eclipse foundation open source community website", available at http://www.eclipse.org 19.11.2012

[13] Object Technology International, Inc., "Eclipse platform technical overview", available at http://www.eclipse.org/whitepapers/eclipse-overview.pdf 19.11.2012 (February 2003).

[14] The Eclipse Foundation, "Eclipse modeling project", available at http://www.eclipse.org/modeling/ 19.11.2012

[15] The Eclipse Foundation, "EMF", available at http://www.eclipse.org/projects/project.php?id=modeling.emf.emf 19.11.2012

[16] D. Steinberg, F. Budinsky, M. Paternostro and E. Merks, EMF: Eclipse modeling framework, Addison-Wesley, 2008.

[17] The Eclipse Foundation, "Eclipse modeling - MDT", available at http://www.eclipse.org/modeling/mdt/ 19.11.2012

[18] The Eclipse Foundation, "MDT-UML2", available at http://www.eclipse.org/modeling/mdt/?project=uml2 19.11.2012

[19] The Eclipse Foundation, "MDT-XSD", available at http://www.eclipse.org/projects/project.php?id=modeling.mdt.xsd 19.11.2012

[20] W3C, "W3C XML schema definition language XSD", available at http://www.w3.org/TR/2012/REC-xmlschema11-1-20120405/ 19.11.2012 (April 2012).

[21] E. Domínguez, B. Perez and MA Zapata. Towards a traceable clinical guidelines application. Methods Inf Med 2010; 49: 571–580.

[22] D. A. Carlson, "hyperModel|XMLmodeling.com", Available at http://xmlmodeling.com/hypermodel 19.11.2012 (2012).

 ISBN: 978-1-61208-247-9