Model Weaving Support for Migrating Software Artifacts from AUTOSAR 2.0 to AUTOSAR 2.x

Juan M. Vara Mesa[1], Marcos Didonet Del Fabro[2], Frédéric Jouault[2], Jean Bézivin[2]

1: Kybele Research Group (University Rey Juan Carlos), Madrid (Spain)
2: ATLAS Group (INRIA & LINA), University of Nantes (France)

Abstract: At a recent meeting, the TNI-Software Company presented the following problem: "An organization has developed a huge quantity of software artifacts conforming to the AUTOSAR 2.0 standard. They wish to migrate these to a new version of the AUTOSAR standard. Which is the best way to proceed in order to automate this migration?". This article presents one solution to this problem using tools from the Eclipse Modeling Project. One of them is the ATL transformation language from the M2M project. The second one is the AMW model weaving tool from the GMT project. The solution provided is quite generic and it can be used to assist in the migration of different versions of software systems. This migration is particularly important because these operations are more and more frequent and error prone. They have lost their exceptional status to become part of the usual day-to-day software maintenance process. Furthermore their boring and repetitive characteristics make them ideal candidates for automation.

Keywords: Software Artifacts Migration, Model Driven engineering, Model transformation, Model Weaving

1. Introduction

Frequently, the set of software artifacts of some kind change according to software evolution: new elements are added, previous elements are ruled out while others just remain, but suffering from some modifications, etc. This task can be very tedious and depending on the complexity of the software artifacts, even impossible to be carried out by hand in limited time.

In this article we show how we handle this task in a real project developed as a response to a problem presented by a company member of AUTOSAR ([8] & [10]).

An overview of the problem is presented in Figure 1. Suppose that AUTOSAR defines a new version of the AUTOSAR standard, which describes the System, Software and Hardware of an automobile. We may want to migrate artifacts conforming to the present version of the standard (AUTOSAR 2.0), to the new one (AUTOSAR 2.x). In this case, the software artifacts handled are models. Additionally, the size of the AUTOSAR standard (around 5000 elements) increases the complexity of the project.

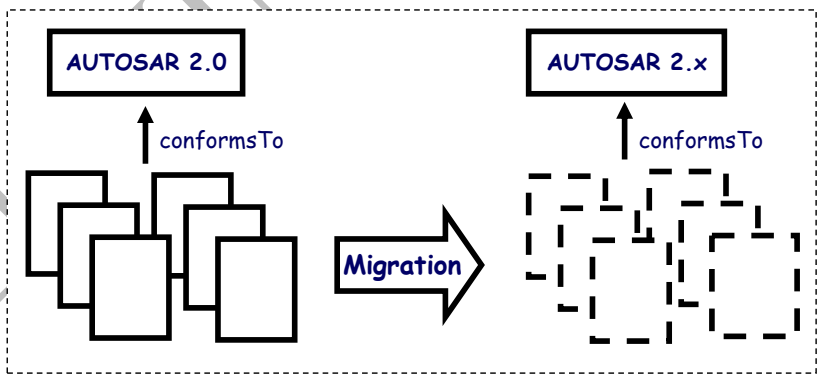

Figure 1. Problem overview

There are several possible solutions for this kind of problem:

- The simplest one is to do it by hand: for every software artifact developed according to the AUTOSAR 2.0 standard, a new software artifact according to the new version has to be developed manually. Obviously, this is the less recommendable solution from the point of view of Software Engineering: it is prone to errors and it is not replicable. This solution is even less suitable due to the size of any version of the AUTOSAR standard.
- The second solution is to develop an object-oriented program (for instance a JAVA program) capable of generating an AUTOSAR 2.x software artifact from an AUTOSAR 2.0 one. However, this approach has similar problems: it is still very prone to errors; it is a very tedious task and it is an ad-hoc solution. New programs need to be developed for every new migration problem. The

reusability proposed here by object-oriented technology can only bring some limited help in this situation.

- Since the software artifacts considered in this problem are models, a better solution is to develop a model transformation program to support the migration task. For each element of an AUTOSAR 2.0 model, the program will create the corresponding elements in the AUTOSAR 2.x model. Nevertheless, although the development cost of a model transformation program is lower than the one from using a general programming language, it is still too costly to develop this program from scratch. Finally, if new versions of the standard are defined, we need to define a new program (a new model transformation) to support the migration of software artifacts to the new version of the standard.

All these solutions share a set of drawbacks: they are very tedious, they are prone to errors and they are ad-hoc solutions: new programs have to be developed if new versions of the standard have to be considered.

The last approach is the path we follow in this article. The idea is to generate automatically the model transformation that will do the migration. This way, we avoid the tedium derived from developing the program from scratch and we reduce the risk of introducing errors in the program. Moreover, we are able to generate new solutions (i.e., new model transformation programs) to support the migration for new versions of the standard. So, the solution proposed is to define an automatic process to generate a model transformation that will support the migration of software artifacts.

Here we show how the problem of migration may be solved in a (semi-)automatic way by using a set of Eclipse plug-ins [6] developed in the framework of the Eclipse Modeling Project (EMP). More specifically, we use the Eclipse Modeling Framework (EMF) [1], the ATLAS Transformation Language (ATL) [8] and the ATLAS Model Weaver (AMW) [4]. With these tools we are capable of defining an automatic solution for the problem of model migration. Likewise we are able to demonstrate how the tools scale by applying them to the difficult migration of AUTOSAR artifacts.

2. Model Transformation, Model Weaving and Cumulative Weaving

We propose using Model Driven Engineering (MDE) [13] techniques to solve the migration problem. The basic assumption in MDE is to consider models as first class entities, just as classes are the basic construction block in object-oriented programming, or software components are the basic unit in component-based software engineering. We translate our problem into a model engineering problem and then we use the existing tools in the field of MDE to solve it. So, we consider any software artifact as a model and the set of constraints directing its definition as a metamodel. We explain this idea bellow. More specifically, the solution proposed is based on two concepts from the MDE domain: model transformation and model weaving. Next, we give a brief overview of these concepts in order to help in the understanding of the proposal.

2.1. Model Driven Engineering

A model is a software artifact that represents a system or given aspect of a system. As every program conforms to the grammar of its respective programming language, each model conforms to a **metamodel** [1]. The metamodel describes the various kinds of elements that can be included in a model and the way they are arranged, related and constrained. In other words, a metamodel is a model that describes the structure of models. A model that is valid according to the corresponding metamodel is said to *conform* to the metamodel, just as a program can be syntactically correct according to the respective programming language. For instance, in Figure 2 Ma and Mb are models conforming to the MMa and MMb metamodels. This relation is represented by the *c2* (conforms to) association. Back to the specific problem addressed here, the AUTOSAR 2.0 and AUTOSAR 2.x standards are the metamodels. The software artifacts we want to migrate are models conforming to these metamodels.

2.2. Model Transformation

Nevertheless, working with multiple, interrelated models requires significant effort to accomplish some tasks related with model management, such as refinement, consistency checking, refactoring, etc. Many of these activities can be performed as automated processes, which take one or more source models as input and produce one or more target models as output, following a set of transformation rules. We refer to this process as **model transformation** [12].

As shown in Figure 2, the MMa2MMb model transformation transforms a model Ma into another model Mb. To do so, it specifies a set of rules that encodes the relationships between the elements from the MMa and MMb metamodels. The model transformation is defined at metamodel level, i.e., it maps elements from the input and output metamodels.

So, it can be used to generate an output model from any set of models conforming to the input metamodel. In other words, the model transformation program works for any model defined according to the input metamodel.

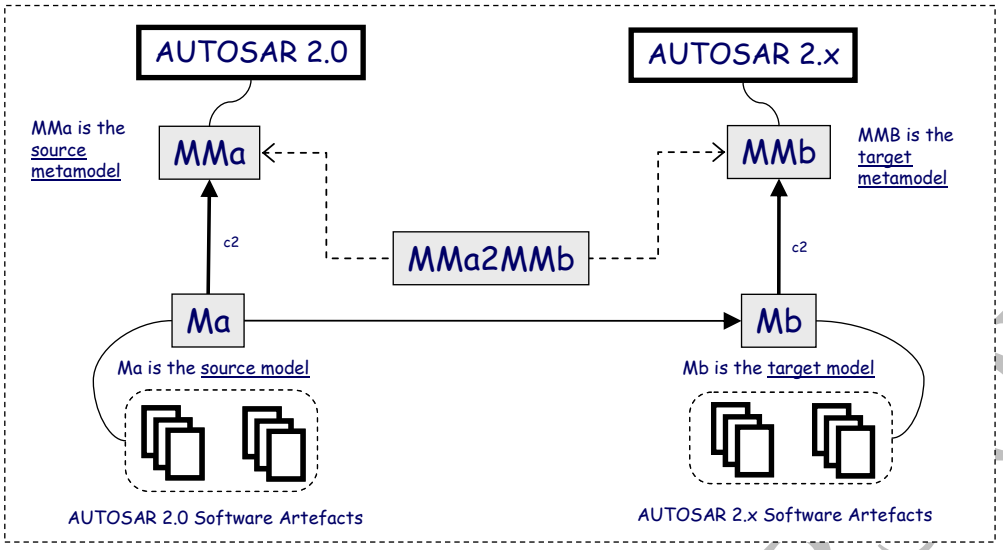

Figure 2. Model Transformation overview

This way, a model transformation developed to solve the AUTOSAR problem specifies which elements from the new version of the AUTOSAR standard correspond to a given element from AUTOSAR 2.0. Once we create the AUTOSAR2.0_2_AUTOSAR2.x model transformation, any AUTOSAR 2.0 software artifact can be migrated to an AUTOSAR 2.x software artifact by executing this program.

2.3. Higher-Order Transformations

If we define the set of rules and constraints that drives the construction of a model transformation in a metamodel, any model transformation can be expressed as a model conforming to that metamodel. Then, any model transformation can be the input or output of another model transformation. We use a specific term to refer to this type of model transformations: a **Higher Order Transformation** (HOT) is a special kind of model transformation whose input or/and output is a model transformation. As we will see, we use an ATL HOT in our solution for the migration problem.

2.4. Model Weaving

Model transformation is essentially intended to define executable operations. Hence it is not always adapted to define and to capture various kinds of relationships between models elements. However, we often need to establish and handle these correspondences between the elements of different domains, each one defined by means of a model. The correspondences may be informal, incomplete, and preliminary. In many cases they may not be used directly to drive an executable operation. Model weaving is the process of representing, computing, and using these initial correspondences.

A set of correspondences between different model elements is represented as a **weaving model** [1]. The computation of a weaving model is named **model matching**. This may be done manually, automatically or more frequently semi-automatically by a heuristic. A **matching heuristic** may be implemented by way of a model transformation. Once a weaving model has been computed, it may be used for a number of purposes.

One typical way to use a weaving model is as the source of a transformation τ that will produce as an output another transformation θ that may be executed to produce the intended result. Let us suppose for example that the weaving model contains equivalence links that state that some elements of two models Ma and Mb represent exactly the same entity. This declarative information contained in the weaving model may be used in different ways. For example one may wish to compute the intersection or the union or the symmetric difference of Ma and Mb. The effective computation of the union, intersection, or symmetric difference will be performed by the transformation θ that will take Ma and Mb as input. But the exact definition of what we mean by intersection, union of symmetric difference of two models is provided by the definition of the transformationτ. Of course these definitions are more complex than simple set operations.

The information captured by a weaving model may be used in different application scenarios, such as tool interoperability, transformation specification, traceability, model merging or model comparison. In the context of Model Driven Engineering, the way to manage this kind information is using weaving models.

A **Weaving Model** is thus a special kind of model used to establish and handle the links between models elements. The matching operation is a special type of operation that, taking two or more models as input, produces a weaving model as output.

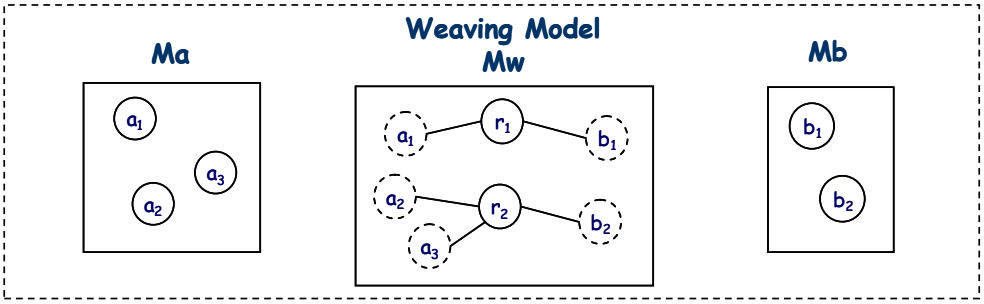

Figure 3. Model Weaving overview

This model stores the links (i.e., the relationships) between the elements of the input models. We illustrate this idea in Figure 3: Mw is a weaving model that captures the relationships between Ma and Mb, denoted by the triple [Mw, Ma, Mb]. Then, each element of Mw links a set of elements of Ma with a set of elements of Mb. For instance, the r_2 element of Mw defines a relationship between a_2 and a_3 from Ma, and b_1 from Mb.

Concerning on the migration problem, we need to establish the relationships between the elements of AUTOSAR 2.0 and AUTOSAR 2.x standards. Furthermore, if we express them as a set of links contained in a weaving model, we are capable of using the information provided by that model to generate the program (the model transformation) that supports the migration. Note that what we have just mentioned could also be applied to metamodel weaving, since every metamodel is indeed a model. So, the generic term of model weaving covers both cases: model and metamodel weaving.

There are several ways to establish the relationships between the elements of two different models, i.e., to create a weaving model. Again, the simplest one is to do it by hand: given those models, one may create an empty weaving model and then add the links one by one. Nevertheless, this approach has similar problems as the manual solution: it is very tedious and it is very prone to errors. The next option is using a generic programming language to develop an ad-hoc program that creates automatically the links in the weaving model. However, we would have to develop new solutions whenever new models have to be considered. Finally, the best option is again to define an automatic process that, given two input models, produces a weaving model containing the links between their elements. In order to provide this functionality, we propose the **matching** process explained below.

2.5. Matching

In the context of model weaving, **matching** is a semi-automatic process for creating links between models elements by applying some hypothesis about the input models [1].

The process is depicted in Figure 4 (*Simple Weaving*). The basis of this process is the execution of a given heuristic that aims to establish the relationships between the elements of the input models. This heuristic is executed by means of a model transformation that takes as input the models we wish to relate, and that generates the weaving model as output. For instance, a given heuristic may create a link between model elements that have the same name in the input models. However, heuristics are inexact by nature. As a matter of fact, heuristics are defined as "a technique designed to solve a problem that ignores whether the solution can be proven to be correct, but which usually produces a good solution". Therefore, the links contained in a weaving model generated by a matching process can be inaccurate. This is why usually the weaving model resulting from a matching operation is considered as a proposal that a human operator may validate, invalidate or update. Another way to improve the result of the matching process is to execute it several times with different heuristics. We call this process cumulative weaving.

2.6. Cumulative Weaving

As described in Figure 4, **cumulative weaving** is a process in which several matching operations are executed sequentially to improve the accuracy of the final weaving model [5]. Each matching executes a new heuristic to filter and refine the information collected in the links from the previous weaving model and stores the result in a new weaving model. These transformations take as input not only the models that we want to relate, but also the previous weaving model. Consequently, the last weaving model obtained is the most accurate one since it is the result of several refinement and filter operations.

For instance, one heuristic may compute a similarity value by comparing the name of each element of the input models. Then, the output weaving model includes links denoted by the tuple [Ma-E1, Mb-E2, X], i.e. the similarity between the names of the E1 element from Ma and the E2 element from Mb is X (a real value in the 0..1 range). The subsequent heuristic may generate a new weaving model that deletes the links whose similarity value is smaller than a given threshold, e.g. 0.173. The relationships represented by the links contained in the second weaving model may thus be considered as more precise.

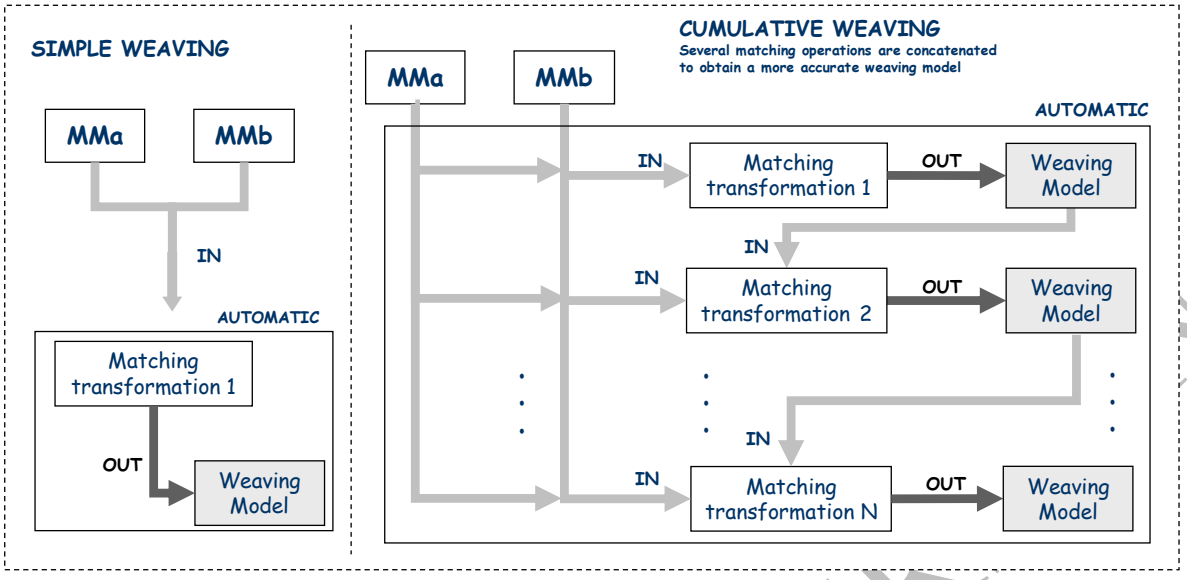

Figure 4. Simple and Cumulative Weaving overview

Cumulative weaving has been successfully applied to some problems including the one presented in this paper. However a systematic study of convergence and impact of the permutation of heuristics in the chain of matching steps falls beyond the scope of the present work.

3. An approach for solving the software artifacts migration problem

The migration problem we address in this article can be generalized as: "*given two different standards for developing a software artifact, find a way to translate a software artifact developed according to one of them, to a software artifact developed according to the other one*". We propose translating this problem into a model engineering problem to solve it using some existing tools proved to work properly. In the MDE context this problem is stated as: "*given two metamodels, find a way to map one model conforming to one of the metamodels to a model conforming to the other one*". In the remainder of this paper, we summarize our proposal, referring to the specific problem of migration between two different versions of the AUTOSAR standard.

First, a matching process based on iterative weaving is carried out over the two different metamodels (the two different versions of the AUTOSAR standard). The result is a weaving model containing the links between the elements of those metamodels; then, the information gathered from the matching process, collected in the weaving model, is used as input to generate a model transformation. This model transformation supports the migration of models conforming to one metamodel (the present version of the standard) to the other one (the new version of the standard).

Notice that this approach is easily generalizable for different contexts. As a matter of fact the program generated to support the migration (the model transformation program) is obtained from the weaving model. So, the accuracy of the weaving model generated through an iterative weaving process is very important. However, the number and the nature of the heuristics to be executed are not fixed. So, for each specific domain, one may add new heuristics, or just modify the already existing ones in order to improve the result of the iterative weaving process. This way, we can generate a more accurate weaving model for each specific context.

4. The Tools

To deploy the proposed solution for the migration problem, we use a set of Eclipse plug-ins [6] developed in the context of the Eclipse Modeling Project [7]. In this section we present an overview of these tools.

- The **Eclipse Modeling Framework** (EMF) project [3] is a modeling framework and code generation facility for building tools and other applications based on a structured metametamodel. From a model specification described in the XMI standard [10], EMF provides tools and runtime support to produce a set of Java classes for the model, a set of adapter classes that enable viewing and command-based editing of the model, and a basic editor. EMF provides a set of tools to import models defined in several formats into the XMI standard. Models that are specified using XML documents, annotated Java or modeling tools like Rational Rose can be automatically imported into EMF (in section 5.1 we show how this facility is used in this work). Once imported, the models are expressed in the *.ecore* format. Besides, since EMF provides the foundation for interoperability with other EMF-based tools and

applications, it is used as underlying technology in a wide range of tools in the context of model engineering.

- The **ATLAS Transformation Language** (ATL) is a model transformation language and toolkit developed by the ATLAS INRIA Group in Nantes that provides ways to produce a set of target models from a set of source models [9]. Developed within the Eclipse platform, the ATL Integrated Environment (IDE) comprises a number of standard development facilities (syntax highlighting, debugger, editor, etc.) that eases the development of ATL transformations. To define and handle the metamodels and models used when developing a model transformation, the ATL IDE uses the model management facilities provided by EMF.
- The **ATLAS Model Weaver** (AMW) is a tool for establishing relationships (i.e., links) between models elements [4]. These links are stored in a weaving model that is created conforming to a weaving metamodel. To be acceptable as a weaving metamodel, a metamodel should extend a core weaving metamodel providing basic definitions for links and linked elements. Each application domain has different needs that must be considered by the developers, so AMW proposes a set of weaving metamodels and the users may enrich this basic library. Like ATL, AMW uses the underlying EMF repository for model management.

5. Motivating problem: solving the AUTOSAR model migration problem

In this section, we describe how our solution is deployed in the AUTOSAR context. The process is showed in Figure 5. The main part is the iterative weaving described on the left hand side of the picture. As a result, we obtain a weaving model that establishes the relationships between the elements of the present version of the AUTOSAR standard (AUTOSA 2.0) and a new version that we have developed for this project (so-called AUTOSAR 2.x). In the right hand side, the AUTOSAR2.0_2_AUTOSAR2.x model transformation encodes these relationships in a set of rules. In the following, we describe each step of this process.

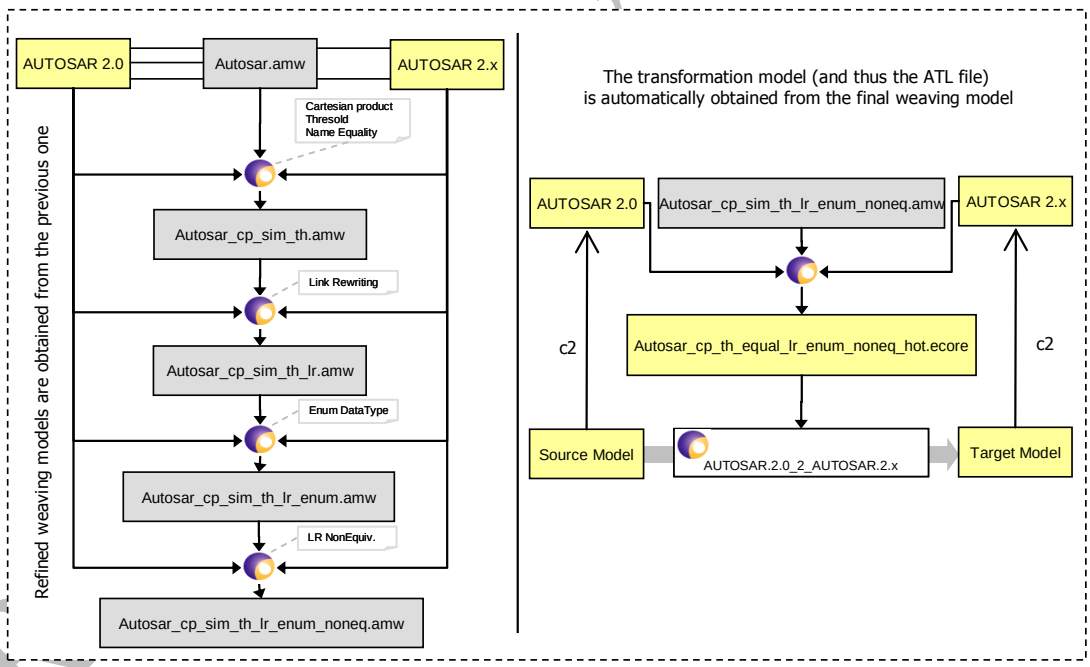

Figure 5. AUTOSAR Project overview

- The first step is to define a new weaving model (Autosar.amw). This weaving model contains just a reference to the AUTOSAR metamodels, as well as a root element.
- Next, we establish the relationships between the elements of the AUTOSAR standards by means of an iterative weaving process that sequentially execute a set of matchings to obtain an accurate weaving model.
- The first weaving model (*Autosar_cp_sim_th.amw*) is obtained by executing an ATL model transformation that takes as input the AUTOSAR 2.0 and AUTOSAR 2.x metamodels, as well as the previous weaving model (the empty weaving model). The resulting weaving model comprises a set of links between the elements of the AUTOSAR standards according to the next set of heuristics:

1. Restricted Cartesian product: this heuristic states that there is a relationship between every pair of elements that have the same type. The resulting weaving model includes a link between each pair of classes from the AUTOSAR 2.0 and AUTOSAR 2.x metamodels. The same for attributes, references and so on. For instance, in the resulting weaving model there is a link that relates the class *OperateInterface* from AUTOSAR 2.0 with each one of the classes from AUTOSAR 2.x, but no link relates this class with a method, an attribute or a reference from AUTOSAR 2.x.

2. Name Similarity: this heuristic assigns a numerical value to each one of the links defined by the restricted Cartesian product by comparing the name of each element referenced by the link. Consider that the previous weaving model includes a link relating the class *OperateInterface* from AUTOSAR 2.0 and the class *OperatingInterface* from AUTOSAR 2.x, and another one relating the same class from AUTOSAR 2.0 with the class *DataInterfaceWidth*; after applying the Name Similarity heuristic, the similarity value for the first link is 0.85 while for the second one it is 0.35.

3. Threshold selection: this heuristic states that all the identified relationships whose similarity value is under a given threshold should not be considered. So, the corresponding links have to be deleted from the resulting weaving model. For instance, if we set a 0.75 threshold value, the previous link between the *OperateInterface* and the *DataInterfaceWidth* is deleted from the resulting weaving model, while the link between the *OperateInterface* and the *OperatingInterface* remains[1].

Note that instead of using a different model transformation to execute each heuristic, we are integrating the execution of the three heuristics in one ATL file that works as follows: first, the restricted Cartesian product creates a link between the elements of the same type. Next, the Name Similarity assigns a similarity value to the new link. Then, the threshold test is executed. If the test is not passed, the link is deleted. More correctly, it is not included in the resulting weaving model. Therefore, we avoid navigating the whole weaving model to compute the threshold over each link. We compute it at the same time the links are created. This way, we improve the performance of the heuristics execution. This is very important, since the size of the AUTOSAR standards may cause out of memory errors when we try to execute each heuristic separately. The table below shows the number of elements of each AUTOSAR metamodel. These data help on the understanding of the result of computing a (restricted) Cartesian product over the input metamodels.

	AUTOSAR 2.0	**AUTOSAR 2.x**
CLASSES	*700*	*1020*
ATTRIBUTES	*2262*	*3254*
REFERENCES	*1607*	*2086*

As a matter of fact, the weaving model obtained when we execute just the Cartesian product and the Name Similarity heuristics would contain more than 1.5 millions of links, while the weaving model obtained after applying also the *Threshold and Name Similarity* heuristics contains only 28.501 links.

- In the next step of the process, a new model transformation produces a new weaving model (*Autosar_cp_sim_th_lr.amw*) by reorganizing the links according to the containment relationships between Classes, Attributes, and References. Consider the example shown in Figure 6. The *Autosar_cp_sim_th.amw* model contains a link between a couple of classes and another one between a couple of attributes contained in those classes. Those corresponding links are reorganized in the resulting *Autosar_cp_sim_th_lr.amw* model to show this containment relationship. Later on, when the model transformation is generated, these links are translated into a rule for mapping the classes (*PortHW_2_PortHardware*) that includes a binding between the attributes (*PortHardware.shortName* = *PortHW.shortName*).
- The cumulative weaving process continues by producing the *Autosar_cp_sim_th_lr_enum.amw* model. Here, we add a set of links on the weaving model to map the *enum data types* included on both versions of the AUTOSAR standard. This step illustrates how our solution can be customized for each specific migration problem. We introduce this step to solve some problems that were found after the initial tests. The first ATL transformation obtained as the result of the process was not able to map properly the *enum data types* included in the AUTOSAR metamodels. Once again, these problems arose due to the special nature of the standards.
- The last heuristic adds a new set of links in order to identify the elements that do not have equivalence in the AUTOSAR2.0 and AUTOSAR2.x metamodels. This way, we identify which elements will not be handled by the ATL program that carries out the migration task. As a matter of fact, this information is not used for generating the model transformation but

[1] Of course a documented trace of all these decisions may be produced for validation purposes. More generally this falls under the various possibilities of establishing trace models of such processes.

for improving it at the end of the process. If we know exactly which elements are not mapped by the model transformation, we are able to refine it in order to add the code for handling those elements.

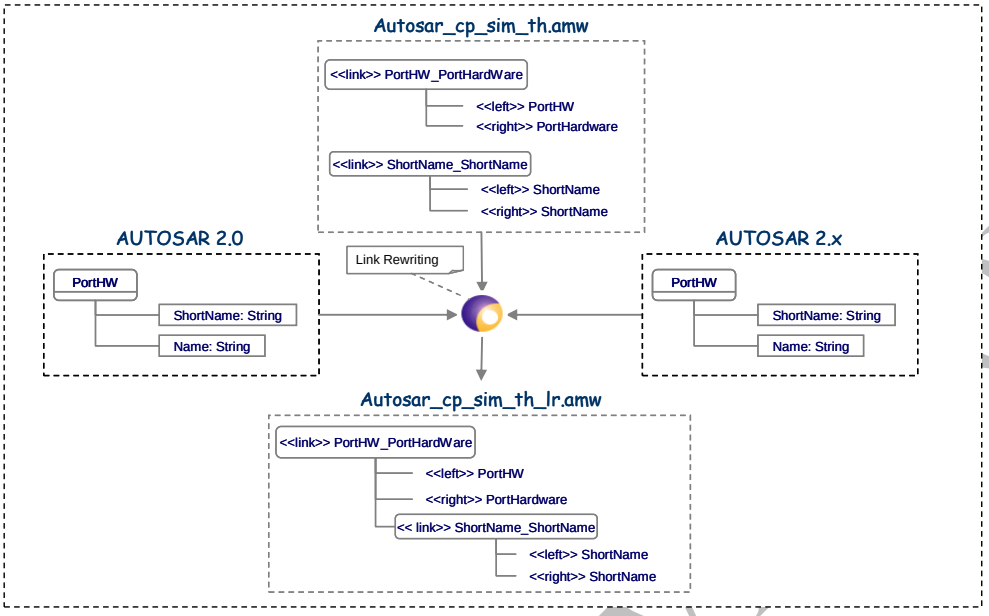

Figure 6. Link Rewriting result

- At the end of this iterative process we obtain the final weaving model. From there we are ready to generate the ATL program that supports the migration task. We develop a higher order transformation that translates the relationships included in the final weaving model into a set of transformation rules collected into an ATL model.
- Finally, using the functionality provided by the AMW plug-in, the ATL model is directly transformed into the corresponding ATL program. This is just a matter of simple translation. This ATL program supports the migration of a model from AUTOSAR 2.0 to AUTOSAR 2.x by encoding the relationships between the elements of both metamodels into 430 transformation rules. It may be executed in the Eclipse environment using the ATL plug-in.

5.1. Maximizing the automation level by means of EMF

Organizations following traditional development approaches are not used to work with models and metamodels, which are the basic of MDE. However, there is no need for a drastic change in the working habits of these organizations in order to take advantage of the MDE principles. There are different facilities that help translating any software development problem to a model engineering problem. In this section, we show how we may use some of the functionalities provided by EMF.

Since the AUTOSAR 2.0 standard is collected in an XML schema, we use another XML Schema to define AUTOSAR 2.x, the new version of the standard. Thus, the software artifacts developed according to those standards are XML documents conforming to those XML schemas. To be able to work with them in a model engineering context, we need to translate them into models and metamodels. More specifically, given that the tools used here are based on the EMF facilities for model management, any model or metamodel we want to use along the development process have to be expressed as an *.ecore* file, the XMI format used by EMF to express and collect models.

Nevertheless, coding an XMI file by hand is a very tedious and error-prone task. So, we use the functionalities provided by EMF to improve this part of the process.

As shown in Figure 7, we generate the AUTOSAR 2.0 and AUTOSAR 2.x *.ecore* files from the respective XML Schemas. These *.ecore* files are the AUTOSAR 2.0 and AUTOSAR 2.x metamodels we use during the process. Furthermore, EMF provides with a set of functionalities to manipulate the *.ecore* files, including a visual editor.

To test the ATL program that has been generated, we need to express the XML documents (the software artifacts to migrate) as models and more specifically, as models that an ATL program can handle. Since the ATL engine runs on top of EMF, all we need is to translate these XML documents to XMI files. Again, this translation is an automatic process. Using the functionality provided by EMF we are able not only to create an *.ecore* file from any XML Schema, but also to generate *.ecore* files from any XML document conforming to that Schema. This way, we are able to use the XML documents as input

models in order to check the correctness of the ATL transformation generated. The whole process is depicted in Figure 7. Notice that each step is completely automatic.

The first step is to create the AUTOSAR 2.0 and AUTOSAR 2.x *.ecore* files using the EMF *xsd2ecore* importer. Then, we use EMF to generate a set of Eclipse plug-ins. Once those plug-ins are generated, any XML document conforming to the AUTOSAR 2.0 XML Schema is recognized by EMF as a model conforming to the AUTOSAR 2.0 metamodel (the same with AUTOSAR 2.x). Those plug-ins also provide with the JAVA code that allows manipulating those models and a visual editor for them. Finally, since ATL uses the EMF registry, it can use those models as inputs to test the ATL program obtained and generate an AUTOSAR 2.x model from an AUTOSAR 2.0 model.

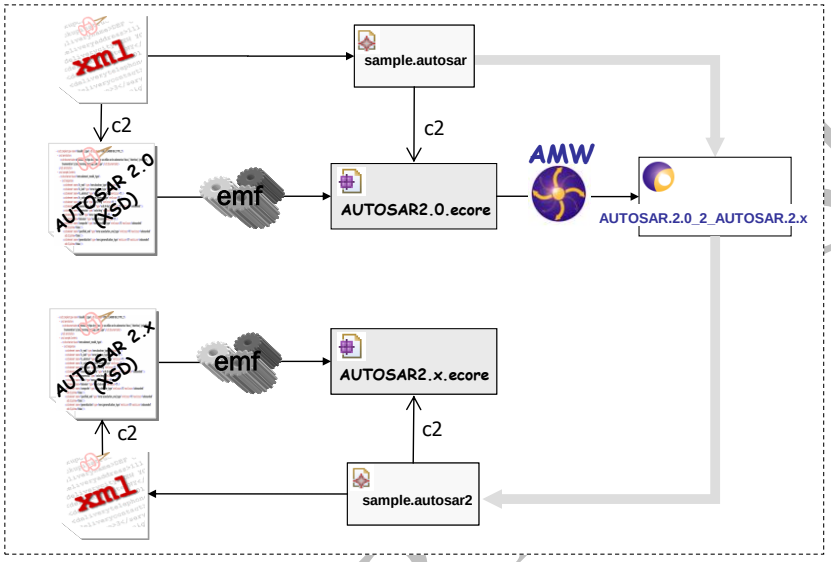

Figure 7. Automatic Generation of Models and Metamodels with EMF

6. Conclusion

In this article we have used some of the functionalities provided by the Eclipse Modeling Project (EMP) to solve a real and complex problem: the migration of software artifacts. We have defined an automatic process to solve this problem. Due to the complexity of the software artifacts considered, the level of automation of our solution was important. We have translated the common problem of software migration into a model engineering problem, and then we have used a set of model engineering tools to develop an automatic solution. More specifically, we have used the ATLAS Model Weaver plug-in to execute a set of heuristics to establish relationships between the elements of two versions of a given standard. Then, we have translated these relationships into an ATL program that performs the migration between the two versions of the standard. Furthermore, we have taken advantage of the EMF capabilities to help in the automation of the whole process.

Finally, the approach presented here can be easily generalized to other domains and contexts. The key of the solution proposed is the accuracy of the relationships found between the elements of the two different versions of the standard. The accuracy of these relationships depends directly on the quality of the executed heuristics. This means it is very important to choose the more adapted heuristics for each specific context. So, new heuristics shall be considered (or the ones used here can be modified) in order to improve the result of the matching process for each given situation.

To the best of our knowledge, the technique presented here is original. The proof of concept has been achieved and the results are available as open source. The approach seems ready for practical deployment.

We may qualify this technique of simple or complex according to the point of view. In comparison of the huge economical challenge, the solution may be considered as quite simple as has been discussed in the paper. This is mainly due to the simplicity, the genericity and power of the AMW tool and its good coupling with the ATL model transformation solution. However there is still much room for improvement. The ATLAS group is currently looking for collaborations in this field to improve the tools and processes that will make software migration through successive versions an easily manageable problem.

7. Acknowledgement

This work has been partially supported by the ModelPlex European Integrated project, as well as the GOLD project financed by the Spanish Ministry of Education and Science (TIN2005-00010) and the

FoMDAs project (URJC-CM-2006-CET-0387) cofinanced by the Rey Juan Carlos university and the Regional Government of Madrid.

8. References

[1] Bernstein, P A. *Applying Model Management to Classical Meta Data Problems*. First Biennial Conference on Innovative Data Systems Research, Asilomar, CA, USA. 2003.

[2] Bézivin, J. *On the unification power of models*. Journal of Software and Systems Modeling, Vol. 4(2): p. 171-188. 2005.

[3] Budinsky, F. et al. *Eclipse Modeling Framework*. Addison-Wesley Professional, 2004.

[4] Didonet Del Fabro, M., Bézivin, J. and Valduriez P. (2006). *Weaving Models with the Eclipse AMW plugin*. Eclipse Modeling Symposium, Eclipse Summit Europe, Esslingen, Germany. 2006.

[5] Didonet Del Fabro, M. and Valduriez, P. *Semi-automatic Model Integration using Matching Transformations and Weaving Models*. The 22nd Annual ACM SAC, MT 2007 - Model Transformation Track, Seoul (Korea), 2007.

[6] Eclipse open development platform. http://www.eclipse.org

[7] Eclipse Modeling Project. http://www.eclipse.org/modeling/

[8] Eiseman, U, and Beine, M. *Transforming function models into AUTOSAR software components*. Embedded Control Europe Magazine. February 2007. pp. 24-28. 2007.

[9] Jouault, F. and Kurtev, I. *Transforming Models with ATL*. Model Transformations in Practice (Workshop at MoDELS Conference), Montego Bay, Jamaica. 2005.

[10] OMG, *XML Metadata Interchange (XMI)*, OMG Document - formal/02-01-01 http://www.omg.org/cgi-bin/doc?formal/2002-01-01.

[11] Pelz, G., Oehler, P., Fourgeau, E, Grimm, C. *Automotive System Design and Autosar*. Advances in Design and Specification Languages for SoCs, pp. 293-305. Springer US, 2005.

[12] Tratt, L. *Model transformations and tool integration*. Software and Systems Modeling, Volume 4, Issue 2, Pages 112 – 122. 2005.

[13] Schmidt, D.C. *Guest Editor's Introduction: Model-Driven Engineering*. Computer, vol. 39, no. 2, pp. 25-31, 2006.

9. Glossary

ATL: ATLAS Transformation Language

AMW: ATLAS Model Weaver

EMF: Eclipse Modeling Framework

MDE: Model Driven Engineering

XMI: XML Metadata Interchange