

Software Modernization through Model Transformation

Mustafa ARIKAN

Maulbertschgasse 7 1190 Vienna AUSTRIA
mustafa.arikan@arikan.at

Xiaoxia LIN

Maulbertschgasse 7 1190 Vienna AUSTRIA
xiaoxia.lin@arikan.at

SUMMARY

Legacy Modernization and Software Documentation are two crucial activities in Software Development. Among many other factors, emerging technologies, changing laws, pressure to deliver software in an agile way at lower costs, force organizations to take precautions for an optimal utilization of their IT assets. ModelCVS*) is an awarded model transformation technology, which among others enables organizations to document and modernize their software. This paper describes the technological corner marks of ModelCVS – ProgGen, which is a model transformation framework [5] embedded in eclipse modeling framework.

*) This work has been awarded and partly funded by the Austrian Federal Ministry of Transport, Innovation and Technology (BMVIT) under grant FIT-IT-810806[2].

Keywords

Semantic systems, Reverse Engineering, Tool Integration, Legacy Renewal, Re-Platforming.. Model Transformation, EMF (Eclipse Modeling Framework).

INTRODUCTION

The shift from code-centric to model-centric development has placed models as first class citizens in model engineering terminology. The lack of interoperability and missing standards make the usage of tools in combination for software development an error-prone and cumbersome task.

In collaboration with two Austrian Universities and the Austrian Ministry of Defence, ARIKAN Productivity Group – technology partner of Computer Associates [1] and IT technology provider for over two decades – has developed a technology called ModelCVS[2]. This technology enables tool integration through transparent transformation of models between metamodels representing the modelling languages of different tools.

A direct integration of (modelling languages) by their metamodels is not a trivial task. It leads to solutions which mostly must be post-processed manually. In the ideal case the integration of tools with each other is an automatic process with at most a few corrections by the model manager.

ModelCVS by APG is a framework for a semi automatic generation of transformation programs, which

transparently transform models between different modelling tools.

This paper deals with one of the most important outcomes of our efforts in model transformation (tool integration) in the software business namely with ProgGen – ModelCVS by APG for Legacy Renewal and Software Documentation.

LEGACY RENEWAL

Why modernize an IT system

For a software system there are many reasons for retirement.

Sometimes it is the level of technology, which develops faster and overtakes the capabilities of an IT system, such that programs can not satisfy any more their organization's needs to remain competitive, since they don't support state-of-the-art technology .

In some other cases the missing quality of IT maintenance because of retired employees and steady grooving systems with programs of poor quality.

Sometimes it is the software architecture or the paradigm, which plays an important role in the decisions of IT managements. Procedural programming languages like COBOL, PL/1 and FORTRAN are now older programming languages, which are being replaced by more modern ones day by day.

The fact that its modelling interface is not contemporary is mostly a reason not to use a programming language, since the modelling interface has a direct impact to its usability and efficiency.

The agility of the software provider against changes and whether he would be able to deliver needed changes on time (time-to-market) or not is a crucial fact.

Finally the re-platforming of systems for cost reduction leads to migrations and language transformations.

Fundamentals of Model Transformation

The migration of an IT infrastructure to another one is a complicated task. A model is an abstract representation of a system. A model in science is a physical, mathematical, or logical representation of a system of entities, phenomena, or processes. Basically a model is a

simplified abstract view of the complex reality. The rules and constraints to which the elements of a model must obey make up the metamodel of that model. A programming language or the whole IT system can be described in terms of their metamodels. The model transformation needs beyond the metamodels (M2) a common meta-metamodel (M3) of the participating models. The same meta-metamodel makes a transformation possible. If the source and target models aren't of the same kind (of the same M3), the transformation would not be easy. For example in ECLIPSE[3] the compatibility of the metamodels with ECORE is a key issue for transformations.

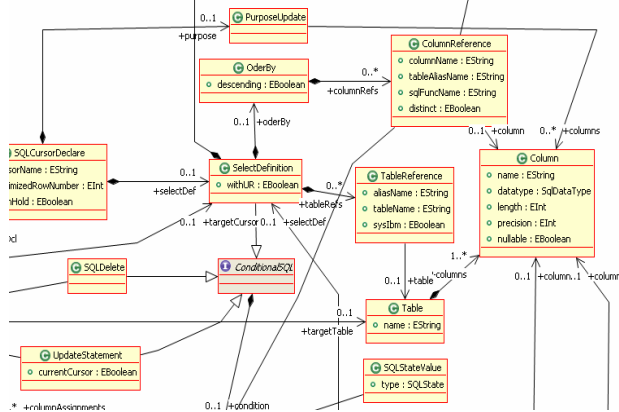


Figure1 – Metamodel PL/1 Excerpt

The metamodels of the systems which participate in a transformation process are not always explicitly available. There are some techniques with which the metamodels can be captured. The Backus-Naur form of programming languages are widely used to create parsers. From the grammar of a language its metamodel can be generated. Another technique is the usage of a metamodeling toolkit with which the behaviours of systems can be traced and metamodels and adaptors can be generated..

State-of-the-art Modernization

In OMG's terminology systems modernization [4] relies on an architecture driven transformation. Basically there are three types of architectures. These are technical architecture(T), application data architecture(A) and business architecture(B). Technical architecture describes the technical infrastructure of the system. The application and data architectures give a blue print for data persistence and the programs. Business architecture consists of the process flows and describes how the processes are build. The modernization process is made up of some transformations which must be formal. Formal transformation means that the rules are mathematically well defined. Otherwise the generation of the target language is not possible.

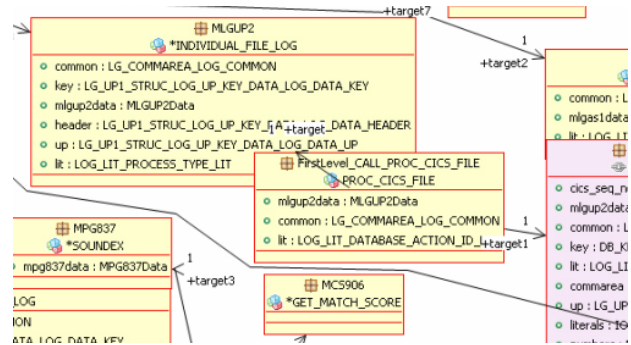


Figure 2 – Program Objects in ProgGen

The first step is the parsing of the source language. The second step is the abstraction (generalization), which is followed by the third step - enrichment. The method is the analysis of the source language and the construction of an instance of a so called Knowledge Discovery Metamodel (KDM). The KDM stores every necessary piece of information about the system. Once stored, further transformations lead step by step towards the target system. Here the knowledge of a systems specialist invaluable, because the abstracting transformation and the following enhancing transformation to build the target system require their know-how.

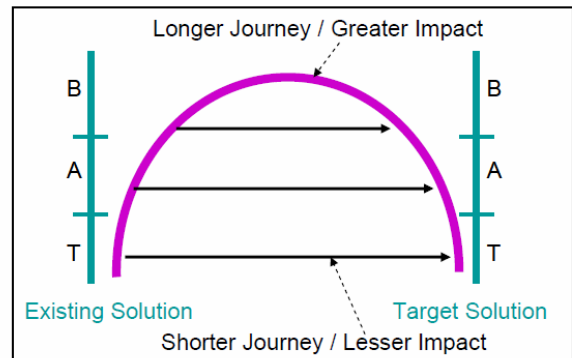


Figure 3 – Horse Shoe from OMG [4]

The transformation of a system from a source to a target language is a kind of rejuvenation. It is a decomposition of all 'cells' of a system and their rebuild under a new context. It is a semantic transformation. The system must be aware of the meanings of the concepts which are part of the system to be transformed. The language in which the source system is written must be understood by the transformation framework. Otherwise a correct transformation is not possible. The transformation can also be done into the same system as the source. Some restructuring of the same system can be targeted. For example an adaptation for SOA (service oriented architecture) can be achieved. This requires the separation of the business logic from the other program parts like presentation logic.

Documentation, Restructuring, Generation

Under code restructuring we mean changes in the code basis of a software system, such that in most cases the functionality of the software doesn't change but moving (relocating) code parts to different modules and packaging them in different packages and if necessary replacing some parts with more efficient code, the efficiency, effectiveness and the usability in new platforms rise.

Reverse software engineering is described as the process to find the hidden or difficult information about a software system. Reverse Engineering of software aims to regain a state of the related software, which can be used to understand the logic and the rules behind the reverse engineered software.

Pre-Processing

The reverse engineering may consist of one or many stages. The first stage is the so called pre-processing. This stage may not be done for every source system. Most of the time the organization, which owns the reverse engineered software has its development strategy and depending to this strategy some sort of pre-processing would be relevant. If we take some PL/1 or COBOL sources and normalize those programs under consideration of some rules of the organization and modularize in order to support the following Reverse engineering steps.

In this stage the code parts of four different categories (program source, copybooks, table-Declgens and screen maps for PL/1 and COBOL) are inserted into a relational database and the program source is divided into modules –into the so called logical units.. The utilization of a relational database instead of the flat file structure is because of the programmatic accessibility of items of interest. In COBOL, we made modules from each SECTION in the Procedure Division and in PL/1 from each PROCEDURE. One advantage of this modularization is to give a preliminary feeling about the software construction and decomposition At this stage also some standardization is done such that the code looks optically easier to understand. After the pre-processing stage the code can be investigated and the first ideas about the content can be gained. So the modules can be bound together to packages. Even some code changes can be made, since the pre-processed code is clear and lucid. Changes made in this stage require text editors.

Analyzer

The pre-processed software is easier to understand and a final mining and understanding of the source code is easy to accomplish. We have to understand the whole

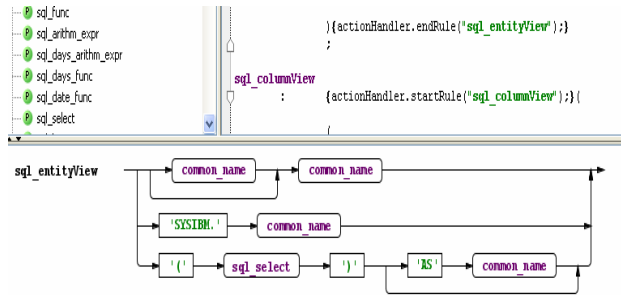


Figure 4 – ProgGen Grammar Preparation

source code syntactically and semantically. The software is written with a programming language like for example COBOL or PL/1. In order to understand the meaning of the code basis. We have to analyse it. Analysis requires parsing of the programming language. In computer science and linguistics parsing, or more formally, syntactic analysis, is the process of analysing a sequence of tokens (for example, words) to determine their grammatical structure with respect to a given (more or less) formal grammar. It means that we have to construct an analyser which is able to analyse the program syntactically. We will describe the construction of the parser and the formal analysis of the language in another paper. Figure 4 depicts such a parser generator. Such parsers can be generated from the Backus Naur Form of a programming language or can be bought in the market. In computer science, Extended Backus-Naur Form (EBNF) is a meta syntax notation used to express context-free grammars: that is a formal way to describe computer programming languages and other formal languages.

Semantic analysis is the phase in which the RE Tool adds semantic information to the parsed tree and builds the metamodel instance. This phase performs semantic checks such as type checking (checking for type errors), or object binding (associating variable and function references with their definitions), or definite assignment requiring all local variables to be initialized before use, rejecting incorrect programs or issuing warnings. Semantic analysis usually requires a complete parse tree, meaning that this phase logically follows the parsing phase, and logically precedes the code generation phase, though it is often possible to fold multiple phases into one pass over the code in implementation.

Now semantic analysis is completed. This means that we populated the meta model instance and now have the ability to use the information we have in the metamodel instance programmatically. The question we have to answer is: Which platform would be suitable for the further work. Our further work is basically about model

operations - Model search, model migration, model transformation and other model operations are necessary to finish the reverse engineering and the software restructuring. Therefore we decided to use Eclipse. EMF offers many components which can be used to fulfil our purpose. The seamless exchange of models among the environments in which we have the pre-processed sources and the analysed meta model instances is crucial for the further manipulation of the model.

ProgGen@ECLIPSE

The EMF project is a modeling framework and code generation facility for building tools and other applications based on a structured data model. From a model specification described in XMI, EMF provides tools and runtime support to produce a set of Java classes for the model, along with a set of adapter classes that enable viewing and command-based editing of the model, and a basic editor. The parsed meta model instance can be opened in Eclipse. The PL/1 view shows the XMI Structure as PL/1 Code.

Besides that, the Ecore generated graphics of the meta model instance can be opened in the graphical editor. In this editor we have many types of model manipulation functions available. Initially the classes correspond to the built logical units of the program and the links correspond to the program calls. The attributes which are used are shown on each module and can be hidden. Defined functions allow that the modules can be copied together into a new module or content of one module can be divided into two or more modules. The next step is the conversion. In this case the PL/1 instance is converted to the CA Gen instance. Finally the converted metamodel instance will be persisted under the target format. This is text format for PL/1 and COBOL and for some other target environments the target may be in binary format.

CONCLUSION

ModelCVS project and its outcomes delivered mature reverse engineering and model transformation techniques. Those techniques can be used among other use cases for system modernization (documentation). ProgGen is the software product in which this technology is implemented.

SOURCES

- [1] Technology Partnership CA
www.ca.com/us/products/collateral.aspx?cid=214266
- [2] ModelCVS www.modelcvs.com
- FIT-IT <http://www.big.tuwien.ac.at/projects/3742.html>
- [3] ECLIPSE <http://www.eclipse.org/modeling/>
- [4] OMG <http://www.omg.org/docs/admtf/07-12-01.pdf>

[5] Lin ModelCVS

http://www.arikan.at/apg_olb/template.do?action=spektrum.forschung.or

BIOGRAPHIES

Mustafa ARIKAN

Mustafa ARIKAN finished his high school education at Istanbul Erkek Lisesi in Istanbul. He started his university Education at Middle East Technical University in Ankara and changed after one year of education in METU to Bosphorus University in Istanbul and finished it in 1983 with an BSc degree in Industrial Engineering. 1983 -1986 He went



to Vienna and studied comprehensive courses from mathematics and computer science at Vienna university of technology. In the period 1982 -2009 Mustafa Arikan worked for major Software Vendors like IBM and Computer Associates. In 1988 he started his own company and served international many institutions in more than 50 large scale IT projects. He won with his company many awards like Excellency in programming from IBM, FIT-IT award from the the Austrian Federal Ministry of Transport, Innovation and Technology. His company is a technology partner of the American software vendor Computer Associates.

Xiaoxia LIN

Xiaoxia LIN started her university education with a medicine study in the Peoples Republic of China. In 1988 after finishing her medicine study Xiaoxia LIN came to Austria where her and Mustafa Arikan's ways crossed each other. Ms. LIN



studied in the period 1988-2004 computer science at the Vienna University of Technology and finished it with an MSc degree with distinction in Computer science. Currently Ms. Lin works at her doctoral work about Model Based Development. Ms. Lin joined ARIKAN Productivity Group in 2000. She is author of many large scale software systems from different branches and also the mastermind of ModelCVS with Mustafa Arikan.