A Tool for Interoperability Analysis of Enterprise Architecture Models using Pi-OCL

Chapter · January 2010

DOI: 10.1007/978-1-84996-257-5_8

4 authors, including:

Markus Buschle
KTH Royal Institute of Technology
22 PUBLICATIONS **540** CITATIONS
SEE PROFILE

Pontus Johnson
KTH Royal Institute of Technology
140 PUBLICATIONS **2,480** CITATIONS
SEE PROFILE

1

A Tool for Interoperability Analysis of Enterprise Architecture Models using Pi-OCL

J. Ullberg[1], U. Franke[1], M. Buschle[1] and P. Johnson[1]
[1] Industrial Information and Control Systems, KTH Royal Institute of Technology, Osquldas v. 12, SE-10044 Stockholm, Sweden

Abstract. Decision-making on enterprise-wide information system issues can be furthered by the use of models as advocated by the discipline of enterprise architecture. In order to provide decision-making support, enterprise architecture models should be amenable to analyses. This paper presents a software tool, currently under development, for interoperability analysis of enterprise architecture models. In particular, the ability to query models for structural information is the main focus of the paper. Both the tool architecture and its usage is described and exemplified.

Keywords: Enterprise Architecture, Probabilistic Relational Models, Software tool, Interoperability

1.1 Introduction

During the last decade, enterprise architecture has grown into an established approach for holistic management of information systems in an organization. Enterprise architecture is model-based, in the sense that diagrammatic descriptions of the systems and their environment constitute the core of the approach. A number of enterprise architecture initiatives have been proposed, such as The Open Group Architecture Framework (TOGAF) [1], Enterprise Architecture Planning (EAP) [2], the Zachman Framework [3], and military architectural frameworks such as DoDAF [4] and NAF [5]. What constitutes a "good" enterprise architecture model has thus far not been clearly defined. The reason is that the "goodness" of a model is not an inherent property, but contingent on the purpose the model is intended to fill, i.e. what kind of analyses it will be subjected to. For instance, if one seeks to employ an enterprise architecture model for evaluating the performance of an information system, the information required from the model differs radically from the case when the model is used to evaluate system interoperability.

Enterprise architecture analysis is the application of property assessment criteria on enterprise architecture models. For instance, one investigated property might be the information security of a system and a criterion for assessment of this property might be "If the architectural model of the enterprise features a communication link with high availability, then this yields a higher level of interoperability than if the availability is low." Criteria and properties such as these constitute the theory of the analysis and may be extracted from academic literature or from empirical measurements. This theory can be described using probabilistic relational models (PRMs) [6] which will be further detailed in section 1.3

Many interoperability concerns are related not only to various attributes of the architecture model but rather to the structure of the model, e.g. that two communicating actors need to share a language for their communication. With the actors and languages being entities in the metamodel, this property corresponds to ensuring that particular relationships exist in the architecture model instance.

This paper presents a software tool under development for the analysis of enterprise architecture models. This tool extends previous versions of the tool [7][8] and the main contribution of this article is to illustrate how to query the architecture models for structural information such as the language compatibility as described above. This extension is based on the Object Constraint Language (OCL) [9] and is further detailed in section 1.3.

The enterprise architecture frameworks described above generally suffer from the lack of methods for evaluating the architectures [10]. A number of enterprise architecture software tools are available on the market, including System Architect [11] and Aris [12]. Although some of these provide possibilities to sum costs or propagate the strategic value of a set of modeled objects, none have significant capabilities for system quality analysis in the sense of ISO-9126 [13], i.e. analysis of non-functional qualities such as reliability, maintainability, etc. However, various analysis methods and tools do exist within the software architecture community, including the Architecture Tradeoff Analysis Method (ATAM) [14] and Abd-Allah and Gacek [15]. In particular, various approaches such as LISI [16] and i-Score [17] have been suggested for assessing interoperability. However, these are generally not suitable in the enterprise architecture domain.

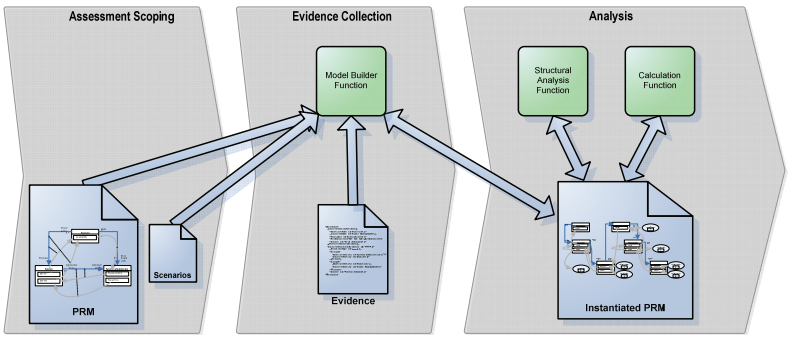

Fig 1. The process of enterprise architecture analysis with three main activities: (i) setting the goal, (ii) collecting evidence and (iii) performing the analysis.

1.2 Process of Enterprise Architecture Analysis

Enterprise architecture models have several purposes. Kurpjuweit and Winter [18] identify three distinct modeling purposes with regard to information systems, viz. (i) documentation and communication, (ii) analysis and explanation and (iii) design. The present article focuses on the analysis and explanation since this is necessary to make rational decisions about information systems. An analysis-centric process of enterprise architecture is illustrated in Figure 1 above. In the first step, assessment scoping, the problem is described in terms of one or a set of potential future scenarios of the enterprise and in terms of the assessment criteria (the PRM in the figure) to be used for scenario evaluation. In the second step, the scenarios are detailed by a process of evidence collection, resulting in a model (instantiated PRMs, in the figure) for each scenario. In the final step, analysis, quantitative values of the models' quality attributes are calculated, and the results are then visualized in the form of enterprise architecture diagrams.

More concretely, assume that a decision maker in an electric utility is contemplating the changes related to the measurement of customer electricity consumption. The introduction of automatic meter reading would improve the quality of the billing process and allow the customers to get detailed information about their consumption. The question for the decision maker is whether this change is feasible or not.

As mentioned, in the assessment scoping step, the decision maker identifies the available decision alternatives, i.e. the enterprise information system scenarios. Also in this step, the decision maker needs to decide on how to determine how this scenario should be evaluated, i.e. the assessment criteria, or the goal of the assessment. Oftentimes, several quality attributes are desirable, but we simplify the problem to the assessment of interoperability.

During the next step, to identify the best alternative, the scenarios need to be detailed to facilitate an analysis of them. Much information about the involved systems and their organizational context may be required for a good understanding of their future interoperability. For instance, it is reasonable to believe that a reliable medium for passing messages would increase the probability that communication is successful. The availability of the message passing system is thus one factor that can affect the interoperability and should therefore be recorded in the scenario model. The decision maker needs to understand what information to gather, and also ensure that this information is indeed collected and modeled.

When the decision alternatives are detailed, they can be analyzed with respect to the desirable property or properties. The pros and cons of the scenarios then need to be traded against each other in order determine which alternative should be preferred.

1.3 An Enterprise Architecture Analysis Formalism

A *probabilistic relational model (PRM)* [6] specifies a template for a probability distribution over an architecture model. The template describes the metamodel M for the architecture model, and the probabilistic dependencies between attributes of

the architecture objects. A PRM, together with an instantiated architecture model *I* of specific objects and relations, defines a probability distribution over the attributes of the objects. The probability distribution can be used to infer the values of unknown attributes. This inference can also take into account evidence on the state of observed attributes.

A PRM *Π* specifies a probability distribution over all instantiations *I* of the metamodel *M*. As a Bayesian network [6] it consists of a qualitative dependency structure, and associated quantitative parameters. The qualitative dependency structure is defined by associating attributes *A* of class *X (A.X)* with a set of parents *Pa(X.A),* where each parent is an attribute, either from the same class or another class in the metamodel related to *X* through the relationships of the metamodel. For example, the attribute *satisfied* of the class *Communication Need* may have as parent *CommunicationNeed.associatedTo.communicatesOver.isAvailable*, meaning that the probability that a certain communication need is satisfied depends on the probability that an appropriate message passing system is available. Note that a parent of an attribute may reference a set of attributes rather than a single one. In these cases, we let *X.A* depend probabilistically on an *aggregated* property over those attributes constructed using operations such and *AND, OR*, *MEAN* etc.

Considering the quantitative part of the PRM, given a set of parents for an attribute, we can define a local probability model by associating a conditional probability distribution with the attribute, *P(X.A |Pa(X.A))*. For instance, *P(CommunicationNeed.satisfied=True|MessagePassingSystem.isAvailable=False) =10%* specifies the probability that communication need is satsified, given the availability of the message passing system.

1.3.1 Pi-OCL

PRMs do not, however, provide any means to query the models for structural information such "as given two actors with a need to communicate, do these actors have a common language (modeled as a separate object)?" The Object Constraint Language (OCL) is a formal language used to describe constraints on UML models. Writing these types of constraints in natural language would lead to ambiguities. In order to resolve this, OCL provides a means to specify such constraints in a formal language without the need for extensive mathematical knowledge. OCL expressions typically specify invariant conditions that must hold for the system being modeled or queries over objects described in a model. [9]

This ability to query models would be of great benefit to interoperability analysis. OCL is, however, a side effect free language, so that the application of an OCL statement cannot change the content of the model. Furthermore it is deterministic and thus incapable of allowing uncertainty in the analysis. This section briefly describes how these two aspects of OCL are addressed by the extension to *probabilistic imperative* OCL or Pi-OCL for short. For a more comprehensive treatment see [19]. For the language to become imperative, i.e. to be able to change the state of the model, the introduction of an assignment operator is necessary. Turning OCL into a probabilistic language also requires redefinition of all operations in OCL, which is accomplished through the mapping of each operation to a Bayesian network and then combining these networks into larger

networks corresponding to a complete Pi-OCL statement. For the sake of the analysis it is also necessary to introduce an existence attribute *E* in all classes and relationships corresponding to the probability that the class or relationship exists.

Each expression views the model from the perspective of a context object *C*. The probability that an instantiated object *O* and its attributes *O.A* exist are thus not only dependent of *O.E* but also the existence of the relationships from the context object *C* to *O*.

This context dependent existence is a key factor in the definition of all other OCL operations. To provide an example of how the operations are defined the following is a description of the equality operator $= (Object2 : OclAny) \ : Boolean$. Since the original OCL equality operator is defined using the deterministic type Boolean, this operation needs to be refined. The probability that a given object o_1 equals some other object o_2, o_1=o_2 is calculated as follows:

$$P(o_1 = o_2) = \begin{cases} P[(self \ldots o_1).E, (self \ldots o_2).E] & if\ id(o_1) = id(o_2) \\ 0 & if\ id(o_1) \neq id(o_2) \end{cases}$$

Where $P[(self \ldots o_1).E, (self \ldots o_2).E]$ is the joint probability that there is a link from the context object to each of the objects o_1 and o_2 and *id(object : OclAny) : Integer* is a function that returns a unique identifier for each object. For a more comprehensive treatment of all of OCL statements in the Pi-OCL setting see [19].

The probability model of an attribute in a PRM expressed in terms of parent attributes can be replaced by Pi-OCL statements. Such statements allow not only attributes to constitute the parents of an attribute but also various aspects pertaining to the structure of the model. A PRM coupled with Pi-OCL statements thus constitutes a formal machinery for calculating the probabilities of various architecture instantiations. This allows us to infer the probability that a certain attribute assumes a specific value, given some (possibly incomplete) evidence of the rest of the architecture instantiation.

1.4 Design of the tool

Here a software tool supporting the enterprise architecture analysis process is outlined. The structural design of the tool is based upon the three basic processes illustrated in Figure 1. Relating to the previous section, the first process step concerns the identification of the decision alternatives and the goals (assessment criteria). An example of a goal is to maximize the interoperability of an IT landscape. Goals are codified in a *PRM*, which also represents how various attributes and the structure of the model affect the goals (through the probability model and Pi-OCL statements). For instance, the PRM might state that the availability of the message passing system will affect the interoperability of the scenario. The *Scenarios* are not detailed in this step, but only given a name.

In the second process step, the scenarios are elaborated. An example of a scenario might be to choose vendor X for the automatic meter reading system. In order to assess whether this is a better scenario than the one based on vendor Y,

more information is required. For instance we might need to detail the communication mediums employed by the respective vendors. Collected information on matters like these is called *Evidence*, and the process of evidence collection is supported by the tool.

From the evidence, the *Model Builder Function* creates an *instantiated PRM*, which is a comprehensive enterprise architecture scenario model. The model thus typically contains instantiations of entities such as *actor, message passing system, language,* etc. Many times, the collected evidence will not be sufficient to allow full certainty of the values of entity attributes (for instance the attribute *availability* of the entity *message passing system*). They are therefore defined as random variables, allowing the representation of uncertainty.

Recall that an important purpose of enterprise architecture models is to answer relevant questions, such as whether the interoperability of scenario A is higher than in scenario B. Oftentimes, it is not possible to directly collect information about properties such as interoperability, but it is possible to collect evidence pointing in a certain direction (e.g. information on the availability of the message passing system) and create models in which the structure of the model can be used for such evidence.

Based on the created architecture model the *structural analysis function* use the Pi-OCL statements defined in the PRM to evaluate various structural concepts of the architecture. This is done by mapping the statements to Bayesian networks, evaluating these networks and writing the result back to attributes of the architecture model. Employing Bayesian theory once again, the *Calculation Function* calculates the values of those attributes that could not be credibly determined directly. The tool employs the collected evidence, the result of the structural analysis as well as the causal relationships specified in the PRM as input to these calculations.

Central artifacts in enterprise architecture frameworks and tools are graphical models. The instantiated PRMs containing the enterprise architecture scenarios is therefore employed for displaying the results and the tool allows various views of the models.

1.4.1 Architecture of the Tool

The tool is implemented in Java and uses a Model-View-Controller architecture, thus having these three main parts, illustrated in Fig 2. The data model for PRMs and instantiated PRMs is specified in XSD files. The data binding tool Castor [20] allows simple marshalling and unmarshalling of XML models to and from java objects (the Model).

The model elements are contained by the Widgets of the Net-Beans Visual Library that also update the view when the model changes. Model operations are controlled by a number of internal tools. This essentially corresponds to the functions of the design outlined above and a set of supporting tools. These internal tools, acting as controllers, provide uniform access to their functionality as they are all implemented in a common way, using the singleton pattern.

One internal tool of particular interest here is the Pi-OCL evaluation tool that, based on the structure of the model, can perform various assessments of the

architecture. At the core of the tool is a Pi-OCL interpreter created using SableCC [21]. SableCC create an abstract syntax tree representation of Pi-OCL statements based on the grammar and provides a skeleton for traversal of the syntax tree using the visitor pattern. This skeleton is then refined to evaluate the result of the Pi-OCL statements. As this analysis is probabilistic each of the nodes in the abstract syntax tree are evaluated using Bayesian networks by employing the Smile library [22] for calculations. The results are then written back to attributes of the model.

The view is implemented as a part of the JApplication framework. The view is in charge of handling all user interactions that are not concerned with the drag-and-drop modeling features of the scene. Several methods handle button and menu actions. Also, loading and saving tools are called by the view.

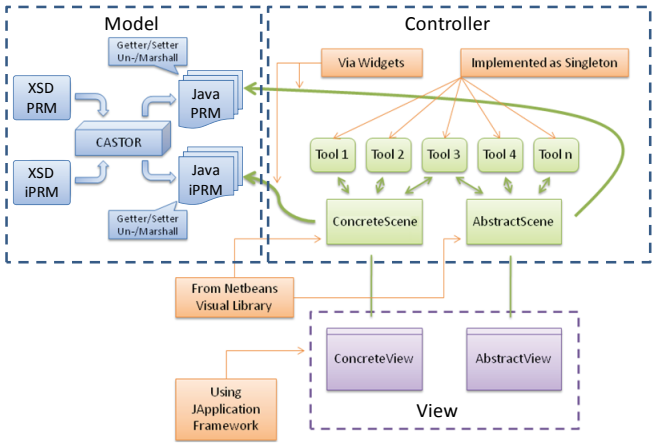

Fig 2. High level architecture of the tool, illustrating the main components. "iPRM" is shorthand notation for instantiated PRM.

1.5 Employing the tool for Interoperability Analysis

In this section a short example of the employment of the tool will be outlined. Interoperability is defined by IEEE as the ability of two or more systems or components to exchange information and use the information that has been exchanged [23]. This can be seen as a communication need and the interoperability analysis would correspond to ensuring that this need is satisfied. Consider the following metamodel for interoperability analysis where two or more *Actors* share a *Communication Need*. The actors communicate over a *message passing system*, e.g. the internet and encode the messages sent over the message passing system in a format, the *language*. On such a metamodel and interoperability definition the following very simple theory of interoperability can be expressed: 1) for interoperation to take place there need to be a path for message exchange between the *actors* of a *communication need*. 2) the *actors* must share at least one *language* for expressing their communication and finally 3) the *message passing system*

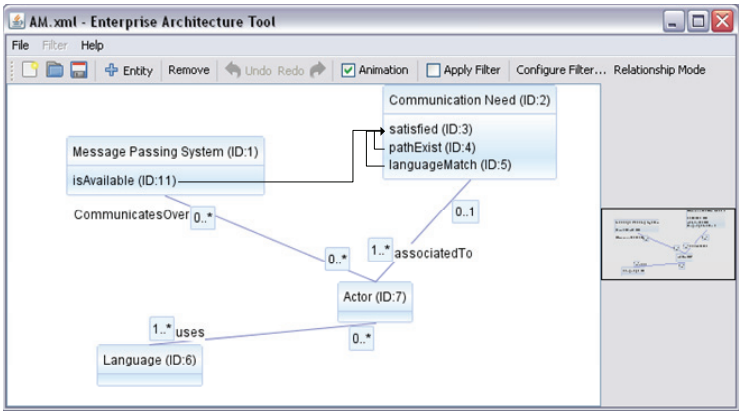

Fig 3. A sample PRM for interoperability analysis detailing the important entities and relationships needed for the analysis

these actors use must be available This theory can be expressed as a PRM using the tool, cf. Fig 3 where the abovementioned entities are modeled. The *Communication Need* class contains three attributes, the *satisfied* attribute corresponding to the goal of the analysis and two attributes corresponding to statement 1 and 2 of the theory. Since the third statement was concerned with the availability of the message passing system this is also included as an attribute of the *message passing system* class. Turning to the probability model of the PRM the first two statements in the theory are based on the structure of the architecture and can be expressed using with the following Pi-OCL statements:

Context: *CommunicationNeed*
self.*pathExist* := self.*associatedTo*->forAll(*a1 : Actor, a2 : Actor* |
a1 <> *a2* implies *a1.communicatesOver.communicatesOver*$^{-1}$ -> exists(*a2*))
self.*LanguageMatch* := self.*associatedTo*->forAll(*a1 : Actor, a2 : Actor* |
a1 <> *a2* implies *a1.uses* -> exists(*l:Language* | *a2.uses*->exists(*l*))

As can be seen from these statements the result of the Pi-OCL evaluation is assigned to two of the attributes of the class *communication need*. These attribute need to be aggregated and combined with the third statement in the theory, that pertaining to the *availability* of the *message passing system*. This can be expressed in terms of parent attributes of *CommunicationNeed.satisfied*, the goal of the analysis. The parents of this attribute thus are: *CommunicationNeed.pathExists*, *CommunicationNeed. languageMatch* and *CommunicationNeed.associatedTo. communicatesOver.isAvailable*, see the solid arrows in the PRM of Fig 3 . Turning to the quantitative part of the probability model the theory states that all these three criteria must be met for the communication need to be satisfied, an AND operation can thus be employed for aggregation.

Fig 4 depicts the concrete modeler where the architectural model is created based on the PRM. This means that specific instances of the abstract concepts are created to reflect the scenario at hand. By entering evidence on the states of the

attributes and using the structural analysis function it is possible to use PRM inference rules to update the model to reflect the impact of the architecture and the evidence. The user can thus find updated probability distributions for all attributes in the model.

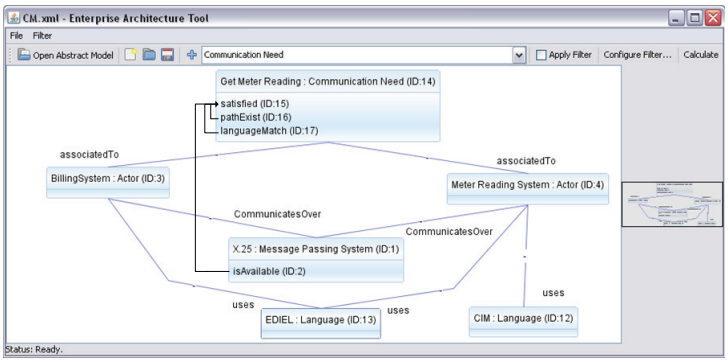

Fig 4. An instatiated PRM for the modeled scenario.

In this example we have a *Communication Need get meter reading* between the *Actors Billing System* and *Meter Reading System*. These are connected using a *X.25* leased line of type *message passing system*. The *meter reading system* uses two *languages*, the common information model (*CIM*) and *EDIEL* whereas the *billing system* only use *EDIEL* Entering information regarding the availability, of the message passing system and running both the structural analysis and the inference will result in the prediction of a interoperability value for the modeled scenario.

1.6 Conclusions

In this paper, a tool for analysis of enterprise architecture scenarios with a particular focus on interoperability assessments is outlined. The tool, currently under development, allows specification of the assessment theory in terms of a metamodel and a probability model encoded as a PRM with Pi-OCL statements. The Pi-OCL statements are used for structural analysis of the architecture. Based on the metamodel it is possible to model the architecture scenarios and using the theory assesses the interoperability of the scenario.

Information system decision making is supported by allowing quantitative comparisons of the qualities, such as interoperability, of possible future scenarios of the enterprise information system and its context.

1.7 References

[1] The Open Group. The Open Group Architecture Framework, Version 8, 2005

[2] Spewak, S.H., Hill. S.C., *Enterprise Architecture Planning. Developing a Blueprint for Data, Applications and Technology*, John Wiley and Sons, 1992

[3] Zachman, J., A framework for information systems architecture, IBM Systems Journal, 26(3), 1987
[4] DoD Architecture Framework Working Group, "*DoD Architecture Framework*" Version 1.0, 2003
[5] NAF, "*NATO C3 Technical Architecture*", Volume 1-5, Version 7.0, Allied Data Publication 34, 2005
[6] L. Getoor, N. Friedman, D. Koller, A. Pfeffer, and B. Taskar. Probabilistic relational models. MIT Press, 2007.
[7] Johnson, P., Johansson, E., Sommestad, T. Ullberg, J., A Tool for Enterprise Architecture Analysis, Proceedings of the 11th IEEE International Enterprise Computing Conference (EDOC 2007), 2007
[8] Ekstedt, M. et al. A Tool for Enterprise Architecture Analysis of Maintainability, Proc. 13th European Conference on Software Maintenance and Reengineering, 2009
[9] Object Management Group Object Constraint Language specification, version 2.0 formal/06-05-01, 2006
[10] Chen, D. et al., Architecture for enterprise integration and interoperability: Past, present and future, Comput Industry (ind), 2008
[11] IBM, *Rational System Architect*, http://www-01.ibm.com/software/awdtools/systemarchitect/, accessed: Dec 2009
[12] Scheer, A.-W., *Business Process Engineering: Reference Models for Industrial Enterprises*, 2nd ed., Springer-Verlag, 1994
[13] International Organization for Standardization/International Electrotechnical Commission, "International Standard ISO/IEC 9126-1: Software engineering – Product quality – Part 1: Quality model," ISO/IEC, Tech. Rep., 2001.
[14] Clements, P., R. Kazman, M. Klein, "*Evaluating Software Architectures: Methods and Case Studies*", Addison-Wesley, 2001
[15] Gacek, C., "*Detecting Architectural Mismatch During System Composition*", PhD. Thesis, University of Southern California, 1998
[16] Kasunic, M., Anderson, W., "Measuring Systems Interoperability: Challenges and Opportunities" *Technical Note, CMU/SEI-2004-TN-003*, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, 2004.
[17] Ford, T., Colombi, J., Graham, S., Jacques, D., "The Interoperability Score", *Proceedings of the Fifth Annual Conference on Systems Engineering Research,* Hoboken, NJ, 2007
[18] Kurpjuweit, S., Winter, R.: Viewpoint-based Meta Model Engineering. In Proceedings of Enterprise Modelling and Information Systems Architectures (EMISA 2007), Bonn, Gesellschaft für Informatik, Köllen, pp. 143-161, 2007
[19] Franke U. et. al, π-OCL: A Language for Probabilistic Inference in Relational Models (2010) to be submitted
[20] ExoLab Group, The Castor Project, http://www.castor.org/ accessed Dec. 2009
[21] Gagnon, É., SableCC, an Object-Oriented Compiler Framework, Master Thesis, McGill University Montreal, 1998
[22] Decision Systems Laboratory of the University of Pittsburgh, SMILE, http://genie.sis.pitt.edu/ accessed Dec. 2009
[23] IEEE, *Standard Glossary of Software Engineering Terminology. Std 610.12-1990*, The Institute of Electrical and Electronics Engineers, New York, 1990.