European Office of Aerospace Research and Development (EOARD)
Grant/Cooperative Agreement Award no. FA8655-11-1-3055
****

Consistent evolution of software artifacts and non-functional models

Cortellessa V., Di Marco A., Di Ruscio D., Muccini H., Pierantonio A.

Dipartimento di Informatica, Università degli Studi dell'Aquila, Via Vetoio 67100 L'Aquila, Italy
Email: *vittorio.cortellessa@univaq.it, antinisca.dimarco@univaq.it, davide.diruscio@univaq.it,henry.muccini@univaq.it, alfonso.pierantonio@univaq.it,*
Web: *http://www.di.univaq.it/cortelle/, http://www.di.univaq.it/adimarco/, http://www.di.univaq.it/diruscio/, http://www.henrymuccini.com/, , http://www.di.univaq.it/alfonso/*
Phone: +390862433165 , Fax: +390862433131

Abstract

The problem of keeping aligned software artifacts and non-functional models is nowaday ever more stringent, also due to the complexity of the maintenance phase of software systems. In fact today the software, after deployment, tends to evolve due to: (i) changes in the context/environment where it runs, (ii) evolution of user requirements. Hence, there is an increasing need for techniques that introduce automation in the management of development artifacts that have to be consistent to each other. In this project we intend to tackle such problem with model-driven advanced techniques that allow to represent the changes carried out on non-functional models and back propagate them onto the software artifacts. Our proposed approach will allow to keep aligned software artifacts and non-functional models basically without manual effort (and without needing any specific expertise) from software developers.

Keywords: model-driven engineering, non-functional analysis of software, change propagation.

Methodological description

The potential of Model-Driven Engineering (MDE) techniques in software development has been largely demonstrated in the last decade in many application domains [1]. Beyond the enhanced capabilities of satisfying (functional and non-functional) requirements through validation in the early development stages, software produced with model-based processes is obviously more maintainable than the one produced with other approaches. In fact, models built during the development process, if not trashed after deployment, can support maintenance/evolution actions that, ever more, are required to tackle changes of requirements, of the environment/context, etc.

Whereas in the functional world MDE techniques are typically aimed at automating code generation, model checking and model-based testing, in the non-functional world a typical goal of MDE is to enable model transformations that can generate non-functional models (e.g. for reliability, performance, etc.) conforming to the original design model [2]. In fact, the automated generation of non-functional models undertakes the most common obstacles to the adoption of non-functional validation of software, that are: (i) the extra-time needed to build by scratch a non-functional model, and (ii) the special skills needed to accurately accomplish this task (i.e. to build by scratch a non-functional model that conforms to the design model).

However, the potential of MDE techniques is far from being fully exploited for sake of non-functional analysis. In Figure 1 a typical process for non-functional analysis of a (set of) software artifact(s) is schematically illustrated. On the left-hand side of the figure a generic development process takes place, which produces (at a certain time) a set of software artifacts that can span from UML models and simulation artifacts to code. In order to perform a certain type of non-functional analysis (e.g performance, reliability, etc.), these artifacts have to be opportunely annotated with missing information, such as operational profile parameters. Model transformations can eb applied to annotated artifacts to generate non-functional models, such as Queueing Networks, Petri nets, Bayesian Belief Networks, etc.

The arrow from a software artifact to a non-functional model enters what in the figure is called *Analysis Loop*. In this loop the non-functional model is first solved and then, from the observation of the results as well as other useful information (like non-functional requirements and legacy constraints), it is modified in order to improve the results and satisfy the software requirements. The solid arrow from non-functional model and analysis results represents the step of model solution that is usually supported by well-assessed tools [3]. Dashed arrows in the

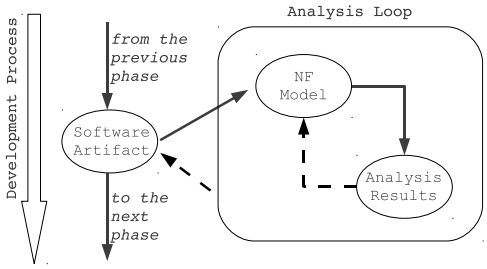

Figure 1: Non-functional analysis process today.

figure represent steps that are not supported by automation and are typically left to the experience of analyzers. In particular, the arrow from analysis results to non-functional model represents the feedback provided by the results in terms of modifications that should be made on the model in order to overcome possible problems emerged from the analysis. This is a typical "informal" step, where several tentatives are made on the model to fix the emerged problems. What happens, at the best, in practice is that such loop is executed many times without taking trace of the tentative modifications brought on the non-functional model. This is the main reason that makes the external arrow, from the analysis loop back to the software artifact, as another dashed arrow. In fact, the knowledge accumulated during the analysis loop is usually lost and only vague and/or rough feedback can be brought back to the software developers in order to improve their design.

This rough *back-propagation* of results on the development process does not help designers to take decisions on the basis of non-functional analysis, because such results are often expressed in vocabularies very different from the languages used for sake of development. For example, typical performance analysis can be conducted on Queueing Network models that, very often, produce (average values or probability distribution functions of) response time, throughput and utilization. These notations and results are very distant even, for example, from UML diagrams (even when annotates with the MARTE profile [4]).

The goal of this project is to exploit advanced MDE techniques in order to make straightforwardly exploitable the knowledge accumulated in the *Analysis Loop* of Figure 1. We intend to modify the non-functional process as illustrated in Figure 2.

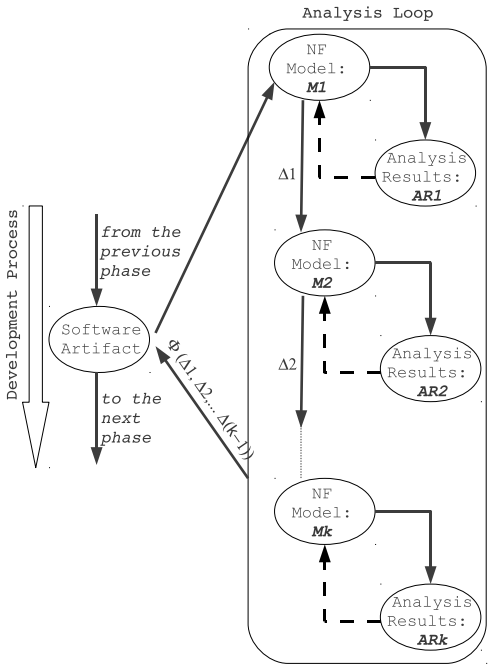

Figure 2: Non-functional analysis process devised in this project.

We assume to unfold the *Analysis Loop* and to observe changes made on the non-functional model within the loop. Some changes bring improvements, whereas others do not. However, at the end of the loop, the latest non-functional model (i.e. the one that possibily satisfies the software requirements) will be obtained by applying a combination of changes $\Delta_1, \Delta_2, ...\Delta_{k-1}$ operated within the loop starting from the initially generated non-functional model.

Basing on MDE techniques, we intend to formally specify these changes in an approriate language [5] and, through change propagation [6], to automate the propagation of these changes on the development side. This would allow the consistent evolution of the software artifacts and the non-functional models, thus contributing to fill the gap between software development and non-functional analysis.

Technical details

In order to support the process shown in Figure 2 there are at least two main issues which have to be tackled:

a Calculation and representation of NF model difference;

b Propagation of changes from NF models to software artifacts and vice-versa.

The problem of determining differences of NF models is intrinsically complex. In general it includes at least the following phases [7]:

- *Calculation*, a procedure, method or algorithm able to compare two distinct models. According to different analysis available in literature [8], there is no single best solution to model difference calculation but instead the problem should be treated by deciding on the best trade-off within the constraints imposed by the context, and for the particular task at sake;
- *Representation*, the outcome of the calculation must be represented in some form which is amenable to further manipulations. Finding a suitable representation for model differences is crucial for its exploitation similarly to what happens, for example, in deriving refactoring operations from a delta document describing how a database schema evolved in time [5].

In order to put in practice the process shown in Figure 2, the calculation and representation phases have to be properly supported since the delta models must represent accurate information and must be subject to automatic manipulations. According to previous experiences of the authors, calculating differences of NF models can be performed by using EMF Compare [7] that provides a generic matching algorithm. However, such algorithm can be customized to best fit the particular problem at sake. In this respect, the Epsilon Comparison Language (ECL)[1] can be adopted to ease the customization of the generic matching algorithm provided by EMF Compare. This is relevant since a generic matching algorithm is agnostic of underlying NF model semantics that, instead, can be considered by ECL to formalize heuristics and provide more accurate results.

We intend to employ JTL [6], a bidirectional model transformation language already used by the authors in different applicative domains (e.g. [9]), to specify how changes on NF models induce transformations of the corresponding software artifacts. JTL does not require the transformations between software and NF models to be bijective, injective or surjective. This has relevant implications when it is not possible to reverse the transformation from a modified target model to one or more source models, because in these cases the JTL engine is however able to deduce a collection of source models that approximate the ideal one, and from which it is possible to generate the previously modified target.

Statement of work

This project proposal covers 3 years. With reference to the above methodological and technical contents, the project will be broken down in the following 1-year phases. YEAR 1 has been funded with 63,000$, and it goes from 1 October 2011 through 30 September 2012.

YEAR 1 -Formalizing Δ steps. We intend to primarily focus on a specific type of analysis, that is the one where we have more experience: software performance analysis[2]. In particular, we expect to achieve the following progressive results in the first year:

- Implementing a transformation from UML models to Queueing Networks (QN) with the JTL language;
- Defining a set of elementary changes that can be operated on QN performance models within the Analysis Loop;
- Representing within JTL such changes and enabling their combination to produce Δ steps;

[1] http://www.eclipse.org/gmt/epsilon/doc/ecl/

[2] We have introduced several transformations from software artifacts to performance models in the last few years, either for general purpose software [10] or, more recently, for context-aware software applications [11]. Hence we intend to consider one of this transformation approaches and work on it to experiment the methodological approach described above.

YEAR 2 - Sequencing and automating Δ steps. The second year will be dedicated to the sequencing of Δ steps in order to build a $\Phi(\Delta_1, \Delta_2, ...\Delta_{k-1})$ transformation, as shown in Figure 2. In particular, we expect to achieve in this year the following progressive results:

- Enabling (more or less complex) sequencing of Δ steps;
- Designing an automated process that allows to back propagate Φ's onto software artifacts.

YEAR 3 - Full validation of the approach. The automation designed above shall be implemented as a prototype tool in this third year. The approach implementation would allow to validate it on in-house (and possibly real world) case studies. The validation task will highlight the scope and limitations of such approach.

Of course, in a more long-term view such approach shall be ported to other transformations in the performance domain and, later on, to transformation in different non-functional domains, such as software reliability assessment.

Biographies and Facility description

SEAlab at University of L'Aquila.
The Software Engineering and Architecture laboratory (SEAlab) at University of L'Aquila was created at the beginning of 2002, when researchers of the Computer Science Department have put together their expertises in the software engineering domain under the leadership of Paola Inverardi. Since then our research group has been able to recruit excellent researchers and to attract very promising Ph.D. students. The record of publications and the impact factors of members demonstrate the high quality of the research conducted at SEAlab. Since 2005 we have been involved in several european research project on a regular basis, such as: PLASTIC (FP6), POPEYE (FP6), MANCOOSI (FP7), CONNECT (FP7), CHOREOS (FP7), VISION (ERC), PRESTO (Artemis JU). Beside this, we maintain regular research collaborations with italian and international (academic and industrial) partners, such as: University College London (UK), Imperial College (UK), University of York (UK), INRIA (FR), Malardalen University (SW), Technical University of Wien (AUT), University of Karlsruhe (GER), West Virginia University (USA), Carleton University (CAN), Politecnico di Milano (IT), CNR Pisa (IT), Thales, Selex Communications, Ericsson Labs, etc.

Different types of skills and experiences are required to successfully conduct this research project. In fact (as evidenced in this proposal) the project objectives can be achieved only by combining expertises in advanced model-driven techniques, functional and non-functional modeling and analysis of software. SEAlab is one of the few research groups that embeds in itself all such expertises. In particular: (i) Pierantonio, Di Ruscio (and their students) represent one of the most active research groups in the domain of Model-Driven Engineering, with particular emphasis on model differences and co-evolution; (ii) Cortellessa, Di Marco (and their students) work in the field of non-functional assessment of software since many years, and are among the founders of the community of Model-Driven Software Performance Engineering; (iii) Muccini (and his students) has been very active in the field of Software Architecture and, in particular, Architectural Description Languages.

Therefore, we retain that the combination of such expertises, based on a collaboration that lasts in the same group since many years, is the key element that makes SEAlab particularly suited for this type of research, as witnessed also by the recent publications that have been produced from this group of people.

Vittorio Cortellessa's short bio.
Vittorio Cortellessa holds an Associate Professorship at the Department of Computer Science, Universit dell'Aquila, since 2005. His research interests are mostly in model-based non-functional verification and validation of complex software systems. He has published about 60 papers in international conferences and journals dealing with these topics. He was Program Chair of ACM Workshop on Software and Performance (WOSP) 2007 and of European Performance Engineering Workshop (EPEW) 2010, and he was General Chair of ACM International Conference on Performance Engineering (ICPE) 2011. He is also member of the ICPE Steering Committee since 2005. He has been PC member of several conferences in the domain of software engineering, among which: ICSE 2009, 2011 and 2012, ASE 2006, 2009, 2010 and 2011, and MODELS 2010.

Antinisca Di Marco's short bio.
Assistant Professor at the University of L'Aquila since March 2008. Previously she has worked as Research Fellow at the University College London, U.K., and she has collaborated as Post-Doc with the Dipartimento di Informatica, Sistemi e Produzione and with Dipartimento di Dipartimento di Ingegneria Elettronica of the Universita' di Roma "TorVergata". Her main research interests include (early) verification and validation of QoS, performance models and stochastic theory, QoS analysis of autonomic services and context-aware mobile software systems to support software adaptation, bio-inspired adaptation mechanisms. She published around 40 papers in international conferences and journals dealing with these topics. She has been in the Program Committee of QoSA 2009-2011, EWDC

2011, Special Session on Quality and Service-Oriented Applications of Euromicro SEAA Conference 2008-2009-2010, INTENSIVE 2009, ICSEA 2009-2010, Valid 2009-2010. She was one of the organizer of the first ARAMIS 2008 workshop and she serves as reviewer for several journals on her research topics. She has been member of several national (Progetto MIUR co-finaziato SAHARA, Progetto FIRB - Valutazione delle Prestazioni dei Sistemi Complessi: Tecniche, Metodologie e Strumenti, Italian PRIN D-ASAP) and european (IST FP6 SIMPLICITY, IST FP6 PLASTIC, Segravis) research projects. She is currently involved in the FP7 CONNECT FET and VISION ERC Starting grant (FP7 ideas project).

Davide Di Ruscio's short bio.
Davide Di Ruscio is Assistant Professor in the Computer Science Department of the University of L'Aquila. His research interests include code generation, methodologies for Web development, Model Driven Engineering, and more specifically model differencing and model evolution. He published several papers in journals and international events in these topics. He has been involved in the PC of several workshops and conferences, and reviewer of several journals like Science of Computer Programming, and Journal of Software and Systems.

Henry Muccini's short bio.
Henry Muccini received his MD degree in Computer Science summa cum laude from the University of LAquila (Italy) and his PhD degree in Computer Science from the University of Rome La Sapienza - in 2002. He is currently an assistant professor at the University of L'Aquila since 2002 and he has been visiting professor at Information and Computer Science, University of California, Irvine in 2001 and 2006. Henrys research interests are in the Computer Science area, Software Engineering field. In particular, his main research interests are on software architecture modeling and analysis, component-based systems, model-based analysis and testing, fault tolerance, and global software engineering. He has published over 60 conference and journal articles on these topics, he has co-edited two books (Software Engineering of Fault Tolerant Systems, World Scientific Publishing, June 2007 and Architecting Dependable Systems V, Lecture Notes in Computer Science 5135, August 2008).He has been co-chairing ROSATEA 2006-2007 (workshop on Software Architecture, Testing and Analysis), the EFTS series of event on Engineering Fault Tolerant Systems, AeroSE 06 (Aerospace Software Engineering), SERENE 08 (Software Engineering for Resilient Systems), and ARAMIS 08 (Workshop on Automated engineeRing of Autonomous and run-tiMe evolvIng Systems). He is in the SC of SERENE and ISARCS. He has been serving as PC member and reviewer in many international conferences and journals. His research and development collaborations are with some research institutes and companies in Italy and abroad.

Alfonso Pierantonio's short bio.
Alfonso Pierantonio is Associate Professor in Computer Science at the University of LAquila (Italy) since 2006, he is currently director of the Master in Web Technology degree program. His current research interests include Model-Driven Engineering and in particular the theory and practice of model versioning/evolution with a specific emphasis on coupled evolution. In particular, he investigated the problem of co-evolution between metamodels and other artifacts in order to define the basis for their (semi) automatic adaptation. More recently, his research focus has been devoted to bidirectional model transformations and to the definition of a new bidirectional transformation language called JTL (Janus Transformation Language). The developed techniques and systems have been largely applied in the domain of enterprise web applications and their evolution and migration. He was General Chair of the International Conference on Model Transformation (ICMT) 2008 and currently is Chair of its Steering Committee. He is also member of the Steering Committee of the Models and Evolution International Workshop. He has been PC member of several conferences, such as ASE 2009, MoDELS 2009, 2010, and 2011, WWW 2010, and ICWE 2009, 2010, and 2011, to mention a few. He co-edited several special issues on Model Transformations which appeared on Science of Computer Programming and on the Intl. Journal of Software and Systems Modeling.

References

[1] D. C. Schmidt, Guest editor's introduction: Model-driven engineering, IEEE Computer 39 (2006) 25–31.

[2] S. Balsamo, A. DiMarco, P. Inverardi, M. Simeoni, Model-based performance prediction in software development: A survey, IEEE Trans. Software Eng. 30 (2004) 295–310.

[3] K. S. Trivedi, R. A. Sahner, Sharpe at the age of twenty two, SIGMETRICS Performance Evaluation Review 36 (2009) 52–57.

[4] UML Profile for MARTE, ptc/08-06-09, Object Management Group, 2008.

[5] A. Cicchetti, D. Di Ruscio, A. Pierantonio, A Metamodel Independent Approach to Difference Representation, Journal of Object Technology 6 (2007) 165–185.

[6] A. Cicchetti, D. DiRuscio, R. Eramo, A. Pierantonio, Jtl: a bidirectional and change propagating transformation language, in: 3rd International Conference on Software Language Engineering (SLE 2010), pp. 183–202.

[7] C. Brun, A. Pierantonio, Model differences in the eclipse modelling framework, UPGRADE journal IX (2008) 29–34.

[8] D. S. Kolovos, D. DiRuscio, R. F. Paige, A. Pierantonio, Different models for model matching: An analysis of approaches to support model differencing, in: ICSE Workshop on Comparison and Versioning of Software Models, pp. 1–6.

[9] R. Eramo, I. Malavolta, H. Muccini, P. Pelliccione, A. Pierantonio, A model-driven approach to automate the propagation of changes among architecture description languages, Software and Systems Modeling (2011).

[10] V. Cortellessa, R. Mirandola, Prima-uml: a performance validation incremental methodology on early uml diagrams, Sci. Comput. Program. 44 (2002) 101–129.

[11] L. Berardinelli, V. Cortellessa, A. DiMarco, Performance modeling and analysis of context-aware mobile software systems, in: Fundamental Approaches to Software Engineering (FASE 2010), pp. 353–367.