Using NSGA-II to compose model transformations alternatives

Smail Rahmoun*†, Etienne Borde†, Laurent Pautet†,
*Institute for Technological Research SystemX - 8, Avenue de la Vauve 91120 PALAISEAU
firstname.lastname@irt-systemx.fr
†Institut Telecom;TELECOM ParisTech; LTCI - UMR 5141 - 46 rue Barrault, 75013 Paris, France
firstname.lastname@telecom-paristech.fr

***Abstract*—The size and complexity of modern systems is rapidly increasing. Meanwhile, the ability to understand and maintain such systems is decreasing almost as fast. Model Driven Engineering promotes the automation of design activities by means of model transformations, enabling to cope with systems complexity. However, the design space produced by the composition of these model transformations is difficult to explore automatically. The main reason that make this exploration difficult is the size of the design space : the number of possible design choices grows exponentially with the number of model transformation alternatives. This paper presents a MDA approach for design space exploration to deal with the ever-growing challenge of designing complex embedded systems. This approach allows the system architects to automatically select the most adequate model transformations compositions by using multi-objectives optimization evolutionary algorithms.**

I. Introduction

Nowadays, modern systems become more and more large and complex and therefore more difficult to develop and maintain. For example, embedded critical systems which are often built to accomplish specific functionalities, can integrate several buses, software and hardware components. To guarantee properties such as safety, availability and robustness makes the design of these systems very challenging.

Model Driven Engineering (MDE) proposes the use of models as the key assets in the development of such systems. Model transformations approach become one of the basic building blocks of MDE. Even though MDE is being successfully applied in many scenarios, it still needs appropriate mechanisms to handle the development of complex, large-scale systems. One such mechanism is a facility to make transformations reusable, so that we can apply them to different contexts. In this work, we use generic model transformation to model design alternatives of the system. This process will help us to predict non-functional properties (NFP) of a system before its development [2]. This prediction can be exploited to drive decisions about which components should be used and how they should be connected so as to meet the NFP imposed on the design.

Moreover, an issue in modeling a system is that, the NFP could conflict[1] with each other. For example, improving the availability of a service is often done by replication of the service, which causes the increasing of the response time of these services [20]. Developing a system that satisfies at best all its NFP simultaneously is often not possible : improving a NFP of a system can deteriorate another NFP. As a consequence, system architects have to come up with several design alternatives and select the solutions which fulfils at best all NFP [4] simultaneously from all feasible alternatives. The selected solutions are called non-dominated solutions or Pareto optimal solutions : a solution x2 is non-dominated by another solution x1, if x2 matches or exceeds x1 in all objectives (NFP).

In order to obtain the set of Pareto optimal alternatives, some advanced multi-objective optimization evolutionary algorithms (MOEA) have been proposed. These algorithms apply some fitness evaluations and generic operators of variation. Thus, they are applicable to search spaces (e.g. design space) taking into account domain specific characteristics of the problem (e.g. prediction of NFP). MOEA also provides selection operators which guide the search towards interesting regions of the search space by (i) favorizing non-dominated solutions; and (ii) finding a diverse set of solutions on the Pareto front..

In this paper, we introduce an approach, tailored for embedded system architecture design, where architecture alternatives are implemented as model transformations driven by NFPs. And to find good transformations alternatives that help system architects perform trade-off analysis, we use some advanced MOEA. More specific, we adapt (or map) model transformations mechanisms into a multiple-objective optimization problem.

The reminder of this paper is organized as follows. Section II is dedicated to the problem statement. Section IV gives an overview of our approach. While Section IV-C details our approach. Section V presents the related work.

II. Problem

During the design of a safety-critical system, it is useful to model non functional properties (NFP), such as reliability, availability or temporal performances, already in early stages of the development process. Early NFP modeling enables the developer of the system to make design decision based on analyses and simulations. But, one major difficulty is that these

[1]improving one property can have a negative impact on other properties

NFP can be in conflict with each other, that is, improving one NFP can have a negative impact on others. To construct a system that fulfils all its NFP simultaneously is often not possible. As a consequence, system architects have to consider several design alternatives, and identify a solution that fulfils most NFP from all feasible alternatives.

A possible solution to this issue is to implement these design alternatives in a way that they can be seen as design patterns. Each pattern implements a well-known architectural solution which fulfils a set of NFP. To formalize these design patterns in a way that allows their reuse in different projects or products we implement them as model transformations.

The second challenge in this work comes from the size of the design space to explore in order to study such NFP trade-offs. With a complex system and a larger design space, it is almost impossible for software architects to find optimal architecture designs. The exploration of such design space can be very difficult to do manually and error-prone. To overcome this issue, we propose an approach to assist architects throughout the modeling phase and automate the design space exploration by using advanced MOEA.

Moreover, using MOEA to deal with mode transformations alternatives can be challenging. We have (i) to generalize the architecture design problem as a multiple objective optimization problem; (ii) select the more suitable evolutionary algorithm to solve this optimization problem; and (iii) adapt the notions of model transformation techniques to this evolutionary algorithm.

III. Multi-Objective Evolutionary Algorithm

An evolutionary algorithm (EA) is characterized by three features : (i) a set of solution candidates is maintained; (ii) a mating selection process is performed on this set, and (iii) several solutions may be combined in terms of recombination to generate new solutions.

The solution candidates are called **genomes** (or individuals) and the set of solution candidates is called the **population**. Each genome is composed of a sequence of genes and represents a possible solution to the problem at hand.

The mating selection process usually consists of two steps: **fitness evaluation** and **mating pool**. In the first step, the genomes are evaluated in the objective space and then assigned a scalar value called **fitness**, reflecting their quality. Afterwards, a mating pool is created by random sampling from the population according to the fitness values. For example using a binary tournament selection. Where two genomes are randomly chosen from the population, and the one with the better fitness value is copied to the mating pool. This procedure is repeated until the mating pool is filled.

Then, the variation operators are applied to the mating pool to obtain better solutions. Usually these operators are called **genetic operators**. With EAs, there are two operators namely the crossover (or recombination) and the mutation.

Based on the above concepts, EAs are simulated by an iterative computation process. First, an initial population is created at random, which is the starting point of the evolution process. Then a loop consisting of the fitness evaluation, selection, genetic operators is executed a certain number of times. Each loop iteration is called a generation, and often a predefined maximum number of generations serves as the termination criterion of the loop.

In order to deal with the multi-objectives nature of our problem (which model transformations achieve the best possible trade-off among NFP) we use MOEA. Among the many MOEAs that have been proposed in the literature : NSGA-II [5], SPEA2 [21], IBEA [22], only one has been considered in this work, namely NSGA-II (Non-dominated Sorting Genetic Algorithm). It was proposed in the early 90's and still considered today as one of the state-of-the-art MOEA for its robustness across a variety of application domain.

A. NSGA-II

NSGA-II is an improved version of NSGA [18] (Non-dominated sorting genetic algorithm). It inherited the non-dominance concept from NSGA and showed three innovations, (i) a crowding approach for diversity preservation; (ii) an elitism operator that helps in significantly speeding up the performance of the genetic algorithm (fast non-dominated sort); and (iii) an objective-wise distance computation.

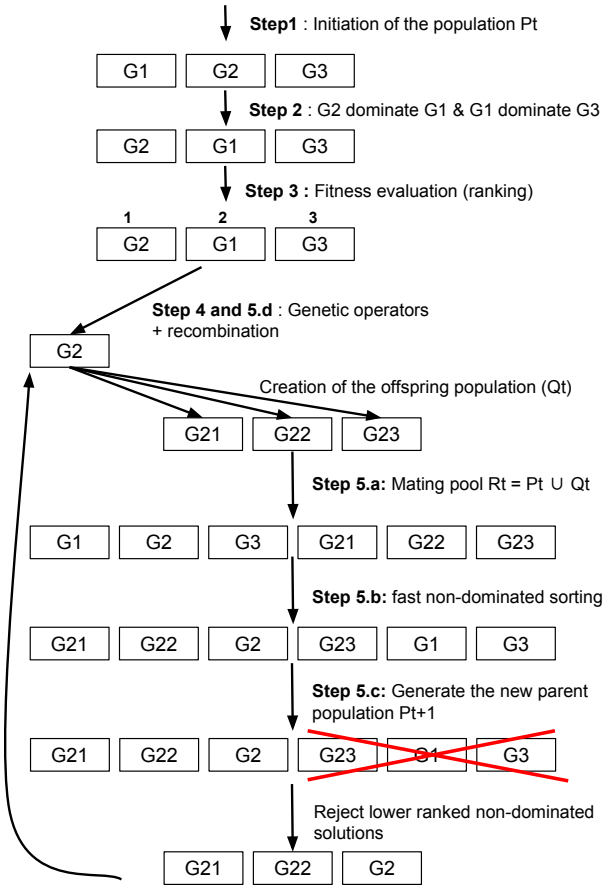

Fig. 1. Schema of the NSGA-II algorithm

Figure 1 gives an overview of how to use NSGA-II to

automate the exploration of complex systems design space. The steps of this algorithm are as follow :

1) Initially, Create a random parent population P_0 composed of N genomes (N = 3 in Figure 1). N represent the size of the population.
2) Sort the current population based on a non domination criterion (e.g. reliability, availability and response time).
3) For each non-dominated solution, assign a fitness (rank) equal to its non-domination level (1 is the best level, 2 is the next best level, and so on).
4) Create a child population Q_0 of size N using binary tournament selection, recombination, and mutation operators.
5) From the first generation onwards, creation of each new generation constitutes the following steps :

5.a) Create the mating pool R_t of size 2N by combining the parent population P_t and the child population Q_t.

5.b) Sort the combined population R_t according to the fast non-dominated sorting procedure to identify all non-dominated fronts $(F_1, F_2, ..., F_k)$.

5.c) Generate the new parent population $P_t + 1$ of size N by adding non-dominated solutions starting from the first ranked non-dominated front $F1$ and proceeding with the subsequently ranked non-dominated fronts $F_1, F_2, ..., F_k$, till the size exceeds N (Figure 1). This means that the total count of the non-dominated solutions from the fronts $F_1, F_2, ..., F_k$, exceeds the population size N. Now, in order to make the total count of the non-dominated solutions equal to N, it is required to **reject** some of the lower ranked non-dominated solutions from the last $Fkth$ front. This is achieved through a sorting process based on the crowding distance assigned to each solution contained in the F_kth non-dominated front. Thus, the new parent population P_t+1 of size N is constructed.

6) Perform the selection, crossover and mutation operations on the newly generated parent population $(P_t + 1)$ to create the new child population $(Q_t + 1))$ of size N (Figure 1).
7) Repeat Step 5 until a stopping criterion is met (e.g. maximum number of generations is reached).

IV. Using evolutionary algorithms for identifying best design alternatives

Figure 2 graphically illustrates the approach proposed in this paper by showing the process underlying it. As can be seen in Figure 2, the process starts with an initial input software architecture which could be designed by software architects by using architecture design languages [16]. After that, the input model is transformed into a set of alternative architecture models (more detailed models) driven by a set of competing NFP. To solve the trade-off between NFP we expressed the architecture alternatives as design patterns implemented by means of model transformations and compositions techniques. The next step consists at finding the best model transformations alternatives. To do so we (i) map these model transformations alternatives as a multiple-objectives optimization problem and (ii) use NSGA-II to find the non-dominated solutions.

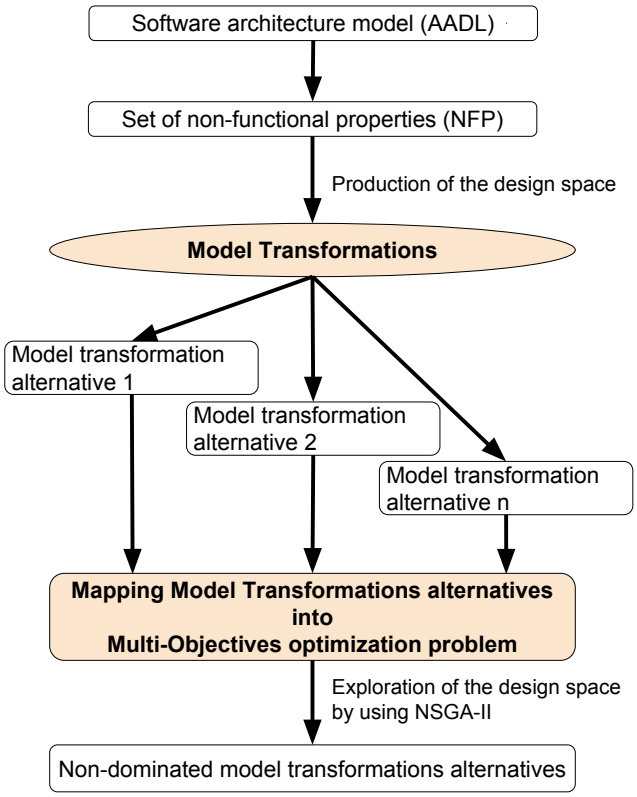

Fig. 2. Approach overview

A. System architecture modelisation

Since our application domain is embedded systems, we have chosen AADL (Architecture Analysis & Design Language) as the underlying architecture description language. AADL has been designed on the foundation of the architecture description language MetaH and its goals are (i) specification of quality analysis (e.g. safety); and (ii) design architectures for complex embedded systems. It offers a standardized semantics, mostly expressed using natural language and the components are composed hierarchically according to standardized composition rules. AADL also allows the designers to expand model by defining new property sets in their own specific way. The other reason of choosing AADL is to integrate our work in an existing framework RAMSES[2] which apply selection of design alternatives.

B. Model transformations composition

In major cases, model transformations are grouped into two main categories of model transformation languages : (i) rule-based transformation languages, or (ii) imperative transformation languages [9].

[2]Refinement of AADL Models for the Synthesis of Embdded Systems. http://penelope.enst.fr/aadl/wiki/Projects

The first category express model transformation logics in a way that is easy to interpret and **adapt**. It permit to modify the pattern matching part of a transformation rule TR which is sufficient to modify the set of elements this rule applies to. In contrary, the adaptation of imperative transformation languages are more difficult and requires a deep understanding of the control flow graph that leads to the execution of one or another transformation.

We used in our work the first category of languages since it more easy to adapt into other methods (in this paper MOEA). Where model transformations are endogenous : the source and target models are the same. In our work we use a AADL to AADL model transformations in order to improve (or refine) the architecture of an embedded system while representing the result of this refinement in an analysable model. More specific, we use the rule-based transformation language ATL (for Atlas Transformation Language).

A model transformation MT is generally composed of a set of transformation rules (TR_i). Each transformation rule TR_i is defined by a pair $< E_j, R_j >$, where E_j is the scope of the transformation rule TR_i : the set of elements from the input model the transformation rule TR_i can be applied to. And R_j is the set of rules or actions which TR_i executes when it is applied to an element of E_j (i.e. creation of elements in the target model).

C. Mapping model transformation alternatives into a multiple-objectives optimization problem

The next step of our approach consist in using model transformations definitions described in Section IV-B and integrate them into a multiple-objectives optimization problem. Where the population to optimize or the design space represent a set of model transformations alternatives and the objectives to fulfils are NFP such as reliability, availability and response time. The integration process is described in the following sections.

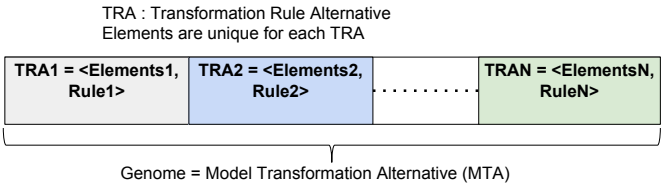

Fig. 3. Genome overview

1) ***Genome representation****:* The major element to define in a multiple-objective optimization problem (MOOP) is the population to optimize. In our work, the population is a set of model transformations alternatives (MTA). Where each genome is a MTA and each gene represent the composition of this MTA.

Figure 3 gives an overview of our mapping of a MOOP genome. A genome G represent a MTA and is composed of a sequences of genes. A gene is composed of a set of Transformation Rules Alternative (TRA). Each TRA is, in turn, encoded as a pair $< Elements, Rule >$ where $Elements$ is a list of elements from the input model, and $Rule$ is a transformation rule which can be applied to $Elements$.

2) ***Fitness evaluation of each genome****:* In each generation, the fitness of every genome in the population is evaluated. The fitness is usually the value of the objective function in the optimization problem being solved. We defined in our work one fitness function corresponding to three objectives reliability, availability and response time. The aim of our work is to maximize these three objectives.

The more fit genomes are stochastically selected from the current population, and each genome is modified (recombined and possibly randomly mutated) to form a new generation.

3) ***Genetic operators****:* In NSGA-II, in each iteration, the N genomes selected from the previous generation are used to create new N genomes using genetic operators. This improves the existing solutions by mixing their genetic material (crossover) and/or by creating new material (mutation). Before applying the operators, the solutions are selected according to their fitness values, see Figure 1. In our work, binary tournament selection, crossover and mutation operators are used.

Binary tournament selection : This operation consists of choosing some genomes at random in the population, and selecting the fittest two for reproduction. The selection criteria are the rank of the containing front and the crowding distance for solutions within the same front. Several tournaments are run to produce the needed genomes.

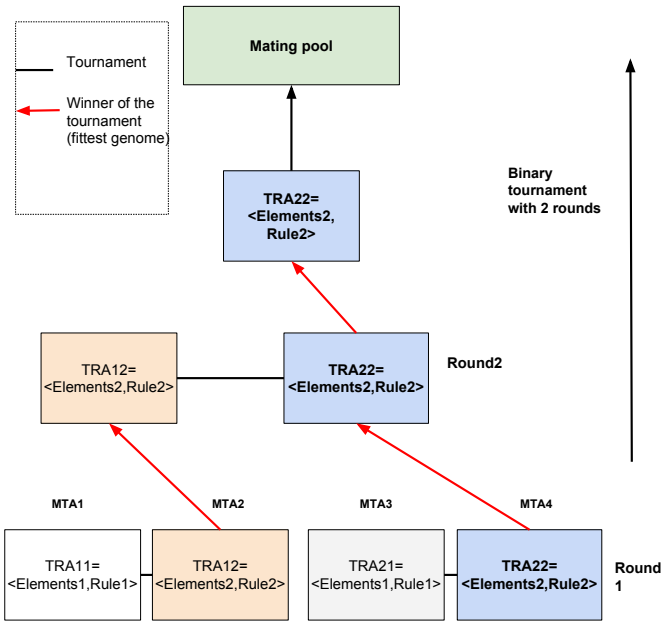

Fig. 4. Tournament Competition between Genomes

Figure 4 illustrates this operation. On this figure, we suppose we have four model transformations alternatives (genomes) MTA1, MTA2, MTA3 and MTA4 respectively composed of the following genes TRA11, TRA21, TRA31 and TRA41. We apply two tournaments between paired MTAs to produce the needed MTAs. In the first round we have two tournaments between two different pairs of MTAs (MTA1 with MTA2 and

MTA3 with MTA4). In the second round we apply another tournament on the victorious MTAs in the previous round (MTA2 and MTA4) to get the fittest MTA which will be put in the mating pool for reproduction (generate new MTAs). In Figure 4 the fittest MTA which is expected to appear in the mating pool is MTA4.

Crossover : The crossover consists of producing new genomes from the existing ones. When two genomes are selected using the binary tournament selection, two offspring solutions are created, with a given crossover probability, by exchanging parts of the parent genomes. This consists in randomly selecting a cut point in the genome, and all the genes beyond that point in either parent are swapped between the two parents.

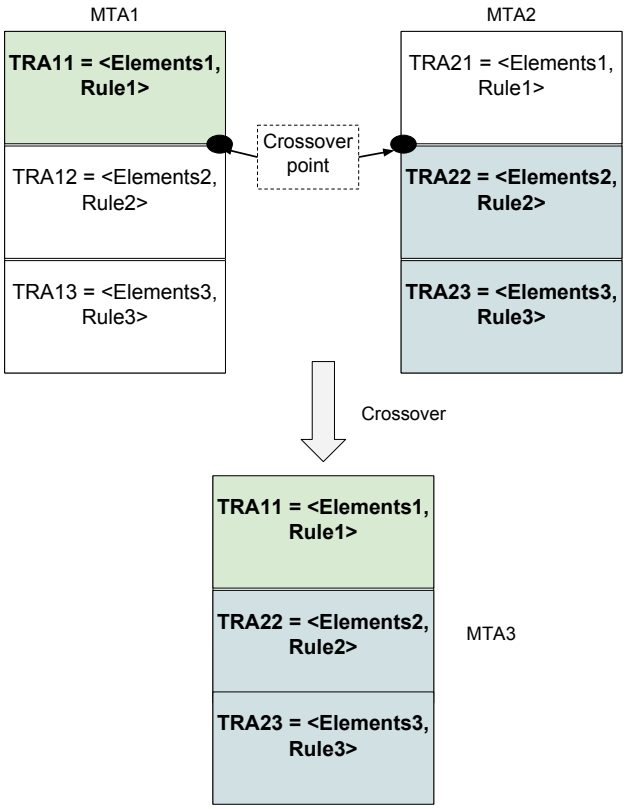

Fig. 5. Crossover operator

Figure 5 illustrates the result of applying the crossover operator to two parent genomes MTA1 and MTA2 respectively composed of the following genes {TRA11,TRA12,TRA13} and {TRA21,TRA22,TRA23} to produce a new genome MTA3 composed of new sequence {TRA11,TRA22,TRA23}. As can be seen in Figure 5, this operator combines TRA11 from MTA1 and TRA22,TRA23 from MTA2 by selecting a single point on the genome and swapping the genes between the two genomes that lie beyond this point.

Mutation : After performing the crossover, the obtained solutions could be mutated with a given mutation probability.

The mutation operator act on a single genome MTA1 to obtained a muted genome MTA11 acts as follows. The

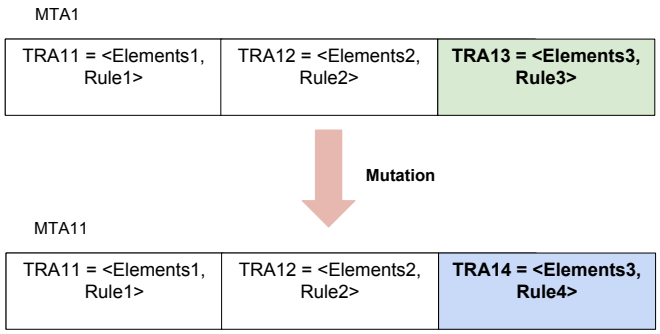

Fig. 6. Mutation operator

mutation operator selects a gene at random in the genome and changes its value. In Figure 6, the third gene TRA13 is selected to mutate into a new gene TRA14 : we apply a different rule (i.e. rule4) to the set of selected elements.

The final step of our approach consists in using the NSGA-II algorithm described in Section 1 to our optimization design problem defined above to find the set of non-dominated model transformations alternatives.

V. Related Work

Our work is based on system performance prediction [2] by using model transformations techniques and multi-objective metaheuristic optimization [4].

A. model transformations techniques

Recently, some works have focused on the definition of model transformation alternatives. The first works introduced the superimposition mechanism [17] to implement transformations alternatives. Another work [13] proposed to implement transformations alternatives by using rule-based transformation languages. These works has been continued in [8] in order to deal with architecture quality problems.

Our contribution is based on this work since it enable the factorization and composition of model transformations. However, this approach doesn't propose any algorithm to select the non-dominated (or pareto) model transformations alternatives which fulfil at best a set of competing non-functional properties.

B. multi-objective metaheuristic optimization

Aleti et al. [1] have developed the ArchOpteryx framework. ArchOpteryx assists the system architects during the design phase to create architectures that meet non-functional properties by using multi-objective optimization strategies.

Koziolek et al. [15] also developed their own framework Peropteryx for improvement of system architecture models using a metaheuristic search guided by architectural tactics.

Unlike these two frameworks, in our approach design alternatives are implemented as model transformations driven by NFP. The use of model transformations help the design architects to (i) automating the system production; (ii) reusing architecture alternatives in other projects (or products) and (iii) solving the competition issue between NFP.

Compared to DesignBots [7], in our method model transformations are not decomposed into subplans, each transformation is performed atomically.

For the optimization of NFP another work consists at using a cyclic process [3]. The benefit of this process is its simplicity. The process can be tailored to different non-functional properties by using specific evaluation methods and architecture transformations. But this process focuses on only one non-functional property at a time unlike us who take into account several properties at a same time (e.g. reliability, availability and response time).

VI. Conclusion

An architecture design method has been presented that explicitly addresses the NFP put on the architecture. The proposed approach aims at improving system architecture models using two mechanisms. On the one hand, model transformations compositions to formalize model design alternatives. On the other hand, an elitist multiple-objective optimisation algorithm to explore this design space.

The main benefits of this approach is to handle a large search spaces of quality optimization problems and to automate the identification of good architecture design alternatives. This will reduce development costs and improve the quality of the final system. Additionally this approach enables the ranking of design alternatives with respect to conflicting NFP and the possibility to reuse these alternatives in other projects or products by reusing model transformations.

In the future, we plan to integrate our approach in the framework RAMSES and evaluating our approach in more details quantitatively (i.e. experimenting it on a an industrial case study) [10].

VII. Acknowledgment

This research work has been carried out under the leadership of the Technological Research Institute SystemX, and therefore granted with public funds within the scope of the French Program “Investissements d’Avenir”.

References

[1] A. Aleti, S. Bjornander, L. Grunske, and I. Meedeniya, “Archeopterix: An extendable tool for architecture optimization of aadl models,” in *Model-Based Methodologies for Pervasive and Embedded Software, 2009. MOMPES ’09. ICSE Workshop on*, May 2009, pp. 61–71.

[2] S. Balsamo, A. di Marco, P. Inverardi, and M. Simeoni, “Model-based performance prediction in software development: a survey,” *Software Engineering, IEEE Transactions on*, vol. 30, no. 5, pp. 295–310, May 2004.

[3] J. Bosch, P. Molin, J. Bosch, and P. Molin, “Software architecture design: Evaluation and transformation,” in *In 1999 IEEE Engineering of Computer Based Systems Symposium. IEEE Computer Based Systems*, 1999, pp. 4–10.

[4] C. A. C. Coello, “A comprehensive survey of evolutionary-based multiobjective optimization techniques,” *Knowledge and Information Systems*, vol. 1, pp. 269–308, 1998.

[5] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, “A fast and elitist multiobjective genetic algorithm: Nsga-ii,” *Trans. Evol. Comp*, vol. 6, no. 2, pp. 182–197, Apr. 2002. [Online]. Available: http://dx.doi.org/10.1109/4235.996017

[6] K. Deb and D. Kalyanmoy, *Multi-Objective Optimization Using Evolutionary Algorithms*. New York, NY, USA: John Wiley & Sons, Inc., 2001.

[7] J. A. Díaz-Pace and M. R. Campo, “Using planning techniques to assist quality-driven architectural design exploration,” in *Proceedings of the Quality of Software Architectures 3rd International Conference on Software Architectures, Components, and Applications*, ser. QoSA’07. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 33–52. [Online]. Available: http://dl.acm.org/citation.cfm?id=1784860.1784865

[8] M. Drago, C. Ghezzi, and R. Mirandola, “Towards quality driven exploration of model transformation spaces,” in *Model Driven Engineering Languages and Systems*, ser. Lecture Notes in Computer Science, J. Whittle, T. Clark, and T. Khne, Eds., vol. 6981. Springer Berlin Heidelberg, 2011, pp. 2–16. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-24485-8_2

[9] P. V. Gorp, O. Muliawan, D. Janssens, P. V. Gorp, O. Muliawan, and D. Janssens, “Integrating a declarative with an imperative model transformation language,” 2006.

[10] G. Loniewski, E. Borde, D. Blouin and E.Insfran ´, “An Automated Approach for Architectural Model Transformations,” in *Information Systems Development, 22nd International Conference, ISD 2013, Sevilla, Spain, September 2013. Proceedings*, 2013, pp. 295-306. [Online]. Available: http://infres.enst.fr/∼borde/isd2013loniewski.pdf

[11] A. Johnsen and K. Lundqvist, “Developing dependable software-intensive systems: Aadl vs. east-adl,” in *Ada-Europe 2011*, A. Romanovsky and T. Vardanega, Eds. Springer-Verlag, June 2011, pp. 103–117. [Online]. Available: http://www.es.mdh.se/publications/1753-

[12] F. Jouault and I. Kurtev, “Transforming models with atl,” in *Proceedings of the 2005 International Conference on Satellite Events at the MoDELS*, ser. MoDELS’05. Berlin, Heidelberg: Springer-Verlag, 2006, pp. 128–138. [Online]. Available: http://dx.doi.org/10.1007/11663430_14

[13] A. Kavimandan, A. Gokhale, G. Karsai, and J. Gray, “Managing the quality of software product line architectures through reusable model transformations,” in *Proceedings of the Joint ACM SIGSOFT Conference – QoSA and ACM SIGSOFT Symposium – ISARCS on Quality of Software Architectures – QoSA and Architecting Critical Systems – ISARCS*, ser. QoSA-ISARCS ’11. New York, NY, USA: ACM, 2011, pp. 13–22. [Online]. Available: http://doi.acm.org/10.1145/2000259.2000264

[14] A. G. Kleppe, J. Warmer, and W. Bast, *MDA Explained: The Model Driven Architecture: Practice and Promise*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2003.

[15] A. Koziolek, H. Koziolek, and R. Reussner, “Peropteryx: Automated application of tactics in multi-objective software architecture optimization,” in *Proceedings of the Joint ACM SIGSOFT Conference – QoSA and ACM SIGSOFT Symposium – ISARCS on Quality of Software Architectures – QoSA and Architecting Critical Systems – ISARCS*, ser. QoSA-ISARCS ’11. New York, NY, USA: ACM, 2011, pp. 33–42. [Online]. Available: http://doi.acm.org/10.1145/2000259.2000267

[16] N. Medvidovic and R. N. Taylor, “A classification and comparison framework for software architecture description languages,” *IEEE Trans. Softw. Eng.*, vol. 26, no. 1, pp. 70–93, Jan. 2000. [Online]. Available: http://dx.doi.org/10.1109/32.825767

[17] E. Navarro, C. E. Cuesta, D. E. Perry, and C. Roda, “Using model transformation techniques for the superimposition of architectural styles,” in *Proceedings of the 5th European Conference on Software Architecture*, ser. ECSA’11. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 379–387. [Online]. Available: http://dl.acm.org/citation.cfm?id=2041790.2041840

[18] N. Srinivas and K. Deb, “Multiobjective optimization using nondominated sorting in genetic algorithms,” *Evolutionary Computation*, vol. 2, pp. 221–248, 1994.

[19] M. Tisi, F. Jouault, P. Fraternali, S. Ceri, and J. Bézivin, “On the use of higher-order model transformations,” in *Model Driven Architecture - Foundations and Applications, 5th European Conference, ECMDA-FA 2009, Enschede, The Netherlands, June 23-26, 2009. Proceedings*, 2009, pp. 18–33. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-02674-4_3

[20] H. Yu and A. Vahdat, “The costs and limits of availability for replicated services,” *SIGOPS Oper. Syst. Rev.*, vol. 35, no. 5, pp. 29–42, Oct. 2001. [Online]. Available: http://doi.acm.org/10.1145/502059.502038

[21] E. Zitzler, M. Laumanns, and L. Thiele, “Spea2: Improving the strength pareto evolutionary algorithm,” Tech. Rep., 2001.

[22] E. Zitzler and S. Knzli, “Indicator-based selection in multiobjective search,” in *in Proc. 22nd International Conference on Information Systems Development (ISD 2013*. Springer, 2004, pp. 832–842.