Acceptance-based Software Architecture Deployment for Improvement of Existing Applications

Hannes Klee*, Michael Buchholz*, Torben Materna†, Klaus Dietmayer*

*Institute of Measurement, Control and Microtechnology, University of Ulm
Albert-Einstein-Allee 41, 89081 Ulm, Germany
Email: {hannes.klee, michael.buchholz, klaus.dietmayer}@uni-ulm.de
†Deutsche Accumotive GmbH & Co. KG, 73230 Kirchheim u. Teck/Nabern, Germany
Email: torben.materna@daimler.com

***Abstract*—A lot of approaches are already published to solve software architecture deployment problems. Most of them are intended for academic use and assume that the software components can be deployed freely on the hardware components. But for an improvement of existing applications, non-functional constraints will have a high influence on the acceptance of the automatically generated solutions. In this paper, the organization of the engineers and their tasks as well as the amount of changes regarding to a currently applied system are considered. To gain a smart and reduced interface between hardware components, a method is presented to reduce the communication overhead for an existing architecture. Additionally, the deployment problem is restricted by the amount of changes in comparison to an initial deployment. This approach is tested on a realistic case study to show that it is possible to achieve high improvements with only small changes of the system.**

I. Introduction

Automation of software architecture deployment has been subject of research in recent years, and great advances have been made. Tasks like software design [1], performance prediction [2], [3], and resource usage [4] have also been addressed, and methods for the latter two are applied in practice. However, despite a huge number of different approaches exists, automated software design methods are either intended for academic use or are very specific, and they have never become a product available on the market. One of the reasons for this might be the fact that most developments do not start from scratch, but are e.g. based on a previous version/release/edition, and several non-functional constraints apply, which are not covered or cannot be solved by the automated methods.

An example for a well known software design problem is the software deployment problem where software components (SWCs) are deployed to Electronic Control Units (ECUs). In the automotive industry, a high number of ECUs are connected through data buses and hundreds of SWCs. The search space of possible solutions easily becomes very big and, thus, an automation of the deployment process would be helpful. However, this is only possible if the assumption can be made that the SWCs are independent of the hardware or at least that they can be categorized in hardware dependent and independent ones. Additional restrictions apply, which are not necessary technical, but based on the organization structure of development teams. Moving SWCs from one ECU to another would mean to move either the respective responsibilities and expertise to another team, or to move people from one team to another. However, this obviously high overhead is not regarded within the existing automated software design frameworks, which is why they have not been adopted by the industry. The acceptance for a change and the will to cope with the considerable additional effort exists only if a big improvement of the quality criteria is achieved. In this paper, a solution should be created that possesses a reduced and smart interface between the ECUs with a reasonable change rate. Possible quality criteria are summarized in [1], however, an acceptance rate is not listed. From a technical point of view, the optimization of the given problem even in case of several ECUs and hundreds of SWCs can be solved. Optimization techniques are widely spread and are able to consider multiple quality objectives and design constraints, e.g. [5], [6], [7], [8], [9] and many others. Commonly, the proposed methods are tested on problems like the Anti-lock Brake System (ABS) or randomly created systems with only few SWCs per ECU (e.g. 9 components and 6 ECUs in [10] or 60 components and 35 ECUs in [11]), which assume that function blocks (FBs) can be deployed freely on different ECUs.

Typically, a function block consists of many strongly connected functions. These functions are developed and maintained by a team of engineers from which one person is responsible for a couple of functions. In this context, the functions are described as SWCs and the surrounding area where they are placed as FB. With this detailed operating level, it is possible to change only small parts of an existing system by deploying the SWCs instead of the whole FBs. Even though this procedure leads to a considerable increase of possible

978-1-4799-7560-0/15 $31.00 © 2015 IEEE
DOI 10.1109/SSCI.2015.255

deployments, the acceptance of a solution of an automated system will be much more likely if only small parts of the system are changed with a maximum benefit with respect to the quality criteria.

In this paper, a subsystem of an electric driven automobile is used to demonstrate the reduction of communication overhead between ECUs. With the existing solution as reference, the amount of changes will be evaluated and the total amount of communicated signals between ECUs is reduced.

The rest of the paper is structured as follows: In Section II, the problem is stated in detail and the resulting tasks for the paper are derived. The proposed approach is introduced in Section III, which allows getting feasible solutions according to the acceptance rate. This method is evaluated on a realistic problem, which is described in Section IV, and the results are discussed in Section V. The paper closes with some conclusions as well as an outlook on future work in Section VI.

II. Problem Description

General product development in the automotive and other areas is influenced by foreign developments. In consequence, a new product is usually a modification of the previous one. Learning from these projects is necessary for progress and leads to constant improvements. Despite a lot of advantages, this procedure runs into the risk of legacy issues. These in turn lead to a constantly growing system and, consequentially, a higher effort for testing and documenting during the development process. To reduce not only unnecessary signals but get a smarter interface between the ECUs, the software deployment problem is defined as one of the most important software design issues in the automotive industry.

Whereas the first problem can only be solved by a detailed analysis, solutions with big changes mostly have great improvements around the quality objectives like performance, cost, and reliability. Big changes possess the high risk to be rejected if they are proposed in an experienced team due to the administrative overhead or corporate team competences. Therefore, the focus of this paper lies on solutions not only with high improvements, but also on those with only small changes on the software architecture. The aim is to reduce the communication overhead (c_{com}) and the amount of signals which are transmitted between different ECUs (cnt_s). Furthermore, solutions must also be evaluated with a change rate on the basis of the necessary deployment changes which indicates the acceptance of a solution (r_{accept}). The overall cost function is represented by

$$c = c_{\text{com}} + \rho_1 * cnt_s + \rho_2 * r_{\text{accept}}. \quad (1)$$

In contrast to other cases (with only a few components per ECU), the approach goes one level deeper to gain more possibilities of small changes of the existing system. Therefore, the term function block is defined as a unit for a collection of related functions. Inside these FBs, there are several functions which are called software components in the following. It is assumed that these SWCs can be deployed freely not only on other FBs, but also on other ECUs. The difference of FBs and

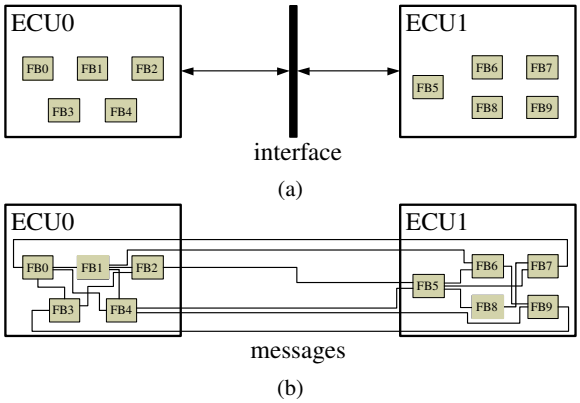

Fig. 1: (a) Information is shared over an interface via messages by a bus. (b) An interface is built on the basis of the real information flow of messages between function blocks (FBs) between ECUs. The intra-communication can be seen as well.

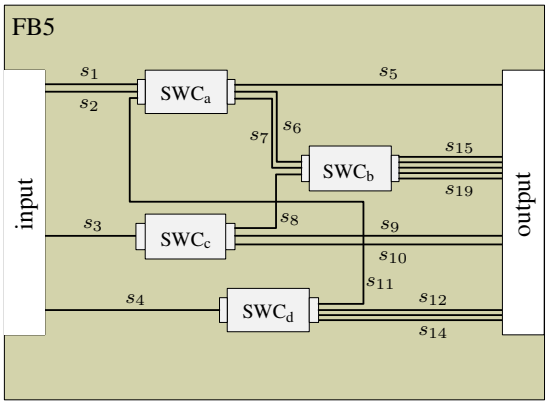

Fig. 2: Demonstration of signal transfer of FB5. This function block possesses 4 inputs and 11 outputs which are required from SWCs. Signals can have the same information. For example, s_{15} and s_{19} contain the same information but differ in the destination.

SWCs can be seen in Fig. 1 and 2, which will be explained in the following.

Fig. 1 demonstrates the difference between the communication between ECUs through an interface, see Fig. 1(a), and the actual communication through messages if the FBs could be connected directly, see Fig. 1(b). Messages consist of different signals generated by the inputs and outputs of the FBs. An example of one FB is shown in Fig. 2. As an FB contains several SWCs, the inputs and outputs of an FB are built on the signals generated by the SWCs itself. SWCs hold inputs and outputs for information exchange. This information is put in signals and sent to the requesting SWCs. Signals with the same source and destination are put in interactions similar to messages, with the difference that messages are based on bus messages which usually are transfered between ECUs only and contain related information. Interactions are created only

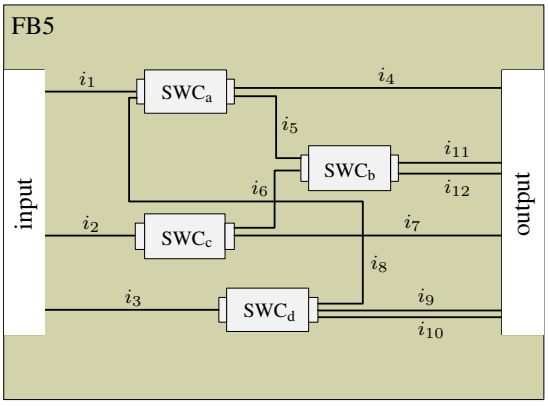

Fig. 3: Demonstration of interaction transfer of FB5. Interactions contain signals with same source and destination. The information of s_1 and s_2 is transmitted by i_1 and analogously for other signals.

to conclude signals with the same destination and have no other physical meaning.

In Fig. 2 and 3, the combination of signals to interactions is shown for FB5. For example, signal s_1 and s_2 are transfered from the same source to SWC$_a$. Here, the interaction i_1 combines these signals. The same holds for the signals s_6 and s_7 which are represented by interaction i_5. SWC$_b$ has 5 output signals which are transfered to two other SWCs. Therefore, the signals are summarized into two interactions (i_{11} and i_{12}). A signal in i_{11} and a signal in i_{12} can both contain the same information, but must have a different destination.

SWCs are defined as components u, ECUs as hosts h. Each component has a unique identifier u_{id}, a description and a collocation list $\boldsymbol{col}$. $\boldsymbol{col}$ is a vector, where the position i represents the binary information if the i^{th} component must be collocated with the current one. Signals s possess the properties data amount s_{data} and transmission frequency s_{freq} as well as the source and destination identifier ($s_{\text{src}}, s_{\text{dest}}$) of the belonging components. Because the same information can be shared with several components, there are signals with the same content but different destinations. These signals are assigned the same identifier s_{id}. An interaction i in turn bundles signals with the same source and destination.

The hardware is split into ECUs and buses. The ECUs are also identified by a unique index h_{id}. Additionally, they contain a vector $\boldsymbol{loc}$ which represents the location list. The i^{th} position describes the permission of the i^{th} component to be deployed to this ECU.

$\boldsymbol{p}_j(\text{sol}_i)$ is a vector for the solution sol_i, where the element j represents the index of the ECU where the j^{th} component is deployed to.

III. Proposed Approach

In the following section, the proposed approach is described in detail.

A. Deployment Process

The technical survey [1] shows that among other optimization strategies, evolutionary algorithms are the most commonly used. These algorithms are inspired by the natural development of reproduction and surviving. The main steps are very easy and many variations exist, where it must be noted that the parameter settings of these algorithms have a big influence on the quality of the resulting (near optimal) solutions. In this paper the non-dominated sorting version of the generic algorithm (NSGA2) is used because of the multi-objective problem and the ability to operate on binary strings. A detailed description of the NSGA2 algorithm can be found in [12]. Based on an initial population, it creates new solutions, evaluates their quality and checks their validity. After iterating this procedure until the stop criterion is met, the best solution will be selected. The creation of new solutions is made by mutation and crossover. The mutation procedure selects randomly a solution and changes the j^{th} entry of $\boldsymbol{p_j}(\text{sol}_i)$ to a random host. Sometimes, it would be necessary to do two or more mutations to get a better convergence of the algorithm. In the application of the NSGA2 in this paper, no clear results are achieved to decide which configuration is the best. Therefore, the standard setting with only one modification per mutation is kept. The crossover procedure is another method to create alternative solutions by selecting two previous solutions and pick the index for the beginning of the crossover. All following components are assigned to the alternative solution. Trying to modify the algorithm with PMX (partially mapped crossover [13]) does not lead to significant runtime or solution improvements, why here as well the standard setting is retained.

After the creation process, all new solutions must be evaluated on their quality (c_{com}, cnt_s and r_{accept}). Besides that, the solutions must be checked to be valid regarding the constraints. The quality objectives and the constraints will be explained in detail in the following subsections.

The whole process, shown in Fig. 4, is repeated until the stop criterion is met. In this paper, the total number of iterations for one single run is chosen as stop criterion and set to 600.

B. Communication Overhead

The quality objective c_{com} calculates only the inter-ECU communication which is also described as communication overhead. For the calculation, it has to be checked if an interaction connects two components which are assigned to different ECUs. The presented method will count the data amount only once per cycle because it is transfered via a bus where the same information is combined to one signal. The receiving ECUs will manage the distribution of the signals to the SWCs internally. An identifier s_{id} is used to decide whether an information is already transfered. For the calculation, a new list (listTxSignals) is created which is filled with the signals where source and destination component have different hosts. If a signal with the identifier s_{id} does not appear in the list, the upcoming communication costs will be calculated by the frequency and the data amount of the signal. Neglecting the

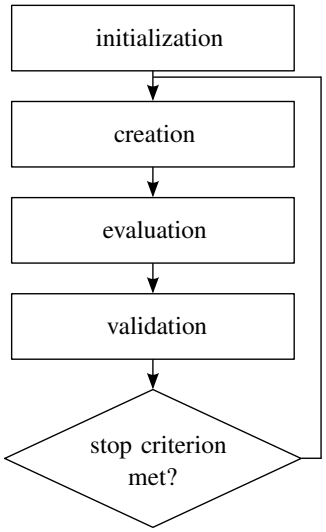

Fig. 4: Optimization procedure of the genetic algorithm.

protocol overhead and the reliability of the bus leads to the result that only the quantitative amount of data which must be transmitted is assessed. However, both features could be easily integrated additionally. For one bus, the calculation can be seen in Algorithm 1.

Algorithm 1 Process of communication overhead calculation.

```
listTxSignals = ∅
c_com = 0
cnt_s = 0
for ∀ interactions i do
  if p[i_src] ≠ p[i_dest] then
    for ∀ signals s in i do
      if s ∉ listTxSignals then
        c_com += s_freq * s_data
        cnt_s++
        listTxSignals.add(s_id)
      end if
    end for
  end if
end for
```

Simultaneously, the number of signals cnt_s, i.e. the information exchanged between different ECUs, is calculated. To take more than one bus into account, the following decisions are possible for the realization of data transfer between two components:

- connected directly with one bus
- not directly connected but through a gateway
- not possible

If an information must be forwarded through a gateway, the communication overhead must be added to the affected buses. For complex system, the routing is an additional task with several problems, which is not the focus of this paper.

If it is necessary to calculate more realistic overhead, the protocol overhead can be evaluated by adding the specific data amount, like header and the frame synchronization, to the interactions. Besides that an extended definition of the communication overhead must be formulated where the reliability and the delay of the bus transmission are taken into account. An exemplary formulation can be seen in [15].

C. Acceptance Rate

The acceptance of a new solution is dependent on several criteria. One of the most significant influences is the predicted amount of overhead for the realization. It is assumed that the acceptance is proportional to the amount of SWC deployment changes:

$$r_{\text{accept}} \sim \frac{1}{r_{\text{change}}}, \text{ if changes } > 0. \tag{2}$$

In the following, the calculation of the change rate is explained. An initial deployment is used which is referred to as "original" solution. For this original deployment $\boldsymbol{p}(\text{sol}_{\text{org}})$, the communication overhead is calculated. After this step, a reference for the following optimization is available including the amount of transferred signals and the number of components for each ECU. With this information, the change rate of new solutions can be evaluated. The calculation is done by comparing $\boldsymbol{p}(\text{sol}_i)$ of the current solution with $\boldsymbol{p}(\text{sol}_{\text{org}})$.

The following equations explain the necessary steps:

$$r_{\text{change},i} = \frac{c_{\text{change}}(\text{sol}_i)}{n}, \tag{3}$$

$$n = \text{number of components}, \tag{4}$$

$$c_{\text{change}}(\text{sol}_i) = \sum_k \text{change}[k], \tag{5}$$

$$\text{with change}[k] = \begin{cases} 1, \text{ if } \Delta(\text{sol}_i)[k] > 0 \\ 0, \text{ else} \end{cases}, \tag{6}$$

$$\Delta(\text{sol}_i) = \left|\boldsymbol{p}(\text{sol}_{\text{org}}) - \boldsymbol{p}(\text{sol}_i)\right|. \tag{7}$$

D. Communication Restrictions

In addition to the restrictions of the location and collocation, the limits of transmitted signals and the predicted communication overhead are defined in the following. With an upper limit number of signals, the optimization process can still find solutions with different communication overheads and change rates. Reasons for a limiting signal number could be maximum utilization of the bus, assessments on improvements with different signal numbers and the specification of the project requirements. Limiting the overhead leads to the effect that the resulting solutions are characterized by different amounts of signals but similar overhead.

E. Result Representation

The solutions are plotted in dependence on the quality criteria whereas just the Pareto front [14] is displayed. This front contains those solutions which are not dominated by another during the optimization. For the selection of one of the solutions, an additional cost function is defined, see equation

1, which combines all the quality objectives with a chosen weighting.

IV. Experiment

The approach is demonstrated with an example motivated by the functionality of an electric driven vehicle focusing on the high voltage battery management system (BMS). The borders of the subsystem are drawn by the energy management as well as other involved control units to ensure a safe and efficient operation of the HV battery. The inputs and outputs of the components are translated into signals with a description of the source and destination component, data amount and frequency.

A. Problem Instances

The investigation object contains 202 SWCs with 450 outputs. These outputs are translated into connections between the SWCs with a resulting number of 796 signals. Signals with the same source and destination component are combined to 544 interactions. The hardware is specified with two hosts (BMS and the rest of the vehicle) and one bus. The original deployment has got 96 SWCs allocated on the BMS and 106 SWCs allocated on the rest of the vehicle. This solution has a transmission amount of 123 signals and a communication overhead of 1895 bits per cycle. The deployment for optimization consists of dependent and independent components. Here, 61 SWCs are fixedly assigned to the BMS and 33 SWCs to the rest of the vehicle. Dependent components contain functionality like sensor interpretation and actor signal generation (directly in contact with hardware). Other reasons might be safety, redundancy or functionality which should be placed on the specific ECU. Subtracting the dependent SWCs, there are 108 components left which can be deployed with the optimization approach.

The used example does not restrict the hardware resources to get a general deployment solution without knowledge of the specific hardware. Furthermore, it is assumed that there is no delay, no failure rate, and the transmission rate is unlimited.

B. Algorithmic Settings

For the experiment, a population size of 500 and 250 children per iteration are chosen. The algorithm stops after 600 iterations. The mutation rate is set to 0.9 and the crossover rate to 0.8. The mating pool has 750 elements. Unfeasible solutions will be discarded. Due to the size of the search space and the risk of suboptimal solutions, the procedure is executed with 60 different seeds. The proposed approach is integrated within the tool ArcheOpterix [15], [5].

V. Results & Discussion

Fig. 5 shows the result for only two quality criteria. The communication overhead and the change rate are evaluated for the first run in red. The solution with the highest improvement (about 39.1% from 1895 to 1154 bits/cycle) has a change rate of 8.9%, which implies the reallocation of 18 SWCs in comparison to the original solution. The results after all runs can also be seen in Fig. 5 in blue. Here, an improvement of

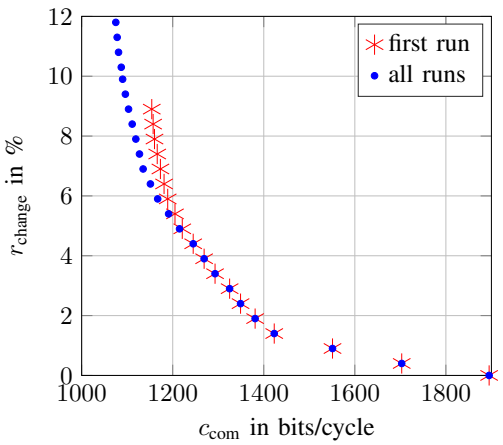

Fig. 5: Non-dominated solutions of 60 runs with communication overhead and change rate.

about 43.3% (from 1895 to 1075 bits/cycle) with a change rate of 11.9% (24 SWCs) is achieved.

In the next step, the signal amount is added as third criterion. Now, solutions must be evaluated for all three criteria which leads to new solutions in comparison to the previous experiment. The results are shown in Fig. 6. The Pareto solutions are taken from 60 instances with different seeds. For the criterion c_{com}, the method yields the same extreme solution as above (highlighted as circle) with minimal communication overhead with a signal amount reduction of approx. 39.8% to a number of 74 transfered signals between the two ECUs (instead of 123). The other extreme solution with a minimum signal amount, shown in black, achieves a communication overhead with only 43% (1081 bits/cycle) in comparison to the original solution. The number of signals which must be transfered can be reduced to 47.2% fewer signals, which means a total amount of 65 signals and a change rate of 11.4% (23 SWCs). Those two solutions differ only in one SWC change in comparison to the original solution and possess a similar communication overhead ($\Delta c_{\text{com}} = 0.3\%$). But a closer look at the deployment shows that the solutions differ in 7.4% of the transferring signal amount. These differences are possible because of the different data sizes specified in the signals.

Regarding Eq. (1), a specific weighting is chosen. If ρ_2 is the dominant weight, with $\rho_1 > 0$ and $\rho_2 \gg \rho_1$, solutions with only a few changes and the same total costs are selected. Again, two non-dominated solutions are found with only one change which possesses completely different properties. One of these solutions reduces the communication overhead from 1895 to 1703 bits/cycle which means an improvement of approx. 10.1%. This solution must transfer 117 signals via the interface (4.9% improvement). The other solution reaches a number of 115 inter-communicated signals (6.5% improvement) and a communication overhead of 1767 bits/cycle (6.8%). With these results, it can be seen, that even

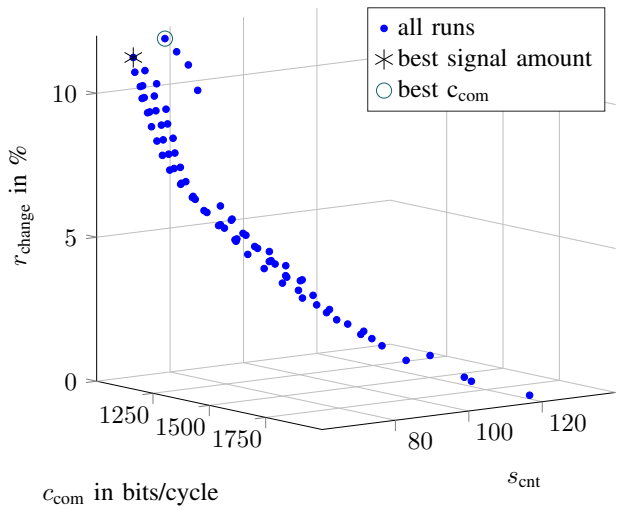

Fig. 6: Non-dominated solutions of 60 runs with communication overhead, change rate and signal amount.

with one SWC change, the communication overhead of the investigated system can be improved by 10%.

By specifying communication restrictions, the cost function is excluded and several solutions can be found. For example, with a fixed signal amount of 99 signals (19.5%), the approach finds three non-dominated solutions with [1325 (30%), 2.9], [1385 (27.1%), 2.4] and [1537 (19.4%), 1.9] (bits/cycle, change rate). The same holds similarly for a fixed communication overhead.

VI. Conclusions

In this paper, the problem of the improvement of the communication overhead of an existing system was addressed. Applying a theoretical best solution runs into the risk to be rejected if it is proposed in an experienced team due to non-technical reasons. Therefore, the presented approach does not only regard the communication overhead, but also an indicator for the acceptance probability. The acceptance is influenced by the change rate of the software components which must be allocated on ECUs. The existing solution was set as reference and an amount of changes was taken into account in form of the change rate. Besides that, the signal amount was used as additional quality criterion for gaining a smart and reduced interface between ECUs.

In the investigated case study, it was shown that with only one change of a SWC deployment, a solution with 10% less communication overhead could be achieved. Furthermore, the method found much better improvements with up to 43.3% less communication overhead at the expense of the change rate growing to 24 SWCs (11.9%).

It should be mentioned that the approach is independent of the optimization algorithm and other methods like Simulated Annealing and Pareto Ant Colony Algorithm could be applied, too.

In the future, more non-functional constraints like the corporate organization of the software components to the engineers will be integrated. Then, solutions can be penalized with changes in the organization structure. Another task will be the priorization of the signals. This can be usefully if solutions have the same transferring signal amount and change rate.

Acknowledgment

We want to thank Indika Meedeniya for the support on the ArcheOpterix tool.

References

[1] A. Aleti, B. Buhnova, L. Grunske, A. Koziolek, and I. Meedeniya, "Software Architecture Optimization Methods: A Systematic Literature Review," *IEEE Trans. Software Eng.*, vol. 39, no. 5, pp. 658–683, May 2013.

[2] S. Balsamo, A. di Marco, P. Inverardi, and M. Simeoni, "Model-Based Performance Prediction in Software Development: A Survey," *IEEE Trans. Software Eng.*, vol. 30, no. 5, pp. 295–310, May 2004.

[3] S. Becker, L. Grunske, R. Mirandola, and S. Overhage, "Performance Prediction of Component-Based Systems: A Survey from an Engineering Perspective," in *Architecting Systems With Trustworthy Components, Volume 3938 of LNCS*. Springer, 2006, pp. 169–192.

[4] A. Ferrari, M. Di Natale, G. Gentile, G. Reggiani, and P. Gai, "Time and Memory Tradeoffs in the Implementation of AUTOSAR Components," in *Proceedings of the Conference on Design, Automation and Test in Europe*, ser. DATE, 2009, pp. 864–869.

[5] I. Meedeniya, A. Aleti, I. Avazpour, and A. Amin, "Robust ArcheOpterix: Architecture optimization of embedded systems under uncertainty," in *ICSE Workshop on Software Engineering for Embedded Systems (SEES)*, June 2012, pp. 23–29. [Online]. Available: http://mercury.it.swin.edu.au/g_archeopterix/experiments/SEES2012/

[6] W. Peng, H. Li, M. Yao, and Z. Sun, "Deployment optimization for AUTOSAR system configuration," in *International Conference on Computer Engineering and Technology (ICCET)*, vol. 4, April 2010, pp. V4–189–V4–193.

[7] E. Wozniak, A. Mehiaoui, C. Mraidha, S. Tucci-Piergiovanni, and S. Gerard, "An optimization approach for the synthesis of AUTOSAR architectures," in *IEEE Conference on Emerging Technologies Factory Automation (ETFA)*, Sept. 2013, pp. 1–10.

[8] M. Walker, M.-O. Reiser, S. T. Piergiovanni, Y. Papadopoulos, H. Lönn, C. Mraidha, D. Parker, D.-J. Chen, and D. Servat, "Automatic optimisation of system architectures using EAST ADL," *Journal of Systems and Software*, vol. 86, no. 10, pp. 2467–2487, 2013.

[9] A. Sangiovanni-Vincentelli and M. Di Natale, "Embedded System Design for Automotive Applications," *Computer*, vol. 40, no. 10, pp. 42–51, Oct. 2007.

[10] I. Meedeniya, I. Moser, A. Aleti, and L. Grunske, "Architecture-based Reliability Evaluation Under Uncertainty," in *Quality of Software Architectures (QoSA)*, Nov. 2011, pp. 85–94.

[11] I. Moser and S. Mostaghim, "The automotive deployment problem: A practical application for constrained multiobjective evolutionary optimisation," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC)*, July 2010, pp. 1–8.

[12] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, April 2002.

[13] D. E. Goldberg and R. Lingle, Jr., "Alleles, Loci and the Traveling Salesman Problem," in *Proceedings of the 1st International Conference on Genetic Algorithms*, 1985, pp. 154–159.

[14] J. D. Knowles and D. W. Corne, "Approximating the Nondominated Front Using the Pareto Archived Evolution Strategy," *Evol. Comput.*, vol. 8, no. 2, pp. 149–172, June 2000.

[15] A. Aleti, S. Bjornander, L. Grunske, and I. Meedeniya, "ArcheOpterix: An extendable tool for architecture optimization of AADL models," in *ICSE Workshop on Model-Based Methodologies for Pervasive and Embedded Software (MOMPES)*, May 2009, pp. 61–71.