An Approach to Guide the System Engineer during the Design Space Exploration Process*

Maike Rosinger, Matthias Büker, Raphael Weber

{maike.rosinger|matthias.bueker|raphael.weber}@offis.de

Abstract: Due to the increasing complexity in the development of embedded systems, it cannot be expected that capturing and formalizing the expert knowledge immediately leads to a result that meets the system engineer's expectations and satisfies all design goals sufficiently well. Therefore, we propose the "Architecture Wizard" as an approach to support the system engineer in finding optimized solutions for the deployment of software components to hardware resources. This engineering process, called Design Space Exploration, consists of different activities, which may be supported by additional wizards. For the Architecture Wizard to be flexible and modular we model the engineering process with the Software & Systems Process Engineering Meta-Model (SPEM 2.0). This enables arbitrary process definitions and the automatic generation of process wizard skeletons. Individual activities may then be integrated into this skeleton to offer a seamless design process to guide the system engineer.

1 Introduction

The development of embedded systems e. g. in the automotive and avionics domain is getting more and more complex consisting of several critical design decisions and a multitude of alternative implementations of the systems functionality. Due to this complexity an - at least partly - automated support at finding design alternatives is desirable. For this purpose techniques from the field of optimization are used, which are denoted in this context as *Design Space Exploration* (DSE) methods (see for example [BDE+13]). The goal of such a design space exploration is to find a valid implementation of a system that fulfills all design goals as good as possible. An implementation is considered to be valid if all constraints of the system are satisfied. Although a lot of different methods and techniques exist in literature they are rarely used in practice to tackle real DSE problems. One reason for this can be found in the high variability and complexity of the design processes for embedded systems.

While automated optimization techniques can be usefully applied in a restricted setting with well-known parameters the process of making critical design decisions as in the mentioned context highly depends on expert knowledge and engineering experience. To be processable by automated DSE optimization techniques this knowledge and experience

*This work was supported by the Federal Ministry for Education and Research (BMBF) under support code 01IS12005M, SPES_XT

needs to be formalized. This is typically done in terms of metrics allowing to specify optimization objectives and constraints. Because the means of expression for those metrics depend on the utilized DSE method the system engineer (also denoted as user) is usually not familiar with them but is reliant on such metrics when specifying the DSE problem to be solved. Furthermore, design goals and constraints often have complex interrelations that are however not always known in advance due to the high system complexity.

All of this makes it hard to predict which effects the specified metrics will have for the optimization technique and which solutions can be expected. Additionally, the input parameters of such methods are often quite abstract or technical which makes it hard for the user to express his concrete objectives and constraints.

Due to these problems and the high complexity of embedded system design it cannot be expected that capturing and formalizing the expert knowledge in terms of metrics leads immediately to a result that meets the user's expectations and satisfies all design goals sufficiently well. Hence, it is usually necessary to apply the optimization methods repeatedly in an iterative process to get a satisfying result [CD08]. Thus, it is crucial to evaluate and visualize the resulting design alternatives help deciding which result fits best or how the input parameters might be adjusted for the next iteration.

The need to involve the user in the DSE process has already been identified in other publications. In [DLZV14] the academic partners and tool vendors of the ARAMiS[1] project were questioned about the importance of certain optimization objectives for a DSE and the characteristics of an *ideal* DSE. One result was that "DSE is a very complex task. Thus [...] the tools should guide/support the developer rather than making full automatic decisions. However, optimization directions have to be given by the developer." Furthermore, user-friendliness, comprehensibility, the possibility of comparing and visualizing results as well as the integration of the DSE methodology into the existing work and tool environment were mentioned as important aspects.

Thus, we define the following goals: (1) The user should be supported in the process of capturing and formalizing input parameters in terms of formal metrics for constraints and optimization objectives. (2) The user should be supported during the iterative application of a set of different optimization methods in a well-defined process allowing the step-wise refinement of input parameters. (3) The user should be enabled to inspect, evaluate and compare the results of a DSE run before proceeding with the next iteration.

The contribution of this paper related to the above mentioned goals is as follows: We propose the conceptual ideas and prototypical implementation of wizards to support the user at formalizing input parameters (1), traversing an iterative DSE process and choosing a DSE method (2), and evaluating results and making a design decision (3).

Related Work: An example for a user-guided iterative DSE approach can be found in [HJRE06] where instead of a fully automated black box method the user is included into an iterative optimization process based on evolutionary algorithms. Here, the user may choose between different pareto-optimal intermediate solutions he wants to consider for the next iteration and may also modify these solutions before proceeding. Another iterative approach is pursued in [NLM+14] where the design space can be searched for valid

[1]Funded by the Ministry of Education and Research (BMBF), http://www.projekt-aramis.de

solutions by using different tools. The knowledge from the results of previous searches can be used to refine the design space for further iterations. Beside the actual DSE tool DESERT other tools are offered as well e. g. to define constraints, manipulate the design space or generate executable models for analyses. Additionally, a tool to visualize the results is integrated.

Furthermore, there are approaches for DSE where the search is guided by model transformations. In [SHL10] user-defined requirements are translated into transformation rules that are applied sequentially on the system model. During this process the user is supported by a "transformation wizard" allowing to select and execute transformation rules. Another similar approach [HHRV11] guides the DSE in terms of hints originating from previous analyses and by identifying search paths that do not lead to a valid solution. Hints are given e. g. as a partial order of operations and a maximum number for the application of certain operations. Based on this, the most promising paths for finding a valid solution as well as the paths that will only lead to invalid solutions are identified.

More and more researchers also tackle the problem of visualizing (intermediate) results of optimization methods, which indicates that the integration of the user plays an increasingly important role. There are many kinds of representations including multi-dimensional coordinate systems as e. g. in [VEH14] and approaches as described in [TP11a] where the results of an evolutionary algorithm for multi-criteria optimization is represented as a tree. Here, each level of the tree corresponds to a parameter of the design space or an optimization objective. By using different colors and different scales for nodes the degree of fulfillment of optimization objectives of a solution are visualized. In [TP11b] this approach was utilized to compare the quality of the results of different evolutionary algorithms with respect to different metrics.

Outline: In the next section we describe the relevant preliminaries for our work. In Section 3 we present our DSE framework and the basic concept of the architecture wizard as a process wizard with several task wizard integrated. Sections 4 and 5 describe the realization of the architecture wizard based on SPEM 2.0 as well as the currently available task wizards. Finally, a conclusion of the paper is given in Section 6.

2 Preliminaries

In this section we describe the foundations of the architecture wizard. A process meta-model and general concept of wizards provide the basis of the architecture wizard.

2.1 Software & Systems Process Engineering Meta-Model 2.0

The Software & Systems Process Engineering Meta-Model (SPEM) 2.0 [Obj08] is a standard, which is defined by the Object Management Group (OMG) as a UML-Profile. SPEM is a process meta-model with focus on engineering processes to define software and system development processes and their input and output artifacts. The central element of the SPEM meta-model is an *Activity*. An activity groups a unit of work, e. g. sub-processes

and may contain other activities and tasks. A *Task* defines a work process that can be performed by one or more *Roles* like a system engineer. Furthermore, a task can be divided into multiple *Steps*. A step represents the most fine-granular unit of work which can be performed. In addition, a task is connected to the input and output *Work Products*. Work products are produced, modified, or used while a task is performed.

SPEM is structured into seven main meta-model packages. The structure divides the meta-model into logical units. Each unit has its own elements to define software and system development processes. The *Core* package contains the base for all classes of the other packages. The *Process Structure* package defines the base for all process models. In order to add textual descriptions (i. e. properties) to processes or process elements, the *Managed Content* package is used. The *Method Content* package provides concepts for modeling reusable elements independently of any processes such as *Work Product Definition*, *Role Definition*, *Task Definition*. Contrastingly, the *Process With Methods* package provides concepts about the use of different elements within the context of a life-cycle like *Work Product Use*, *Role Use*, *Task Use*, *Activity* and *Process*. The *Process Behaviour* package is intended for externally-defined behavior models (i. e. BPMN models). Finally, the *Method Plugin* package provides concepts to reuse whole processes (e. g. the DSE-Process).

The following terminology is applied for the use of SPEM. In the SPEM key terminology (see Figure 1) a distinction is made between method contents and processes. Method contents use work product definitions, role definitions, task definitions, and guidance. The process concept consists of elements used to represent processes. The main element is the activity that can be also nested in another activity.

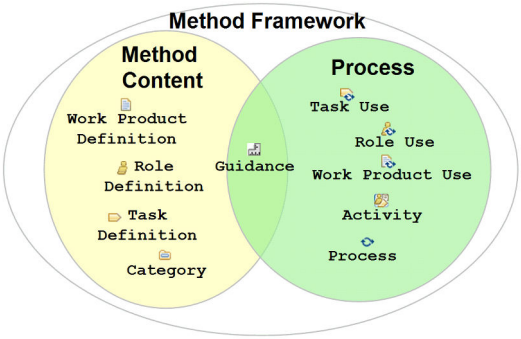

Figure 1: Key terminology of SPEM [Obj08].

2.2 Wizard and Wizard Pattern

In general, a *Wizard* is an assisting tool that is integrated in an application or software. The role of a wizard is to assist the user for a long or complicated task. Usually the task is new, the user not often performs it or he would like to have a fine-grained control over this task. A wizard is a concept to guide the user through a task step by step in a prescribed way and helps to avoid mistakes.

The *Wizard Pattern* enables to break up tasks into a series of units or groups of activities. It is also possible to define a strict order how units and groups of activities have to be performed but this is not required. Tidwell [Tid10] describes two ways how tasks may be split. The thematic breakdown of tasks defines that "the presentation order doesn't much matter because later choices don't depend on earlier choices." Another way is "[...] to split up tasks at decision points so that choices made by the user can change the downstream steps dynamically." The right balance between the size and number of units is important. Furthermore, the concept of the wizard pattern differentiates between two physical structures to build a wizard. In the *multiple page* wizard, each step is represented by a separate wizard page. In the *single page* wizard, all steps are represented on one page.

3 Solution Concept

The generic framework for Design Space Exploration proposed in [BHST13] serves as a basis for the architecture wizard. It describes an abstract DSE process consisting of activities and their interfaces in terms of required input artifacts and produced output artifacts. It further allows the integration of different concrete DSE methods if they are compatible to the respective interface. This framework may evolve over time and can be extended or refined e. g. by adding new activities or artifacts. To offer more user support we already introduced a first extension of this framework in [RBW14] where we added activities to determine optimization objectives and to get an evaluation of deployment alternatives with respect to a certain set of evaluation parameters.

In this work, we add another activity named *Select Appropriate Solution* to be able to offer wizards that support the system engineer in the process of deciding to which degree the results from the application of a DSE method satisfy his needs and to choose his favored solution. This activity utilizes of the results of the previously added evaluation activity. Figure 2 shows this extended DSE framework. As we will apply SPEM for implementing the DSE process the figure already contains the SPEM elements we will use later in the implementation section. Next to activities these are mainly input and output artifacts, which are denoted as work products in SPEM, and tools. For the sake of clarity, some elements of SPEM are not shown explicitly in this figure as for example tasks and roles. Thus, the relations from tools to tasks are depicted as relations to the respective activities.

In the concept of the architecture wizard a distinction is made between process wizards and task wizards. A *Process Wizard* is an assistant for a whole process consisting of several activities. In our approach the architecture wizard has the role of the process wizard to guide the user through the DSE process. A *Task Wizard* is an assistant to execute a single or several tasks of an activity in the process. Based on the DSE-Framework, the architecture wizard assists the system engineer in the process of exploring various design alternatives by deploying software components to hardware resources and deciding which alternative fits best to his needs. So, the system engineer uses the architecture wizard as a process wizard to traverse through the activities of the DSE process. If the engineer needs more support within the tasks of specific activities, he may use task wizards that are integrated into the architecture wizard. Task wizards support the user to perform tasks by traversing

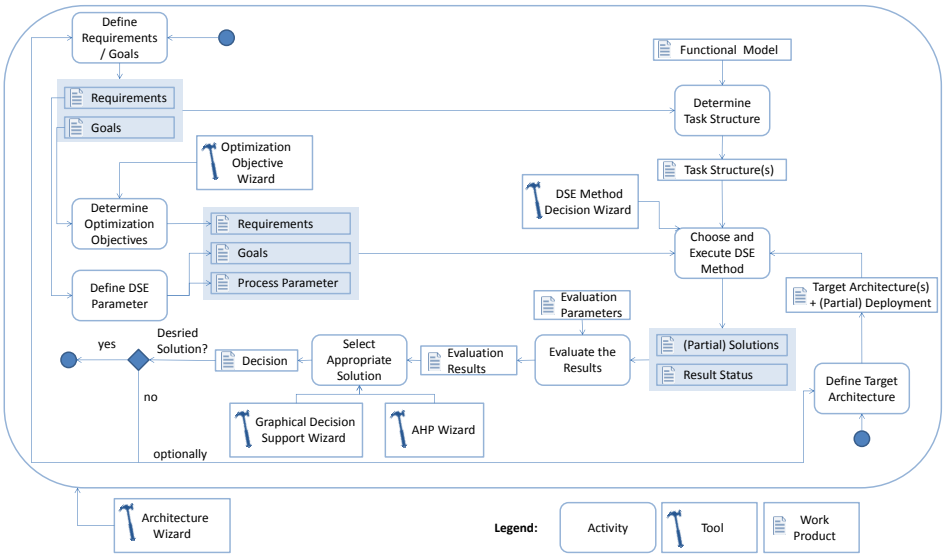

Figure 2: DSE framework with wizards in DSE process.

all needed steps of the task of an activity. In Figure 2 an overview of all available wizards is given including their relations to the respective activities.

The goals from Section 1 are addressed by the concept of the architecture wizard as follows: To address (1) the *Optimization Objective Wizard* may be used to determine the importance of optimization objectives with respect to the customer goals. Goal (2) is addressed firstly by the architecture wizard itself, which guides the user through the proposed iterative DSE process allowing to apply different DSE methods repeatedly and to refine input parameters. Secondly, the *DSE Method Decision Wizard* supports the user in choosing the best suited DSE method in each iteration. To achieving goal (3) the task wizards we propose the *Graphical Decision Support Wizard* and the *AHP Wizard* to visualize results and support the user in choosing his favored solution.

4 Architecture Wizard

Based on SPEM and the concepts of Section 3, we define an adapted meta-model containing all the elements required to realize the architecture wizard as shown in Figure 3.

In addition to the original SPEM elements described in Section 2, we use *Tool* as an adaption of the SPEM elements *Tool Definition* and *Tool Mentor* to represent wizards and other related tools of the DSE process (e. g. the optimization tools themselves). *Tool Definition* can be used to specify a tool's participation in a task and *Tool Mentor* describes the usage of the tool. A tool supports exactly one task while for the same task a set of different tools and wizards may be offered. The implementation of the architecture wizard is realized with *Eclipse*. We use the *Eclipse Process Framework (EPF) Composer* as the technical

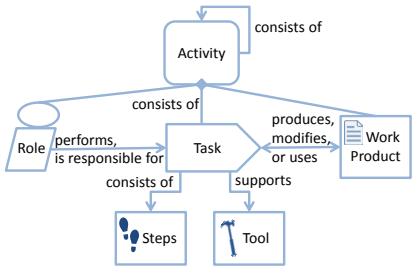

Figure 3: Adapted process meta-model for DSE process.

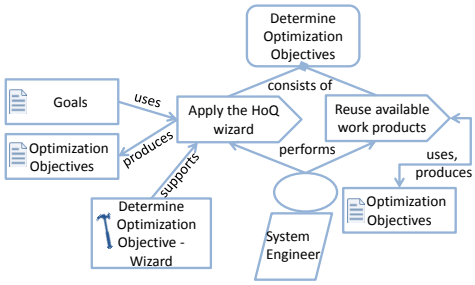

Figure 4: Instance of an activity of the DSE process meta-model.

infrastructure for implementing our DSE process with SPEM. EPF Composer is "[...] a tool platform for process engineers, project leads, project and program managers who are responsible for maintaining and implementing processes for development organizations or individual projects."[2] The implementation of this wizard is based on the *Cheat Sheets*[3] concept of Eclipse to guide the user quickly and easily through the DSE process. Cheat sheets are mini-tutorials, which consist of short instructions for multi-step processes in Eclipse.

In Figure 4,an instance of an activity of the DSE process meta-model is displayed. The activity *Determine Optimization Objectives* consists of a role *System Engineer*, two tasks *Apply the HoQ method*, and *Reuse available work products*. The task *Apply the HoQ method* uses *Goals* as an input work product. The latter task *Reuse available work products* uses a work product, which could be a text document with *Optimization Objectives* from previous iterations of the DSE process. Both tasks produce work products like *Optimization Objectives*. Furthermore, the task *Apply the HoQ wizard* is performed by the *Determine Optimization Objectives-Wizard* (see Section 5).

5 Task Wizards

In the following we describe the prototypical realization of task wizards that are currently available for the architecture wizard.

Optimization Objective Wizard: This wizard addresses goal (1) and realizes the House of Quality (HoQ) [HC88] method to support the user in the activity of determining the importance of optimization objectives. House of Quality is an established method in systems engineering and is usually used in the planning phase during the development of industrial products. For this purpose, we adapted this method as already described in [RBW14]. In this adaption the customer goals, which were identified in previous design phases, are set in relation to the optimization objectives that are supported by the available DSE methods. This is done by deciding for each combination of customer goal and optimization objec-

[2]EPF Composer: `https://eclipse.org/epf/general/EPFComposerOverviewPart1.pdf`
[3]Cheat Sheets: `http://bit.ly/1zcN4VA`

tive whether there is a strong, medium, weak or no relationship at all. Additionally, the customer goals are compared pairwise with respect to their importance for the customer. The result of the adapted HoQ is an absolute and a relative weighting as well as a ranking of optimization objectives. This can be used for determining weights or priorities for the optimization function that should be optimized by the DSE methods.

DSE Method Decision Wizard: The proposed DSE framework allows for the integration of several optimization methods to solve different DSE problems. The *DSE Method Decision Wizard* addresses goal (2) and asks questions to the user that should help him to choose the DSE method that fits best to his needs and is applicable to the given system model. This decision depends on the optimization objectives, constraints, and other aspects such as the scheduling properties of the hardware architecture and the overall complexity of the system. With the help of a classification of the available DSE methods with respect to those aspects a decision tree can be derived resulting in a list of questions the wizard is built upon. However, some questions may be skipped in the case that the information can be derived automatically from the system model or is available from previous process steps as e. g. the choice and importance of optimization objectives.

Graphical Decision Support Wizard: This wizard addresses goal (3) and enables to visualize the design alternatives produced by the DSE methods i. e. in different kinds of charts. By using these graphical representations the system engineer may compare different alternatives according to various optimization and evaluation criteria. The offered visualization helps to assess different alternatives in order to choose the best one among all the alternatives. Constraints can be visualized as well and it is possible to restrict the amount of solutions to those satisfying these constraints. Furthermore, the constraints may be varied for the visualization which may for example be used to get some ideas on how constraints may be adjusted in subsequent iterations to get more appropriate results. One graphical representation is the spiderweb chart as we already proposed in [WHR14]. It is commonly used in statistics to show data sets with multiple criteria. In Figure 5 a spiderweb chart of the graphical decision support wizard is depicted. This chart has three axes (weight, costs, cable length) and shows some example values for three different results in different colors. Furthermore, the constraint in terms of the maximum allowed value is shown for each axis in dark grey.

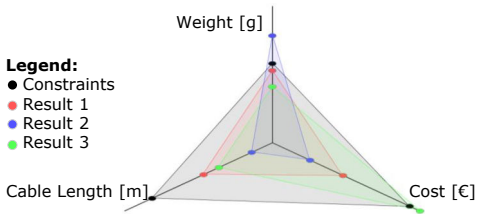

Figure 5: Spiderweb chart in the graphical decision support wizard.

AHP Wizard: AHP (Analytic Hierarchy Process) [Saa08] is a common method from the operation research area for solving qualitative multi-criteria decision problems in a structured way. The *AHP Wizard* implements this method and enables system engineers to compare and evaluate deployment alternatives to find the solution that best suits their

needs, which addresses goal (3). The method starts with the definition of the decision problem in terms of a hierarchical structure. This hierarchy consists of a goal on the top level and the available alternatives on the lowest level. All other elements on the levels in between represent the decision criteria (e. g. optimization objectives). In the second step each criterion is compared pairwise with all other criteria. Then a pairwise comparison of all design alternatives with respect to the criteria is performed. Based on the comparisons done in the previous steps, priorities are calculated for the compared elements resulting in a ranking order. After calculating the priorities, the ratings of the comparisons can be evaluated by the consistency ratio. This ratio indicates how consistent the judgments of the generated pairwise comparison are. If the consistency ratio lies above the limit of 0.1, the pairwise comparison should be restarted again. At the end all alternatives get a ranking order by all calculated priorities. The alternative with the highest priority value is the one which should be preferred.

6 Conclusion

We presented the concept of the architecture wizard as a process wizard to support the system engineer in traversing the design space exploration process to find an optimized deployment. As part of the architecture wizard task wizards may be integrated to support the user at specific activities in the process. We implemented the basic structure of the architecture wizard and a set of task wizards as prototypes. We showed how the support of the user in determining the importance of optimization objectives, performing an iterative DSE process, selecting a DSE method, and choosing a solution by visualizing design alternatives and assisting in the decision process. For the implementation of the DSE process we adapted the Software & Systems Process Engineering Meta-Model (SPEM) 2.0 to keep the architecture wizard flexible and extensible. This enables arbitrary process definitions, seamless meta-model integrations and the tool-supported generation of an implementation.

In the future we will investigate how to help the system engineer when refining the input parameters based on the evaluation results of previous DSE process iterations. Additionally, for a more concrete evidence that the architecture wizard helps the system engineer to find optimized solutions, further evaluations are planned.

References

[BDE+13] Matthias Büker, Werner Damm, Günter Ehmen, Stefan Henkler, Detlef Janssen, Ingo Stierand, and Eike Thaden. From Specification Models to Distributed Embedded Applications: A Holistic User-Guided Approach. *SAE International Journal of Passenger Cars- Electronic and Electrical Systems*, 6:194–212, May 2013.

[BHST13] Matthias Büker, Stefan Henkler, Stefanie Schlegel, and Eike Thaden. A Design Space Exploration Framework for Model-Based Software-intensive Embedded System Development. In *Workshopband Software Engineering 2013*, pages 245–249. GI–Fachbereich Softwaretechnik, GI, March 2013. ENVISION2020.

[CD08] Massimiliano Caramia and Paolo Dell'Olmo. *Multi-objective Management in Freight Logistics*, pages 11 – 36. Springer London, 2008.

[DLZV14] Philipp Diebold, Constanza Lampasona, Sergey Zverlov, and Sebastian Voss. Practitioners' and Researchers' Expectations on Design Space Exploration for Multicore Systems in the Automotive and Avionics Domains: A Survey. In *Proceedings of the EASE'14*, pages 1:1–1:10, New York, NY, USA, 2014. ACM.

[HC88] John R. Hauser and Don Clausing. *The House of Quality*. Harvard Business Review. Harvard Business Publishing, 1988.

[HHRV11] Abel Hegedus, Akos Horvath, Istvan Rath, and Daniel Varro. A Model-driven Framework for Guided Design Space Exploration. In *Proceedings of the 2011 26th IEEE/ACM International Conference on Automated Software Engineering*, ASE '11, pages 173–182, Washington, DC, USA, 2011. IEEE Computer Society.

[HJRE06] Arne Hamann, Marek Jersak, Kai Richter, and Rolf Ernst. A Framework for Modular Analysis and Exploration of Heterogeneous Embedded Systems. *Real-Time Syst.*, 33(1-3):101–137, July 2006.

[NLM+14] Himanshu Neema, Zsolt Lattmann, Patrik Meijer, James Klingler, Sandeep Neema, Ted Bapty, Janos Sztipanovits, and Gabor Karsai. Design Space Exploration and Manipulation for Cyber Physical Systems. In *IFIP First International Workshop on Design Space Exploration of Cyber-Physical Systems (IDEAL' 2014)*, Berlin, Germany, 04/2014 2014. Springer, Springer.

[Obj08] Object Management Group, Inc. Software & Systems Process Engineering Meta-Model Specification (SPEM) 2.0, 2008.

[RBW14] Maike Rosinger, Matthias Büker, and Raphael Weber. A User-Supported Approach to Determine the Importance of Optimization Criteria for Design Space Exploration. In *Proceedings of IDEAL'14 Workshop*, IFIP Springer Series. CPSWeek 2014 - IDEAL'14 Workshop, Springer, April 2014.

[Saa08] Thomas.L. Saaty. Decision making with the analytic hierarchy process. In *International Journal of Services Sciences*. Inderscience Publishers, 2008.

[SHL10] Bernhard Schätz, Florian Hölzl, and Torbjörn Lundkvist. Design-Space Exploration through Constraint-Based Model-Transformation. In Roy Sterritt, Brandon Eames, and Jonathan Sprinkle, editors, *ECBS*, pages 173–182. IEEE Computer Society, 2010.

[Tid10] Jenifer Tidwell. *Designing Interfaces: Patterns for Effective Interaction Design*. O'Reilly Series. O'Reilly Media, Incorporated, 2 edition, 2010.

[TP11a] T. Taghavi and A.D. Pimentel. An interactive visualization tool for the analysis of multi-objective embedded systems design space exploration. In *Proceedings of the 3rd Workshop on Rapid Simulation and Performance Evaluation: Methods and Tools (Rapido '11), in conjunction with HiPEAC'11*, pages pp. 1–10, 2011.

[TP11b] Toktam Taghavi and Andy D. Pimentel. Design metrics and visualization techniques for analyzing the performance of MOEAs in DSE. In Luigi Carro and Andy D. Pimentel, editors, *ICSAMOS*, pages 67–76. IEEE, 2011.

[VEH14] Sebastian Voss, Johannes Eder, and Florian Hölzl. Design Space Exploration and its Visualization in AUTOFOCUS3. In *SE (Workshops)'14*, pages 57–66, 2014.

[WHR14] Raphael Weber, Stefan Henkler, and Achim Rettberg. Multi-Objective Design Space Exploration for Cyber-Physical Systems satisfying hard Real-Time and Reliability Constraints. In *Proceedings of IDEAL'14 Workshop*, IFIP Springer Series. CPSWeek 2014 - IDEAL'14 Workshop, Springer, April 2014.