# A Design Space Exploration Framework for Model-Based Software-intensive Embedded System Development

Matthias Büker, Stefan Henkler, Stefanie Schlegel, Eike Thaden

bueker@offis.de, henkler@offis.de, schlegel@offis.de, thaden@offis.de

**Abstract:** We propose an abstract framework for Design Space Exploration (DSE) in the context of model-based embedded system development. The goal is to enable the integration of a set of concrete DSE methods addressing different system characteristics and design goals while still having a common understanding of design artifacts.

## 1 Introduction

For the model-based design of complex embedded systems a structured and well-defined engineering process is essential to guarantee high quality products that fulfill all requirements of the stakeholders. In the project *Software Plattform Embedded Systems* (SPES) 2020[1] the *SPES Matrix* was proposed as part of the SPES Modeling Framework, which enables to handle the complexity of such systems by introducing different viewpoints and abstraction layers to model the system under development [Po12, Chap. 3]. An important step to obtain a system implementation is the *Design Space Exploration* (DSE), where design decisions are taken based on goals and requirements defined in previous phases.

We propose an abstract DSE process framework that is embedded into the SPES Matrix and aims at allocating a functional model to a distributed hardware architecture. The set of valid solutions may be restricted by constraints, which could be derived from previously defined requirements considering e.g. aspects like safety and real-time. Valid solutions are rated with respect to defined goals to facilitate a decision which candidate should be chosen as desired system implementation. Based on this abstract framework, concrete DSE methods can be implemented addressing different kinds of DSE problems that are relevant in industrial practice. This allows the user to choose from a set of methods to tackle a DSE problem where all methods are compatible to each other because they have a common understanding of system artifacts and their relations. This enables, in contrast to existing DSE approaches, to define a seamless Design Space Exploration methodology.

Some examples for concrete methods we plan to integrate are the following: In [SHL10] a DSE process is proposed based on formalized model transformation rules. These rules are derived from design constraints limiting the possible solution space and allow a mechanized exploration of possible design alternatives. Another work [KH08] is based on a synchronous language named *COLA* and presents a way to deploy clusters (tasks) on

[1]Funded by the Ministry of Education and Research (BMBF), http://spes2020.informatik.tu-muenchen.de/

a multi-processor platform. In [Bü11a; Bü11b], we proposed a concrete DSE method addressing the allocation of automotive software-applications on hierarchical distributed hardware architecture.

In the next section, we give an overview of the relevant concepts of the SPES Matrix. In Section 3, we introduce the proposed abstract Design Space Exploration Framework. Finally, a short summary of the paper is given in Section 4.

## 2 The SPES Matrix

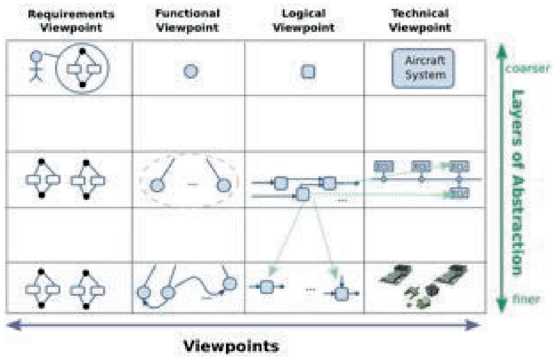

Figure 1: Matrix of Abstraction Layers and Viewpoints

To handle the complexity of embedded systems development it is useful to focus on certain aspects of the system in different development phases. In the project SPES2020, two concepts to reach this goal were introduced namely *abstraction layers* and *viewpoints* as depicted in Figure 1. Abstraction layers allow to model system artifacts on different granularity and level of detail. Viewpoints regard the same system from different perspectives while focusing on certain aspects as functional or technical concerns. In SPES2020 the following four different viewpoints were defined.

**Requirements Viewpoint:** To support the requirements engineering process the *Requirements Viewpoint* is introduced. It is intended to cover the relations of the system under development and its context such as users, stakeholders and external systems. To achieve this, goals and requirements are derived that the system under development should satisfy.

**Functional Viewpoint:** In the *Functional Viewpoint* the functions, the system under development should offer, are modeled and its syntactical interface is specified. Typically, functions arise from the requirements and goals defined on the Requirements Viewpoint. These functions may be decomposed into sub-functions and their behavior may be specified as well as their interaction with other functions or sub-functions.

**Logical Viewpoint:** The *Logical Viewpoint* is intended to specify a structural decomposition of the system under development into logical components independently from any technological aspects. For this purpose, the functions identified on the Functional Viewpoint are mapped to logical components by means of trace links. The interaction of logical components is modeled by logical channels and an interface is defined, which the system offers to communicate with its context. Nevertheless, the Logical Viewpoint may still model the functionality of the system but now realized in a logical architecture.

**Technical Viewpoint:** The *Technical Viewpoint* describes the physical architecture of the system in terms of hardware resources in combination with its software parts in terms of tasks. By means of trace links a deployment mapping is specified defining on which hardware resources tasks of the Logical Viewpoint are executed and how logical subsystems are realized. Important aspects that are treated on this viewpoint are the specification and verification of timing and safety properties such as resource consumption and redundancy.

## 3 Design Space Exploration Framework

In the context of this work, *Design Space* refers to all configurations of a given embedded hardware/software system satisfying a set of requirements and constraints. *Design Space Exploration* refers to the process of systematically searching a Design Space for (near-) optimal solutions with respect to one or more given optimization objectives. In Figure 2 the proposed framework for DSE is shown and described in the following.

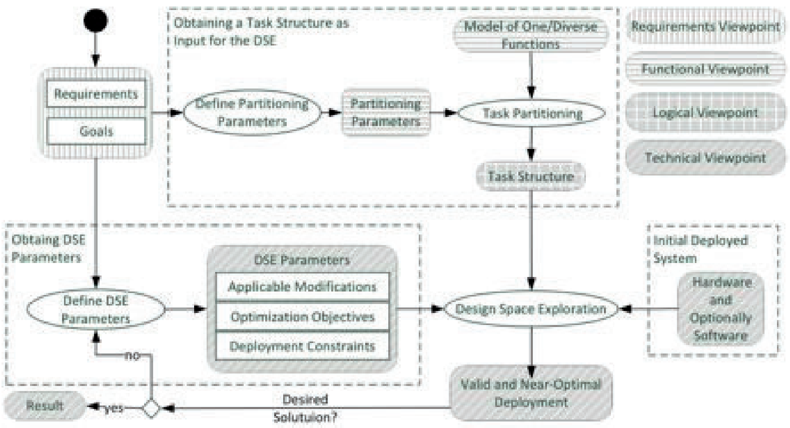

Figure 2: Overview of Design Space Exploration Framework

### 3.1 Defining the Design Space

**Initial Deployed System:** Developing embedded systems is typically done incrementally. This means that systems are built on top of an existing predecessor model, like for example a model line of a car. This is what we refer to as *Initial Deployed System*. Such a system is built of Electronic Control Units (ECUs) with some already allocated tasks. The overall capacity of all ECUs, less the needed capacity for the already allocated tasks, is the available resource for task allocation. At this point, we explicitly differ between software and hardware parts of the system, which is a characteristic of the Technical Viewpoint.

**Task Structure:** New functions usually are developed with the help of some design tool, like for example Matlab Simulink. In such tools the system is modeled primary with respect to functional aspects. That is why such a model is related to the Functional Viewpoint. To be able to allocate functions to ECUs we need to define atomic processes (tasks) that are to be executed on hardware resources. Thus, the functional model is partitioned into tasks under logical aspects, which are typically different from the functional decomposition. Such logical aspects might be, for example, communication dependencies between functions or safety constraints. The result of this *Task Partitioning* process is a logical task structure, which serves as input for the succeeding DSE step.

**The Influence of the Requirement Viewpoint:** In the Requirements Viewpoint goals and requirements are specified, which are refined to constraints and objectives relevant for DSE. On the one hand, these might be constraints that influence the Task Partitioning process in terms of *Partitioning Parameters*. Such a constraint might, for example, forbid that two functions are mapped to the same task. An example for an objective is that communication between tasks should be minimized.

On the other hand, the Requirements Viewpoint also influences the DSE itself via *DSE Parameters* in terms of constraints, objectives, and applicable modifications. An example for a relevant goal might be that the system costs should be low leading to the objective to minimize costs. A safety requirement, for example, might lead to a constraint stating that some tasks are not allowed to get allocated to the same ECU. Modifications allow modifying the hardware architecture by adding new ECUs or exchanging existing ones.

## 3.2 Design Space Exploration

In the DSE process the given task structure should be allocated to the Initial System while considering all applicable modifications, satisfying all constraints and optimizing the given objectives. Because of the high complexity of the optimization problem often heuristic methods are used to search the Design Space leading to near-optimal solutions.

The described abstract DSE framework may be realized by different concrete DSE methods that are refinements of the abstract process. Refinement means here, for example, that the DSE step may be detailed in terms of different phases describing a concrete DSE method. Furthermore, the DSE Parameters need to be concretized by describing what kind of DSE Parameters are supported by a certain method and how they are extracted from the Requirement Viewpoint. To be able to rate solutions with respect to an optimization goal, this goal needs to be formalized to an optimization function.

### 3.3 DSE Results and Iterations

For a solution to be valid all tasks have to be allocated and yield in a feasible system whereas all given constraints are satisfied. In this case complete solutions exist, and the DSE returns the best solutions for the optimal deployment problem with respect to the optimizing goal. Depending on the optimization method such solutions might be optimal or near optimal. If there are several (near-) optimal solutions the user can decide whether one of them fits his needs. If this is the case, the process is finished. Otherwise, the DSE Parameters may be adjusted to guide the process into the desired direction. If no valid solution exists, the DSE returns an incomplete result whereas some tasks got allocated. In this case it is possible to modify the DSE Parameters and run the DSE step again to allocate the remaining tasks and to obtain a valid solution. Changes of the DSE Parameters may be valid refinements of the linked requirements but they might also be changes that have influences to the Requirement Viewpoint. Thus, it might be necessary to adapt requirements such that the refinement relation is restored. Based on these changes the DSE may be executed again in another iteration to obtain further variants of the deployed system.

## 4 Conclusion

We presented an abstract DSE framework embedded into the concepts of viewpoints and abstraction layers of the SPES Matrix. This DSE framework allows the integration of a set of concrete DSE methods with well-defined interfaces to address different DSE problem characteristics.

## References

[Bü11a] Büker, M. et al.: An Automated Semantic-Based Approach for Creating Tasks from Matlab Simulink Models. In: Proc. of 16th International Workshop on Formal Methods for Industrial Critical Systems (FMICS). 2011.

[Bü11b] Büker, M. et al.: Automating the Design Flow for Distributed Embedded Automotive Applications: Keeping Your Time Promises, and Optimizing Costs, too. In: Proc. International Symposium on Industrial Embedded Systems (SIES). 2011.

[KH08] Kugele, S.; Haberl, W.: Mapping Data-Flow Dependencies onto Distributed Embedded Systems. In: Proc. of SERP 2008. 2008.

[Po12] Pohl, K. et al.: Model-Based Engineering of Embedded Systems: The SPES 2020 Methodology. In: Berlin Heidelberg: Springer, 2012. ISBN: 978-3-642-34613-2.

[SHL10] Schätz, B.; Hölzl, F.; Lundkvist, T.: Design-Space Exploration through Constraint-Based Model-Transformation. In: Proceedings of the 2010 17th IEEE International Conference and Workshops on the Engineering of Computer-Based Systems. ECBS '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 173–182. ISBN: 978-0-7695-4005-4.