Design Space Exploration and its Visualization in AUTOFOCUS3

Sebastian Voss, Johannes Eder, Florian Hölzl

Software and Systems Engineering
fortiss GmbH
Guerickestr. 25, 80805 Munich, Germany
{voss, eder, hoelzl}@fortiss.org

Abstract: Software-intensive embedded systems are characterized by an increasing number of features that implement complex safety-critical functionalities. These systems are more and more developed in a model-based fashion that has been considered as a central design approach to deal with the increase in software complexity. These kinds of embedded systems always require multiple constraints both functional and non-functional ones.

AUTOFOCUS3 is a model-based development framework using tightly integrated models that enable to perform design space exploration for multi-criteria problems. Finding suitable deployments, meaning the (efficient) assignment of software components to hardware components, is one of these problems. This paper illustrates how such a Design Space Exploration approach in a model-based framework can support the system designer in a (semi-) automatic way, enabling to compare different valid design solutions, w.r.t. a set of given system requirements. We propose a visualization technique to efficiently guide the system designer through such a calculated solutions space. The presented approach has been implemented in the AutoFOCUS3 framework.

1 Introduction

Software-intensive embedded systems – like automotive vehicles – are characterized by an increasing number of complex features. These new functionalities perform more and more safety-critical functions. Therefore, the design of such systems becomes a design space exploration problem using for multi-criteria system requirements. This increases the challenge on assuring that configurations for such systems fulfill given system requirements, e.g. timing requirements but also that their construction achieve acceptable levels of safety.

Model-based development is becoming state-of-practice in domains like automotive or aeronautic. Different tools (e.g., Simulink or ASCET) can be used that facilitate such an development via abstract component-models of the system and support the system designer with different levels of abstractions and supporting views.

System designers have to take all given system requirements into account, while facing more and more complex systems. Finding valid solutions in such an ever-increasing design space becomes a manually unsolvable task. Therefore, tool support seams necessary to solve these problems, namely finding valid system configurations. Usually – in the industry – various tools (tool-chains) are used in a system design. Each of these tools provide a reasoning of a certain aspects, e.g. calculating timing/scheduling guarantees of a certain system configuration. However, a friction loss can be expected when using such tool-chains in a design process due to incomplete tool interfaces. This leads to a situation that, when changing certain system parameter (e.g. a different allocation of software parts to hardware resources), all effects of such a decision are not traceable anymore. The whole design process need to be re-run again. Thus, different valid design solutions are not comparable directly.

Therefore, in this paper, we illustrate how such a Design Space Exploration approach in a model-based framework can support the system designer in a (semi-) automatic way, enabling to compare different design solutions. We propose a visualization technique to efficiently guide the system designer through various solutions. The paper is based on the fact that different constraints problems (e.g. timing constraints, efficient deployment of software components to hardware components and safety constraints) can be solved jointly in an integrated model-based design framework. When formalizing these design constraints, a design space exploration (technique) is used to guide the exploration of possible design alternatives. In short, the contribution of this submission is twofold:

1. The provision of a Design Space Exploration approach in an integrated model-based development tool
2. A visualization technique supporting the system designer to enable the evaluation of intermediate solutions and/or defining further system constraints based on these solutions

We use the AUTOFOCUS3 framework (`http://af3.fortiss.org`) as a model-based tool to seamlessly specify embedded systems using different layers of abstraction, while supporting different views on the system model. AUTOFOCUS3 allows modeling and validating concurrent, reactive, distributed, timed systems on the basis of a formal synchronous reactive discrete time semantics.

2 Background

The development process of embedded systems may be characterized by a sequence of refinement steps. Each design step - in general - involve decisions made by the system designer. These decisions are based on design constraints that limit the set of valid solutions in the design space and may reflect different kind of system objectives or requirements [SHL10].

Design Space Exploration is the development activity of exploring design alternatives for multi-criteria problems. A technique to guide a solver through the design space efficiently

has been proposed in [KJS11]. Performance evaluation of embedded system based on building blocks (e.g. design evaluation, search strategies and design representation) for a design space exploration framework is intensively discussed in [Kue06].

The usability of SAT solvers for scheduling synthesis of distributed system has been presented by [MFHS05]. Steiner [Ste10] provides a formal specification of scheduling constraints for time-triggered multi-hop networks and demonstrates the appropriateness of SMT-solvers for scheduling synthesis and verification. In [VS13], we describe a deployment and scheduling synthesis approach using SMT-Solver constraints for multi-core architectures using shared-memory architectures.

Model-based design of embedded systems allows to integrate more constraints, e.g. safety-related constraints into the optimization problem [VSKC13], through an integrated model. These safety constraints may be relevant to the deployment and scheduling synthesis problems as they constrain, for instance, certain mappings of software components to hardware components.

Safety constraints can be derived from safety standards, e.g. ISO 26262 [iso11] is the road vehicles specific instantiation of the electrical and electronics systems safety meta-standard IEC 61508 [Gal08].

3 The AUTOFOCUS3 Framework

AUTOFOCUS3 (`http://af3.fortiss.org`) is a research CASE tool that allows modeling and validating concurrent, reactive, distributed, timed systems on the basis of a formal semantics. It provides a graphic user interface supporting the specification of embedded systems in different layers of abstraction while supporting different views on the system model (e.g. from the model-based requirements view down to the hardware-related platform view).

AUTOFOCUS3 uses a message-based, discrete-time communication scheme as its core semantic model. Systems are model using networks of components communicating via messages. Messages are exchanged synchronously with respect to a global, discrete time base. This computational model supports a high degree of modularity. The discrete time base abstracts from implementation details such as detailed timing or communication mechanisms. The communication model allows for both periodic and sporadic communication behavior. Furthermore, AUTOFOCUS3 provides different component semantics with respect to causality of observations: the notion of *strong* and *weak* causality [VS13]. This corresponds to the perfect synchrony hypothesis – e.g., used in Simulink – where the current output of a time step depends on the current input. However, cyclic communication is forbidden in AUTOFOCUS3.

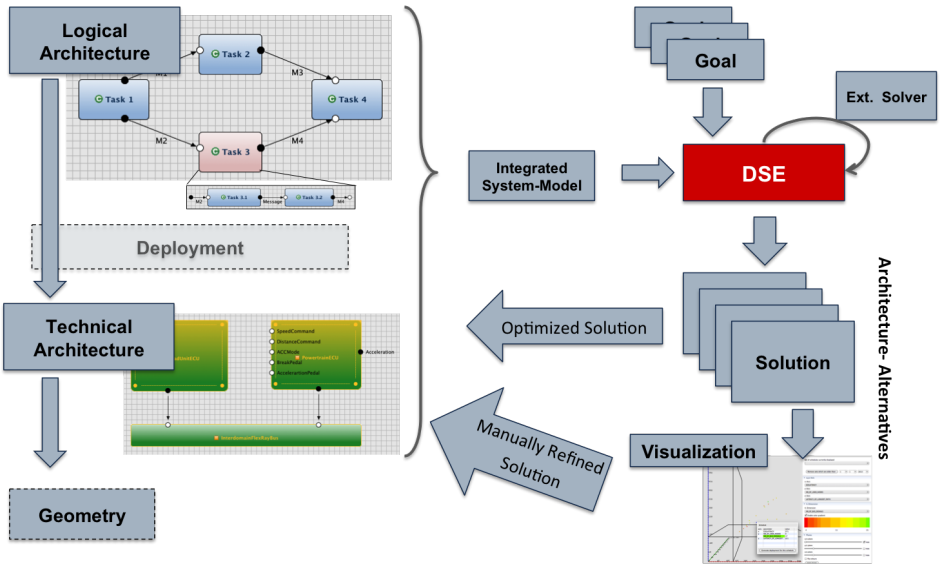

Abbildung 1: AutoFOCUS3 Layers of Abstraction and Process of System Synthesis

3.1 Different Levels of Abstraction

An AUTOFOCUS3 system model is divided into several design viewpoints that provide different levels of abstractions. For the synthesis problems presented in Sec. 4 the *logical view* and *technical view* are the most relevant (left-hand side of Fig. 1).

The *Requirements Specification and Analysis View* of AUTOFOCUS3 provides for requirements specification, documentation, and analysis of the requirements of a system (not shown in Fig. 1). The *Logical Architecture* of a system is defined by means of logical components communicating via channels. Each component exposes defined input and output interfaces (i.e., sets of typed input/output ports) to its environment, either to other components or to the system environment, describing the functionality of a (sub-)system.

The *Technical Architecture* describes a hardware topology that is composed of hardware units, e.g electrical control units (ECUs), hardware ports (sensors and actuators) and busses (left-hand side of Fig. 1). Fig. 1 also illustrates such a hardware topology, with two ECUs using a couple of sensors and actuators and a interconnected a single bus. Furthermore, AUTOFOCUS3 supports the specification of inter-level allocation between different models. The deployment view is one example of such an allocation by mapping elements from the *logical* to elements of the *technical architecture*. This provides traceability between models artefacts.

AUTOFOCUS3 proposes a model-based development process using tightly integrated models for the related system levels of abstraction. These tightly integrated models are useful for extracting essential information (e.g. hardware related properties or safety-oriented

software-mapping constraints) needed as constraints for the synthesis problem under consideration. AUTOFOCUS3 provides the capabilities to directly inter-related them (i.e., Safety Requirement x to Component y).

In Section 4.3 we show how - based on integrated model-based approaches - design decisions (e.g. constraints defined by a system designer) can be used for a next design step of the construction of such deployments.

3.2 Solving Synthesis Problems in AUTOFOCUS3

In the following, we describe the general process (as illustrated in figure 1) and the different constraints that are investigated AUTOFOCUS3. This serves as a basis for the proposed (semi-) automatic synthesis approach, enabling to compare different design solutions. We propose a visualization technique to efficiently guide the system designer through these solutions.

Based on the **Logical-** and **Technical Architecture**, we extract an **Integrated System Model** (ISM) that is used as a basis for defined synthesis problems. A set of **Goal**s are described, mostly formalized system constraints (e.g. timing constraints, deployment constraints, etc.).

The definition and process of **Design Space Exploration** (DSE) is described in section 4, that systematically finds a solution from the set of possible designs. Our approach includes multi-parameter problems that are jointly solved by the use of an external state-of-the art SMT-Solver. The **Solution**s or results generated by the solver are interpreted and directly transferred back to our system models. Furthermore, we propose to include the system designer by enabling to compare different various solutions, by even changing certain parameters (compare section 4.3).

An Integrated (Intermediate) System Model ISM is gathered using information from the logical architecture as well as the technical architecture of AUTOFOCUS3models. Based on these models, we are able to extract the ISM system model that is used for *Design Space Exploration*. The logical architecture provides a set of components that, in general, corresponds to a set of tasks T, and a set of channels that corresponds to a set of messages M. The technical architecture provides the hardware resources N (e.g. number of control units (nodes), busses, ...).

The dependency of tasks is described by a precedence relation defining the execution ordering and is represented as a directed labeled graph, called *precedence graph* $\mathcal{G} = \{T, E\}$, where $E \subseteq T \times M \times T$ represents the dependencies between these tasks via the exchanged messages. (cf. figure 1). Furthermore, we define:

1. **Dependencies** among components can be described using two different functions: $\tau : T \rightarrow 2^M$ such that $\tau(t) = \{m \mid \exists t'.(t, m, t') \in E\}$, and $\rho : M \rightarrow T$ such that $\rho(m) = t'$ for $(t, m, t') \in E$, where τ describes the set of messages $m \in M$ triggered by a task t, and ρ describes for each message $m \in M$ the corresponding receiving task $t' \in T$.

2. **Deployment/Allocation** is defined by $\eta : N \rightarrow 2^T$. This function assigns to every hardware resources N a set of tasks running on it.

Besides the precedence relations, each task, message and node may have a set of additional attributes needed, e.g. Safety Integrity Levels (SIL). These attributes are defined a priori and can be described as follows: For instance: $\models t_{i,sil}$ = t , where e.g. $t_{i,sil}$ comprises the annotated safety information (Safety Integrity Levels (SIL) [iso11]) that is a constraint in the synthesis problem.

It is important to state that the AUTOFOCUS3 semantics, as described previously, are defining timing constraints used for the scheduling synthesis.

4 Design Space Exploration in AUTOFOCUS3

The development process for software-intensive systems is characterized by a sequence of refinement steps [SHL10]. Each design step - in general - involve decisions made by the system designer. These decisions are based on design constraints/goals that limit the set of valid solutions in the design space and may reflect different kind of system objectives or requirements. Typically, these design constraints can be formalized and a design space exploration (technique) is used to guide the exploration of possible design alternatives.

In this paper, we focus on a set of system constraints defining the multi-criteria problem to be solved using our proposed DSE approach:

1. **Deployment** comprises an (optimized) assignment of components to computation resources, as well as channels to buses. This assignment fulfills constraints w.r.t. computation and communication load. Furthermore, as part of the solution, a suitable *Deployment* fulfills all *Timing* and *Safety* requirements.

2. **Timing** refers to the (optimized) order of components concurrently executed on separate execution units as well as their communicating over a shared communication. Therefore, a configuration (schedule) is needed to guarantee functional and non-functional system requirements.

3. **Safety** refers to so called *Safety Integrity Levels (SIL)* [iso11] that can be applied to components and computing resources. To ensure a safe and efficient treatment of mixed-criticality systems, deployments are generated that respect Safety Integrity Levels while optimizing usage of resources.

4.1 DSE Constraints

In the following, we describe how these constraints are solved efficiently using state-of-the-art SMT-Solvers integrated into the AUTOFOCUS3 framework and how the system designer is integrated in a (semi-)automatic design space exploration approach to find for

(optimized) system design solutions (section 4.3). In the next subsections, we focus on a joint generation of deployments and schedules fulfilling given timing-, deployment- and safety-constraints.

4.1.1 Timing - Constraints

For such a system composed of a number of tasks concurrently executed on separate hardware resources and communicating over a shared communication ressource. A configuration is needed to guarantee functional and non-functional system requirements. Such a configuration can be obtained by a scheduling policy that provides a suitable off-line schedule, namely an execution ordering of task and messages, corresponding to a given deployment, i.e., the allocation of tasks to cores. Some of the constraints necessary are described in the following: The goal is to synthesize a schedule, where all precedence relations defined in $\tau(t_{send}) = \{m_i\}$ and $\rho(m_i) = \{t_{rec}\}$ are met. The semantics of AUTOFOCUS3 are important, meaning the causality of a task under consideration. Therefore, a task (t_{send}) derived from a weak-causal component should meet the following timing constraints:

$\models$ `(`m_i`.start_time =`t_{send}`.complete_time)`$\wedge$`(`m_i`.complete_time`$\leq$$t_{rec}$`.start_time)`

, where message $m_i \in M$ and $t_{send}, t_{rec} \in T$. In case a sender task (t_{send}) is derived from a strong-causal AUTOFOCUS3 component this semantic intends a different behavior: The complete time of the sender task ($t_{send} \in T$) should be greater or equal to the start time of the message (m_i): $\models$ `(`m_i`.start_time >=` t_{send}`.complete_time)`. In [VS13], we describe how this can be used for multi-core architectures using shared-memory architectures.

4.1.2 Deployment - Constraints

Another system design criterion are deployment constraints, meaning the (efficient) allocation of elements of the logical architecture to technical architecture elements. Some software functions, resp. tasks ($t \in T$) may have a pre-defined allocation constraint to a certain hardware resources ($n \in N$), for various kinds of reasons (e.g. safety, vendor-specific, ...). These *Allocation*-Constraints may cover a subset of task in the taskset T and can be easily specified as: $\models$ `(`t_i`.allocated_node =` n_j`)` , where node $n_j \in N$ and $t_i \in T$. The remaining allocation of tasks to nodes is been generated by the solver, fulfilling other given system constraints (e.g. safety, cpu_capacity of a node, ...). Furthermore, there may be deployment constraints, w.r.t. dis-location of certain software components (resp. tasks $t \in T$) to hardware resources (resp. nodes $n \in N$). These *Dis-Locality*-Constraints are specified as: $\models$ `(`t_i`.allocated_node !=` n_j`)`

4.1.3 Safety-Constraints

Furthermore, embedded systems are often safety-critical, meaning different parts of the application may have different levels of criticality. Current standards like the IEC 61508 or derived standards like the ISO 26262 require a separation of individual parts of an application with different levels of criticality. This can be ensured by assigning levels of

criticality – called Safety Integrity Levels (SIL) – to application tasks and computing resources, and avoiding the allocation of higher-level tasks to lower-level resources during deployment. To ensure a safe and efficient treatment of mixed-criticality systems deployments are generated that respect criticality-levels while optimizing usage of resources. For each $t_{i,sil} \in T$ and $n_{j,sil} \in N$, safety constraints can be easily formulated as: $t_{i,sil} \leq n_{j,sil}$, where node $n_{j,sil} \in N$ corresponds the node-specific SIL level and $t_{i,sil} \in T$ corresponds to the task-specific SIL level.

4.2 Satisfied Solution Model

The Design Space Exploration approach relies on a symbolic encoding scheme, based on an integrated system model ISM that is derived from the system architecture. We use state-of-the-art SMT-Solver (e.g. Z3 [DMB08]). The function of a SMT solver is to check the satisfiability of logical formulas over one or more theories. The solution model provided by Z3 is a valid deployment for the given deployment problem under consideration. However, the SMT-solver outputs one solution that fulfills all the defined constraints, including defined *Timing*- and *Safety* - constraints. A *valid* solution, a *model*, consists of interpretations for the variables, functions and predicate symbols that makes the formula `true`. In order to obtain optimal solutions, w.r.t. optimizing certain constraints, a meta-search (e.g. binary search, branch-and-bound, ...) is used to run and guide the solver. All solutions are interpreted and transferred back into the AUTOFOCUS3 framework.

4.3 (Semi-) Automatic Design Space Exploration

Using Design Space Exploration techniques during system development involve the software engineer/designer itself. The system designer is often not just interested in an automatically synthesized solution, but even more in various solutions, to be able to compare these solutions. Especially in a complex multi-criteria system design, the ability to find more than one solution and be able to see difference in goals and constraints w.r.t. the effect on other system parameters. Therefore, we propose a visualization techniques as part of the Design Space Exploration in AUTOFOCUS3that guides the system designer through the solution space for finding a system design w.r.t. multi-criteria system requirements.

The synthesis problem can be separated into a sequence of refinement steps that - at each step - may highlight valid solutions for the problem under consideration. These solutions may be used either for further optimization or as an input for the next refinement steps, containing changed parameterized constraints (e.g. reflecting additional objectives). The visualization of such calculated results has been implement in AUTOFOCUS3 . We enable various visualization techniques. All results stored in an ISM can be displayed w.r.t. different parameters (see figure 2). Some of them are mentioned in the following:

1. **End-To-End Latency** describes the (logical) response time of the system, w.r.t. an

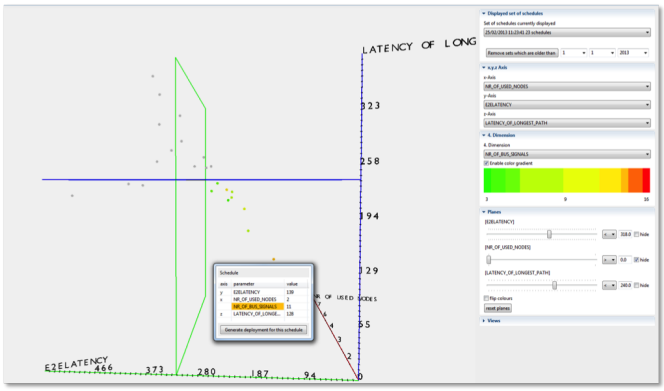

Abbildung 2: Visualization of deployments and their respective timing behavior

input from the system's environment

2. **Number of Used Nodes** specifies the number of execution units (ECUs) on which at least one task $t \in T$ is deployed on
3. **Number of Bus Signals** counts the number of Messages $m \in M$ that are send on the bus
4. **Latency of Longest Task Path** describes the (logical) duration of the longest task path in the precedence graph (compare section 3.2)

Besides these visualization parameters, safety constraints, when selected, are already part of the constraint problem, meaning that all calculated solutions fits the described safety constraints. In AUTOFOCUS3 , a three dimensional coordinate system is used, whereby the previously described parameters can be assigned to each axis. A deployment (including all timing properties) is represented as a point in this three dimensional space. Furthermore, color variations are used enable that enable a fourth dimension, depending on the needs of the system designer. Additionally, the visualization provides constraint functionality to be used by the system designer. Each of the parameters assigned to the axis in three dimensional space can be limited. These constraints are visualized using planes as shown in figure 2. These panes may be used as additional for further execution, finding for an (optimized) final solution. Thus, the visualization supports for evaluating of intermediate system solutions by efficient visualizing the possible solution of the problem under consideration.

5 Conclusion

We have demonstrated that multi-criteria problems can be efficiently solved using a Design Space Exploration approach for based on a seamless model-based development frame-

work. Furthermore, we have illustrated, that a (semi-) automatic system design approach supporting the system designer enabling for the evaluation of intermediate solutions and/or defining further system constraints based on these solutions.

Literatur

[DMB08] Leonardo De Moura und Nikolaj Bjørner. Z3: an efficient SMT solver. In *Proceedings of the Theory and practice of software, 14th international conference on Tools and algorithms for the construction and analysis of systems*, TACAS'08/ETAPS'08, Seiten 337–340, Berlin, Heidelberg, 2008. Springer-Verlag.

[Gal08] Heinz Gall. Functional safety IEC 61508 / IEC 61511 the impact to certification and the user. In *Proceedings of the 2008 IEEE/ACS International Conference on Computer Systems and Applications*, AICCSA '08, Seiten 1027–1031, Washington, DC, USA, 2008. IEEE Computer Society.

[iso11] *ISO 26262 - Road vehicles â Functional safety*. Geneva, Switzerland, 2011.

[KJS11] Eunsuk Kang, Ethan Jackson und Wolfram Schulte. An approach for effective design space exploration. In *Proceedings of the 16th Monterey conference on Foundations of computer software: modeling, development, and verification of adaptive systems*, FOCS'10, Seiten 33–54, Berlin, Heidelberg, 2011. Springer-Verlag.

[Kue06] Simon Kuenzli. *Efficient Design Space Exploration for Embedded Systems*. Dissertation, ETH Zurich, April 2006.

[KW04] Tim Kelly und Rob Weaver. The Goal Structuring Notation â A Safety Argument Notation. In *Proc. of Dependable Systems and Networks 2004 Workshop on Assurance Cases*, 2004.

[MFHS05] A. Metzner, M. Fränzle, C. Herde und I. Stierand. Scheduling Distributed Real-Time Systems by Satisfiability Checking. In *RTCSA '05*, Seiten 409–415, Washington, DC, USA, 2005. IEEE Computer Society.

[SHL10] Bernhard Schätz, Florian Hölzl und Torbjoern Lundkvist. Design-Space Exploration through Constraint-Based Model-Transformations. In *Proceedings of the 2010 17th IEEE International Conference and Workshops on the Engineering of Computer-Based Systems*, Seiten 173–192, 2010.

[Ste10] Wilfried Steiner. An Evaluation of SMT-Based Schedule Synthesis for Time-Triggered Multi-hop Networks. *Real-Time Systems Symposium, IEEE International*, 0:375–384, 2010.

[VS13] Sebastian Voss und Bernhard Schätz. Deployment and Scheduling Synthesis for Mixed-Critical Shared-Memory Applications. In *Proceedings of Engineering of Computer based Systems (ECBS) 2013*, 2013.

[VSKC13] Sebastian Voss, Bernhard Schätz, Maged Khalil und Carmen Carlan. Towards Modular Certification using Integrated Model-Based Safety Cases. In *VeriSure Workshop on 25th International Conference on Computer Aided Verification*, 2013.