Usable Design Space Exploration in AutoFOCUS3

Johannes Eder, Sebastian Voss

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

Abstract. Software-intensive embedded systems are characterized by an increasing number of features that implement complex functionalities. To effectively manage this complexity, development processes in general, and model-based approaches in particular, support the development of such systems as model-based approaches have been considered a central design approach to deal with increasing complexity in software and hardware development.
A valid system design and configuration, especially a safety-critical system design, has to fulfill a corresponding set of requirements describing all desired system constraints and objectives. In general, these constraints may be contradicting and correspond to different dimensions (e.g. timing, safety, energy, cost, etc.). Thus, considering all system constraints during system design is a manually unsolvable task. To support the system designer, usable *Design Space Exploration* methods are needed. Therefore, a proper tool implementation is needed that supports the usability.
In this paper, we describe a *Design Space Exploration* process which aims to explore the architectural design space during system design. This process has been implemented in the open source CASE tool AutoFOCUS3[1] with the focus on usability.

1 Introduction

The design space of distributed embedded systems is characterized by an ever increasing number of features, like for example in automotive vehicles. Due to this growing complexity, the process of manually exploring possible designs for these systems is getting even harder for system designers. Therefore, a design space exploration process is needed, which guides the system designer in a semi-automatic way to explore possible design solutions.

In this paper, we describe such a semi-automatic Design Space Exploration process which has been implemented in AutoFOCUS3. AutoFOCUS3 is a CASE Tool that allows for seamless, model-based development of distributed, embedded systems.

[1] http://af3.fortiss.org

The goal of our approach is to provide a usable *Design Space Exploration* (DSE) process, enabling the system designer to calculate, explore and compare different design alternatives during system design. These design alternatives satisfy all requirements (constraints) that are of interest for the system design. Naturally, such requirements are mostly contradicting. Due to this fact, a semi-automatic approach is proposed, where not only one single final design can be calculated. In fact, it enables trade-off analysis by automatic generation of possible (optimized) design alternatives, that can then be evaluated by a system designer.

2 Related work

In this section, we will shortly introduce related work concerning this paper. On the one hand AutoFOCUS3 and some related case studies and on the other hand other existing DSE frameworks.

2.1 CASE tool AutoFOCUS3

AutoFOCUS3 is a scientific CASE tool, which supports the development of component based, reactive distributed embedded systems on different levels of abstraction [1, 2]. It is based on the notion of streams and stream processing functions introduced in [3]. The tool has been used in several industrial case studies, e.g. for modeling a Siemens train automation system [4]. [5] and [6] are other examples where AutoFOCUS3 was applied in a case study.

AutoFOCUS3 also provides a design space exploration process, which will be explained in more detail in the next section.

2.2 DSE frameworks

There are a variety of widely used UML/SysML modeling tools like IBM Rhapsody, PTC Integrity, Enterprise Architect and Papyrus. Until now, they do not offer any design space exploration techniques. Research-wise however, there are a few frameworks existing, which offer DSE techniques.

Here we will mention two of them. There is e.g. the FORMULA framework [7] which uses a formal specification language to encode a model driven architecture. The Z3 SMT solver is then used to enable model synthesis, providing design alternatives.

The CoBaSA tool provides a language for expressing system constraints [8]. This tool then uses PBSAT or SAT solvers to generate optimized solution architectures, which satisfy all given constraints.

The difference from AutoFOCUS3 to the tools mentioned above is that AutoFOCUS3 has a strong focus on usability. This entails, e.g., that the tool provides a user interface which is easy to use and understand. In particular, this means that the user is only exposed to as little formalism as possible, while still being able to use the power of these methods.

3 The AutoFOCUS3 Design Space Exploration Process

A usable, tool-supported DSE process starts with a pre-defined model of the system (e.g. using SysML [11]). We consider such system models to be basically divided into several models, which represent different levels of abstractions:

Logical Architecture The logical architecture represents the functionality of the system in terms of components. These components can be either hierarchically composed or represent a behavior (like for example a state-automaton). Components can have typed input and output ports. Communication can be realized by through channels, which connect ports with each other.

Technical Architecture The technical architecture consists of all hardware parts of the system, independently from its functionality. This architecture basically represents execution units which are connected to communication units (e.g. bus), sensors and actuators.

Deployment The Deployment is the connection between logical and technical architecture. The components are deployed onto execution units, where their functionality will be executed. The communication between components, which are mapped onto different execution units, is deployed to a communication unit connecting these execution units.

Schedule A schedule adds timing information to a deployment. That means starting times and durations of components deployed on execution units and messages deployed on communication units.

Depending on the synthesis step, which will be explained in the next section, this system model may differ. A logical architecture model and a technical architecture model is needed in order to perform a deployment synthesis. For synthesizing schedules a deployment model is needed.

3.1 The DSE process

Based on these models, we propose a three fold DSE-process, as depicted in figure 1:

1. **Objective and constraint modeling**: Consideration of so called *Exploration Targets* (see also [10]), namely constraints and objectives of the corresponding design space.
2. **Model synthesis**: Based on the the system model and the *Exploration Targets*, different synthesis methods are implemented that explore the design space and return Pareto efficient solutions.

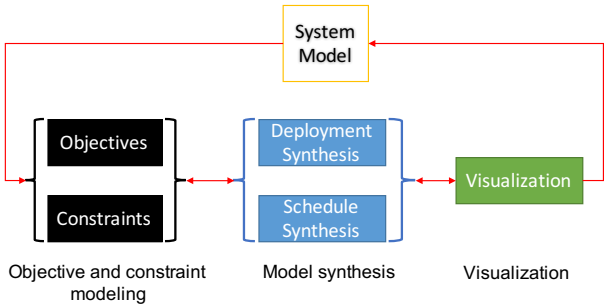

Fig. 1. The design space exploration process in AUTOFOCUS3

3. **Visualization**: Calculated results are visualized in an (interactive) *Visualization* process step using different visualization techniques to support the system designer.

In the following, we describe these process steps in detail:

3.2 Objective and constraint modeling

The definition of constraints and objectives is essential as they constrain the design space and define cost-functions in order to optimize solutions. The process of constraint modeling limits the set of possible solutions and is (often) derived from system requirements, such as safety requirements which require certain safety integrity levels (SIL) [9] (e.g. that a logical component requires a SIL of 3), or timing requirements which give certain time bounds (e.g. that the overall latency of a system should not exceed 200ms).

On the other hand, objectives describe a cost function which shall be optimized. These objectives can also be derived from requirements such as minimization of hardware costs (e.g. if it is required to have the least possible hardware costs), or of energy consumption (e.g. if the energy consumption of the whole system shall be kept as minimal as possible).

Therefore, AUTOFOCUS3 provides a domain specific modeling language which enables the formalization of such constraints and objectives. Equation 1 shows an exemplary constraint, which states that if a software component is mapped to a hardware component, the software component's SIL must not exceed the hardware components SIL (see also [9]).

$$\forall h \in HW, \forall s \in SW, sw \rightarrow hw : s.sil \leq h.sil \tag{1}$$

Equation 2 shows an exemplary objective which is a cost function over all hardware components (financial) costs, where at least one software component is deployed on. This function shall be minimized.

$$min(\forall s \in SW \sum_{\exists h \in HW : s \rightarrow h} h.cost) \tag{2}$$

To effectively support the system designer, constraints and objectives are defined on a visual basis through editors. These editors offer patterns for this language, in order to abstract the exact syntax and semantics of the language. These patterns are provided through the graphical user interface of AUTOFOCUS3 and do not require any knowledge of formal languages. Figure 2 shows how equation 2 describing an objective is modeled in AUTOFOCUS3. Constraints are modeled similarly.

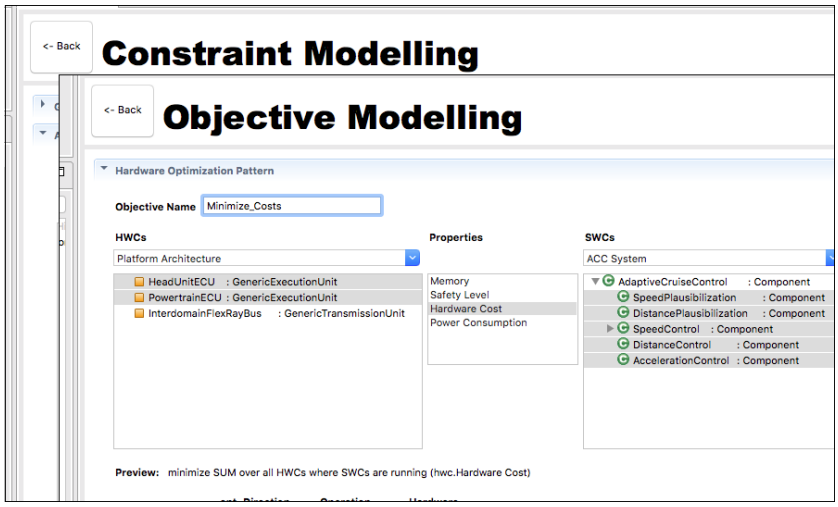

Fig. 2. Objective modeling via user interface in AUTOFOCUS3

3.3 Model synthesis

Among all constraints and objectives, sub-sets of constraints and objectives are defined, in order to explore a certain design space exploration problem. Such a *Sub-Set* categorizes constraints and objectives modeled in the previous process step. The selection of desired subsets (compare figure 3) provides the input of the design space exploration problem under consideration.

The design space exploration process in AUTOFOCUS3 provides synthesis mechanisms for deployment synthesis and schedule synthesis. We provide a symbolic encoding scheme, resp. a formalization describing the DSE problem as a satisfiability problem using boolean formulas and linear arithmetic constraints. A state-of-the-art satisfiability modulo theory (SMT) solver is used to compute design alternatives. We use the *Z3 Theorem Prover* by Microsoft [2].

If a synthesis process cannot find any solutions, the user gets notified which sub sets and which constraints in these sub sets made it impossible to synthesize

[2] https://github.com/Z3Prover

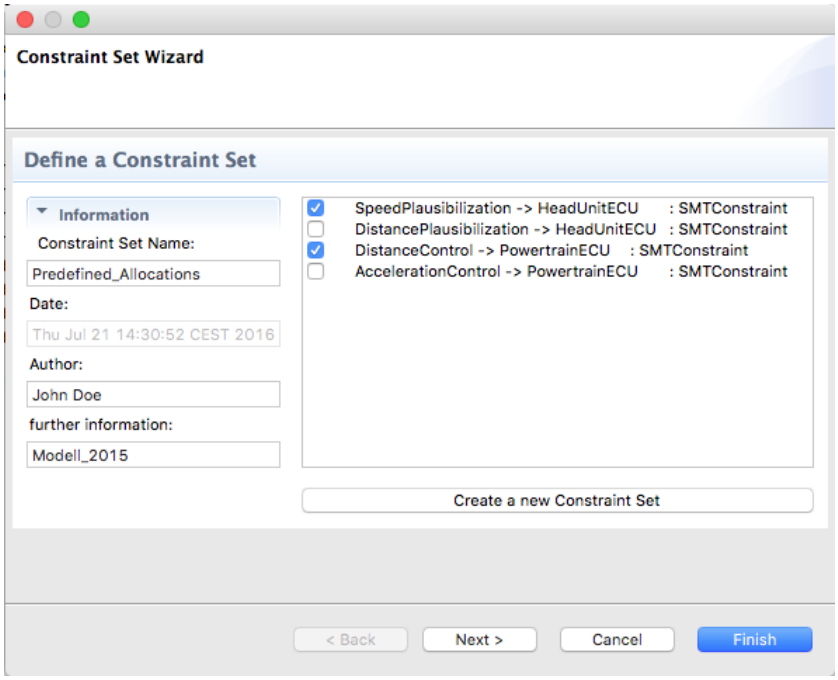

Fig. 3. Constraint sub-set definition in AUTOFOCUS3

anything. Thus it is possible to check if either the constraint itself was phrased wrong or if the requirement this constraint is derived from is wrong.

Deployment synthesis The deployment synthesis enables the exploration of the design space of deployments from logical components to technical components. Deployment means, that the functionality represented by a logical component is executed as a task, on the specific electrical control unit it is deployed on, at runtime.

Thus, this synthesis method needs, besides the constraint and objectives sub sets, models of logical component architecture and technical architecture as input. Given all these three artifacts one can explore design space of possible deployments.

Schedule synthesis The schedule synthesis enables the exploration of the timing behavior of a system. Given the input of a deployment (mapping) model (which could have been synthesized in the previous step), this mechanism synthesizes possible solutions of execution times of tasks and messages while considering given constraint and objective sub sets.

3.4 Visualization

The last DSE process step is the visualization of the calculated design alternatives. According to the given objectives in the previous synthesis step the results can be displayed in a 3D visualization or spider chart (compare figure 4), where each axis can be assigned one objective. They may also be visualized in a table where each column displays one objective. For schedules a gantt chart is provided which displays the (timed) sequence of tasks and messages.

By using these visualization techniques, a system designer is supported to select the desired solution. This solution includes information (either deployment or schedule information, depending on the performed synthesis step), that can be transferred back into the system model.

Another iteration of the DSE process can then be performed if necessary. For instance, if one wants to explore the design space of possible schedules with a recently chosen deployment.

Furthermore, it is also possible to go one or even two steps back to alter the previous steps. Either to model new constraints and objectives and/or to adapt the constraint and objective sub sets which where used for synthesis.

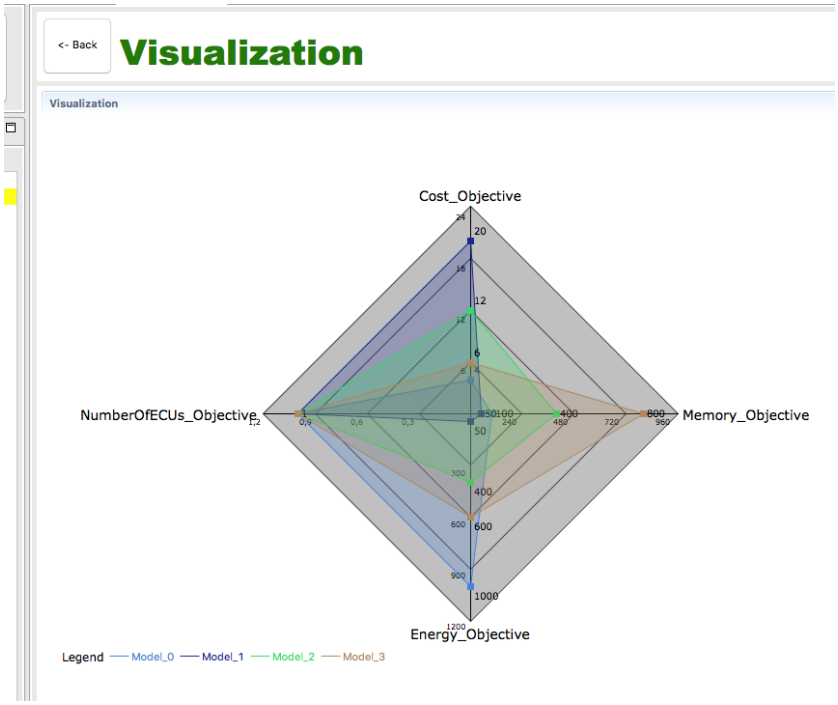

Fig. 4. Visualization of DSE results in AutoFOCUS3

4 Conclusion

In this paper, we have presented a Design Space Exploration process implemented in the CASE tool AutoFOCUS3. This process is in particular usable, due to the fact that it guides the system designer through all the required steps of a DSE. Moreover the process provides easy to use formalization methods, which do not require any knowledge in the field of formalization.

References

1. AutoFocus 3 - A scientific tool prototype for model-based development of component-based, reactive, distributed systems, Hölzl, Florian and Feilkas, Martin, Model-Based Engineering of Embedded Real-Time Systems, 317–322, 2010, Springer
2. AutoFOCUS 3: Tooling concepts for seamless, model-based development of embedded systems, Aravantinos, Vincent and Voss, Sebastian and Teufl, Sabine and Hölzl, Florian and Schätz, Bernhard, Joint proceedings of ACES-MB, p. 19, 2015
3. Specification and development of interactive systems: focus on streams, interfaces, and refinement, Broy, Manfred and Stølen, Ketil, 2012, Springer Science & Business Media
4. A formal systems engineering approach in practice: an experience report, Böhm, Wolfgang and Junker, Maximilian and Vogelsang, Andreas and Teufl, Sabine and Pinger, Ralf and Rahn, Karsten, Proceedings of the 1st International Workshop on Software Engineering Research and Industrial Practices, 34–41, 2014, ACM
5. M. Feilkas, A. Fleischmann, F. Hölzl, C. Pfaller, K. Scheidemann, M. Spichkova, D. Trachtenherz, A Top-Down Methodology for the Development of Automotive Software, Technische Universität Mnchen, Tecg. Rep. TUM-I0902, 2009
6. M. Feilkas, F. Hölzl, C. Pfaller, S. Rittmann, B. Schätz, W. Schwitzer, W. Sitou, M. Spichkova, D. Trachtenherz, A Top-Down Methodology for the Development of Automotive Software, Technische Universität Mnchen, Tecg. Rep. TUM-I1103, 2011
7. Components, platforms and possibilities: towards generic automation for MDA, Jackson, Ethan K and Kang, Eunsuk and Dahlweid, Markus and Seifert, Dirk and Santen, Thomas, Proceedings of the tenth ACM international conference on Embedded software, 39–48, 2010, ACM
8. Automating component-based system assembly, Manolios, Panagiotis and Vroon, Daron and Subramanian, Gayatri, Proceedings of the 2007 international symposium on Software testing and analysis, 61–72, 2007, ACM
9. ISO 26262 - Road vehicles - Functional safety, Geneva, Switzerland, 2011.
10. A Lightweight Design Space Exploration And Optimization Language, Diewald, Alexander and Voss, Sebastian and Barner, Simon, Proceedings of the 19th International Workshop on Software and Compilers for Embedded Systems, 190–193, 2016, ACM
11. OMG Systems Modeling Language (OMG SysML), Version 1.3, http://www.omg.org/spec/SysML/1.3/, 2012