Pareto-efficient deployment synthesis for safety-critical applications in seamless model-based development

Sergey Zverlov fortiss GmbH
Guerickestr. 25, 80805 Munich, Germany
Email: zverlov@fortiss.org
Maged Khalil fortiss GmbH
Guerickestr. 25, 80805 Munich, Germany
Email: khalil@fortiss.org
Mayank Chaudhary fortiss GmbH
Guerickestr. 25, 80805 Munich, Germany
Email: chaudhary@fortiss.org

***Abstract*—Increasingly complex functionality in automotive applications demand more and more computing power. As room for computing units in modern vehicles dwindles, centralized architectures - with larger, more powerful processing units - are the trend. With this trend, applications no longer run on dedicated hardware, but share the same computing resources with others on the centralized platform. Ascertaining efficient deployment and scheduling for co-located applications is complicated by the extra constrains which arise if some of them have a safety-critical functionality.**

Building on our pre-existing design space exploration solution, we integrated safety constraints, such as ASIL and HW failure rates, as well as practical aspects, such as component costs, and extended the approach to allow for multi-criteria optimization. The work was implemented into our seamless model-based research CASE tool AutoFOCUS3 and evaluated using a non-trivial industrial-inspired case study. The solution is capable of synthesizing deployments together with corresponding schedules, which satisfy different safety and resource constraints. The deployments can subsequently be integrated into the safety case argumentation of AutoFOCUS3, leveraging the tool's seamless capabilities to support safety evidence and certification.

Moreover, we are not interested in merely valid solutions, but in good ones. We hence developed a multi-objective optimization algorithm, which synthesizes solutions pareto-optimized for safety, resource usage, timing and any other constraints the user defines. Our approach demonstrates the feasibility and effectiveness of using formal methods to generate correct solutions for safety-critical applications, increasing the confidence and validity of safety cases.

***Index Terms*—Design Space Exploration, AutoFOCUS, Optimization, SMT, Safety, Resource Optimization, Deployment, Scheduling**

I. Introduction

Connected vehicles and advanced driver assistance systems (ADAS) are just a few of the latest technological drivers increasing the demand on computing power in classical domains of embedded systems development, such as the automotive sector. Given the already high number of ECUs (electronic control unit) in today's automobiles, there is a clear trend towards centralized architectures featuring larger, more powerful (potentially multi-core) platforms. This trend lends increased urgency to the problem of mapping logical computational tasks to the hardware (HW) nodes that will execute them. This task is intrinsically difficult in embedded systems, because their limited resources add many constraints to the deployment problem that have to be respected, e.g., execution timing, memory constraints, power consumption, bus loads and many more. If the applications to be deployed are safety-relevant, this adds a new dimension to the deployment problem, increasing the difficulty many-fold. Safety standards dictate design patterns, such as partitioning according to criticality, to maintain freedom from interference, but also set requirements for the hardware nodes' reliability. This adds multiple criteria (some orthogonal) to both the problem space (software components) and the solution space (hardware nodes). Mapping logical/software components to a hardware architecture goes beyond the simple task of mere allocation, and onto the generation of a deployment and corresponding schedule, which satisfies different constraints, be it related to safety, resource or cost.

The work presented in this paper is a part of ongoing research efforts into design space exploration (DSE), which aims at facilitating the seamless development of safety-critical applications in the embedded domain. DSE is defined as a process of systematically finding a solution from a set of possible designs, w.r.t. a set of given constraints [1]. Building on our pre-existing approach for the efficient generation of embedded system architectures with multi-criteria optimization (e.g., memory usage, power consumption, bus loads), we expanded the approach to include safety attributes, such as hardware failure metrics, but also practical aspects of real-time development, such as HW costs. The presented approach computes task and message schedules that are optimized with respect to a global discrete time base. As a part of the solution, the approach generates an optimized assignment of tasks to

computation resources (cores). The approach is integrated into the AutoFOCUS3 (AF3) tool-chain, as presented in [2].

This paper presents the approach, and demonstrates its usefulness using examples which illustrate the effect of different criteria on deployment synthesis and how the approach can be used to optimize the generated valid solution for one or more criteria. Prior to this work, there was no support for generating and optimizing deployments in AF3 while simultaneously satisfying safety metrics, resource constraints and hardware costs.

Furthermore, AF3 integrates multiple dedicated architectural views, covering not only logical (software) and technical (hardware) architectures, but also requirements and safety case expression. Therefore, the method presented here can be integrated into a seamless development approach that allows arguing evidence on a formal basis, as well as reusing argument structures and reducing the costs of component as well as design (re-)certification for safety-critical applications.

A. Related Work

Approaches where computer-aided methods are used for system analysis and optimization are not new, having been proposed as early as in the late fifties, such as the optimization approach for register transfer systems towards cost and performance presented in [3].

Recent works address higher levels of abstraction, such as the mapping of software to hardware and the corresponding scheduling [4]; memory optimization [5]; or finding suitable platform architectures [6][7][8]. In this paper we address the joint problem of finding an optimized platform, deployment and schedule w.r.t. to safety, cost, performance and resource consumption.

The applicability of solvers for finding deployments and schedules was shown in [9] and [2]. The possibility of using various solvers for DSE problems was further discussed in [10] and [11], which – similarly to our work – use a solver to find valid or optimized solution.

Furthermore there are approaches, which take safety into account. [12] – for instance – combines FTA analysis and optimization techniques to find optimal system configurations; [13] optimizes automotive architectures towards cost and safety; and [14] uses a brute-force algorithm to generate safety-compliant deployments for avionic systems. Compared to those approaches, the optimization problem we are considering is more complicated (higher number of criteria and/or degrees of freedom), and furthermore we integrate our approach into a model-based framework, which makes it more usable for practitioners.

There are a number of frameworks with Design Space Exploration capabilities, which were proposed over the years. Some of them use their own DSLs to specify DSE Problems, such as FORMULA [15] and AAOL [16]. Others use well-established formalisms, such as EAST-ADL[17], UML [18] and AADL[19]. We integrated our approach in the AutoFOCUS3 Framework [1], which is based on the FOCUS methodology [20] and was shown to be compliant to AUTOSAR [2] in [21], leveraging the framework's seamless development capabilities.

The safety metrics used in the multi criteria optimization have their foundations in the ISO26262 automotive functional safety standard, and investigations into architecture benchmarking in the ITEA2 SAFE research project[3] published in [22].

Due to the seamless nature of AF3, our approach demonstrates how an integrated model-based framework can pragmatically provide formal support for practical problems in the development and certification of safety-critical systems in compliance with the relevant standards – in our case ISO26262.

B. Structure of Paper

Chapter II gives an overview of background information building the fundamentals of this paper. In III the model-based CASE tool AutoFOCUS is presented. In this tool, we integrate our proposed approach from chapter IV. We evaluate our work using a non-trivial case-study in chapter V and finally we conclude in VI.

II. Background

This section introduces briefly some concepts necessary for understanding the rest of the paper.

A. SMT

A satisfiability (SAT) problem consists of variables represented in a formula, with the goal being to determine whether there exists an assignment of variables that makes the whole Boolean formula satisfiable [23]. Satisfiability modulo theories (SMT) generalize boolean satisfiability by adding equality reasoning, arithmetic, arrays, quantifiers and other first order theories. SMT solvers are tools for deciding the satisfiability of formulas in those theories [24], which can be used to solve various classes of problems, such as software and hardware verification; test case generation; planning; or scheduling and deployment [25, Ch. 2, p. 89ff]. Solvers may be designed to not only be capable of determining whether a formula is unsatisfiable or not but also point to the set of clauses which are not satisfiable, called unsatisfiable cores or UnSAT Cores.

B. Safety

The work presented here was carried out within the SAFE research project, which targeted the model-based development of software architectures for safety-critical applications in the automotive domain. Hence, we focus on metrics identified in the ISO26262 automotive safety standard. These include ASIL (Automotive Safety Integrity Level), as well as hardware failure metrics identified in clauses of part 5 of the standard. The proposed metrics include PMHF (probabilistic measure of random hardware failures) the maximum probability of violating a top level safety goal, shown mapped to corresponding ASIL levels in Table 1 - and FRC (failure rate classes) an evaluation of each cause of safety goal violation. These metrics are explained in more detail in [22].

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

[2] http://www.autosar.org/

[3] http://www.safe-project.eu/

ASIL Level	Probability of random hardware failures (PMHF)
D	$< 10^{-8}h^{-1}$
C	$< 10^{-7}h^{-1}$
B	$< 10^{-7}h^{-1}$
A	$< 10^{-6}h^{-1}$

TABLE I: Target values for hardware architectural metrics

C. Multi-objective Optimization

As an optimization problem, we consider the search of a good (optimized) or best (optimal) solution of a decision problem among a set of alternatives, assuming the existence of a set of objectives, according to which the quality of the alternatives can be measured [26]. Cases in which an optimization problem does not only have one objective, but several ones are called: multi-objective optimization, and these objectives are often contradictory. Therefore, in most cases of multi-criteria optimization problems, there is no single optimal solution for the problem under consideration, but a set of solutions, where each is optimal to a subset of objectives. This set of solutions is called Pareto-efficient or the *Pareto front*.

III. AUTOFOCUS3

AUTOFOCUS3 (`http://af3.fortiss.org`) is a research CASE tool that allows modeling and validating concurrent, reactive, distributed, timed systems on the basis of formal semantics [27]. Furthermore – in [21] – it was illustrated how AF3 modeling concepts fit into concepts provided by AUTOSAR.

A. Levels of Abstraction

An AF3 system model is divided into several models that provide different levels 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.

The *Requirements Specification and Analysis View* provides for requirements specification, documentation, and analysis of the requirements of a system.

The *Logical Architecture View* of a system is defined by means of components communicating via message passing through typed channels, using a clearly defined model of computation. Messages exchange is synchronized with respect to a global, discrete time base. Components can directly implement behavior or consist of other components that do so.

The *Technical Architecture View* describes a hardware topology that is composed of hardware units, e.g. CPUs of a Multi-Core Board, hardware ports (sensors or actuators), buses and a shared-memory component.

Furthermore, AF3 supports so-called integrated or weaving models, which hold information from different levels of abstraction and thereby connect them. The deployment view is one example of such an integrated model, which maps elements from the *component* to elements of the *technical architecture*. This provides traceability between modeling artifacts [1].

Another example is the AF3 support for a safety-case view, which is supported by integrated models, linking the safety case elements to all the other AF3 artifacts in support of the argument structure.

B. Design Space Exploration in AF3

A valid deployment and schedule pair for any system under consideration has to fulfill certain constraints, for instance, precedence constraints or resource utilization (two tasks can't run in parallel on the same node). These constraints can be formalized in such a way that they can be used as input for a state-of-the-art SMT solver. In this sub-section we provide a small overview of work done in [2], which we extend in the following sections.

The approach uses the integrated deployment model, which combines information from the logical and technical architecture (cf. III-A). From the logical architecture this model contains a set of logical components (which, in this context, we call tasks) $T = t_0, t_1, ..., t_n$ passing messages $M = m_0, m_1, ..., m_k$ via channels. This communication structure results in dependencies between the tasks. Those dependencies can be used to derive constraints, such as the execution order of tasks, for the schedule generation.

Additionally, this model contains information from the technical architecture, such as a set of computational resources (nodes) N, a set of buses B, and in some cases memories MEM. Nodes are used to store and execute the tasks from the logical architecture, whereas buses and memories are used to exchange messages between the tasks.

By encoding this information in a SMT solver, it is possible to synthesize a valid deployment and schedule pair, if they exist. Furthermore, it is possible to produce not only valid but also optimized solutions (cf. [2] or [28]), using a meta-search on top of the SMT solver, which manipulates constraints – max. execution time, for instance–after every SMT request.

IV. SAFETY-ORIENTED DEPLOYMENT AND SCHEDULING IN AF3

In this section we present how using a SMT solver makes it possible to synthesize safety oriented deployments, which are either valid or optimized w.r.t. to certain criteria. Furthermore we discuss how we integrated this approach in AutoFOCUS3.

A. Artefact properties in AF3

As already explained in III, AutoFOCUS3 view on the system is divided in different levels of abstraction. Each of these levels contains a set of artifacts (i.e. ECU in Technical Architecture). Using the "annotation" concept of AF3 it is possible to assign properties to each artifact (i.e. Failure Rate to an ECU). For our approach a set of safety and resource related properties were needed, which partly already existed and partly were added by us.

In context of safety we added the possibility of annotating PMHF values for ECUs and busses. This value is then

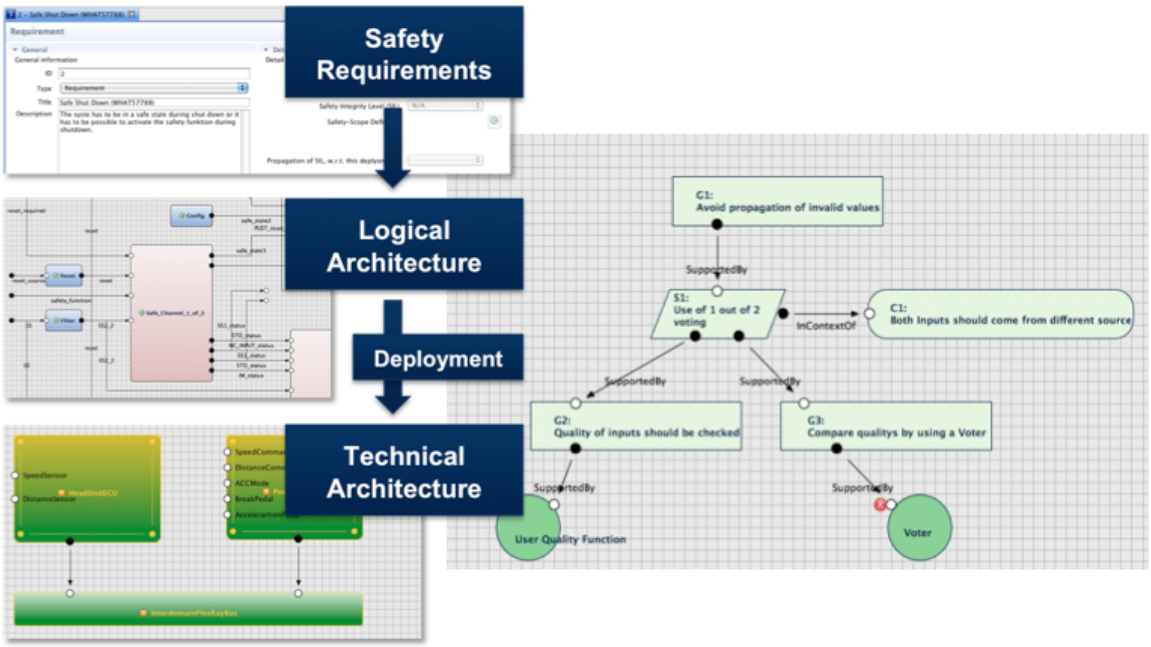

Fig. 1: Seamless model-based development in AF3

translated to ASIL according to table I. This way it is more convenient to compare with the safety integrity level values of the logical components. The possibility of assigning (A)SIL values to logical components was already given.

Furthermore we added the possibility of assigning energy consumption, memory and cost to the hardware elements as well as memory consumption and taswk duration to the logical components.

B. Scheduling and Deployment Constraints

The constraints – introduced in this sub-section – are already formalized. Together with instructions, such as in [29] or examples such as in [2], these formalized constraints provide enough information to be encoded as a SMT problem.

1) PMHF Constraint for ECU: This constraint states that a task $t \in T$ can only be mapped on a node $n \in N$, if the PMHF value of the node n is compliant to the ASIL value of the task t according to table I. Let $\Phi : T \mapsto N$ be a function that allocates tasks to nodes, then the constraint can be formalized as follows:

$$\forall t \in T', \text{where } T' = \{t' | \Phi(t') = n\} \rightarrow t.asil \leq n.asil \quad (1)$$

2) PMHF Constraint for BUS: As defined in [2], if two tasks $t_i t_k \in T$ communicate with each other and are deployed on different nodes, such that $\Phi(t_i) \neq \Phi(t_j)$, they have to communicate over a bus $b \in B$, which also has a PMHF value. Furthermore, we respect ASIL propagation, which means that tasks can only exchange messages via bus, if the bus is sufficiently reliable, as indicated by its PMHF value. This constraint affects the allocation of communciating tasks. If t_i sends a message to t_k and those tasks are mapped on different nodes, the constraint can be formalized as follows:

$$b.asil \geq t_k.asil \quad (2)$$

3) Memory Constraint: In real-life embedded systems resources, such as memory, are limited. We assume that each hardware node n offers a certain amount of memory that can be used by the tasks t allocated to it, and must not be exceeded. To define this constraint, we first have to sum up the memory which is used by each node:

$$\forall t \in T', \text{where } T' = \{t' | \Phi(t') = n\} \rightarrow n.used\_memory = \sum t.mem \quad (3)$$

Having determined the memory used by each node, we can now ascertain that the used memory does not exceed the available memory ($n.ram$):

$$n.used\_memory \leq n.ram \quad (4)$$

4) Maximum Number of Nodes: In some cases it is interesting to know whether the deployment of tasks is possible on less nodes than currently used or available. To define a constraint limiting the number of used nodes, we first have to find out which nodes are currently $used$ (has tasks deployed on it). We can formalize this as follows:

$$n.used = \begin{cases} 1, & \exists t \in T : \Phi(t) = n \\ 0, & \text{otherwise} \end{cases} \quad (5)$$

Next, we identify how many nodes are currently used:

$$total\_used\_nodes = \sum_{n}^{N} n.used \quad (6)$$

Finally, we can formalize the constraint as follows:

$$total\_used\_nodes \leq max\_nodes \quad (7)$$

5) Cost: Cost is a very important factor in industrial projects. This is especially true in context of automobile industry, since even a small saving in component costs can impact a lot due to the large numbers of produced cars. As already discussed, PMHF values of hardware components can be mapped to ASIL values, but also to component costs, under the assumption that hardware with more stringent failure rates is also more expensive to build. It is thus relevant to define a separate constraint to limit component costs, without violating any other constraints. We first need to know which nodes are used in the current deployment, the formalization of which has already been shown in (eq. 5). Based on this knowledge, it is possible to calculate the total cost of the system:

$$total\_cost = \sum_{n}^{N} n.cost \times n.used \tag{8}$$

Subsequently we restrict the maximum cost of the system as follows:

$$total\_cost \leq max\_cost \tag{9}$$

6) Power Constraint (Energy Constraint): Energy consumption is another important factor in embedded systems and hence we generate deployments that take energy efficiency into account. In our simplified energy model we assume that each node consumes energy only as long it is active (non-idling). Therefore the non-idling time can be calculated as follows:

$$n_{non\_idle\_time} = \sum\nolimits_{t \in T'} t_{duration} \quad \text{where } T' = \{t' | \Phi(t) = n\} \tag{10}$$

Since power is defined as energy over time the energy consumption of a node n can be formalized with the following equation:

$$n.energy = n.power \times n.non\_idle\_time \tag{11}$$

Using (eq. 12) it is now is possible to calculate the total energy consumption of the system. This value again can be constrained by the maximum allowed energy consumption (eq. 13).

$$total\_energy\_consumed = \sum_{n}^{N} n.energy \tag{12}$$

$$total\_energy\_consumed \leq max\_energy \tag{13}$$

C. Pareto-efficient Deployments

In the previous sub-section we presented the constraints, which – in scope of this work – are used to synthesize valid deployments with certain characteristics. In the following subsection we discuss how to find optimal (or at least optimized) deployments w.r.t. certain criteria.

The problem of generating the pareto-optimal deployments can be formulated in terms of objectives to be optimized and constraints to be satisfied (as shown below). The formulation is followed by a brief explanation of the objectives and an introduction of the proposed optimization algorithm.

Minimize
$number\_of\_nodes$
$memory\_per\_node$
$total\_sil$
$e2e\_latency$
such that
$number\_of\_nodes \leq max\_nodes$
$memory\_per\_node \leq max\_memory$
$total\_sil \leq upper\_bound\_sil$
PMHF constraint holds

The values of variables max_nodes, max_memory and $upper\_bound\_sil$ are entered by the user, if they choose to generate pareto-optimal deployments. The PMHF constraint are also needed to be satisfied while generating the pareto-optimal deployments. Since we are aiming at minimizing the total SIL of the hardware architecture, we do not use *cost constraint* and *energy consumption constraint* explicitly.

1) Optimization Criteria: In [30], it was illustrated that participants of the survey – if it comes to system design optimization – are interested in such criteria as safety, timing, resource usage, energy consumption and cost.

Following this line of argumentation we implemented a multi-objective optimization algorithm which takes number of used nodes, used memory per node, as well as hardware costs and energy consumption as optimization criteria. For simplicity, we assumed hardware costs and power consumption to be directly correalted to the ASIL value, and thus used total ASIL as a proxy optimization criterion.

Furthermore, we took timing (end-to-end latency) into account using a solution, which, while not being Pareto-optimal according to the definition in II-C, was still optimized as explained in IV-C2.

2) Optimization Algorithm: Our algorithm works in three steps: reduction of the search-space, generation of all valid solutions for the reduced search-space, and elimination of dominated solutions, i.e., those solutions which are superseded by the pareto front.

In the first step, our approach reduces the search-space by calculating lower bounds for the criteria *memory per node* and *number of nodes*. Consider a scenario wherein the logical architecture contains 5 components and each component requires 10 units of memory. Let the values of variables *max_nodes* and *max_memory* (as entered by the user) be 4 and 20 respectively. It is clear that there will be no possible solution if the number of nodes in the technical architecture is less than 3, since the total memory needed to accommodate the 5 logical components is 50 and maximum memory allowed per node is 20. Similarly, the memory per node can be no less than 20 (assuming memory per node is a multiple of 10). As the result of this step we get a set of possible configurations in terms of a range for *number of nodes* i.e. $[min\_nodes, max\_nodes]$ and for *memory per node* i.e $[min\_memory, max\_memory]$.

In the second step, we generate valid solutions for all possible combinations of nodes and memory found in the previous step. This is illustrated in the algorithm 1 below.

Lines 2-4 shows that for each pair of nodes and memory in decision variable space, a constraint satisfaction problem

Algorithm 1 Valid Solutions

```
1:  procedure ValidSolutions
2:      for nodes in min_nodes .. max_nodes do
3:          for memory in min_memory .. max_memory do
4:              result ← CheckSAT(nodes, memory, upper_bound_sil)
5:              if result is SAT then
6:                  solution ← parse(result)
7:                  store solution to UniqueSolutions
8:                  total_sil ← extractSIL(solution)
9:                  while result is SAT do
10:                     total_sil ← total_sil − 1
11:                     result ← CheckSAT(nodes, memory, total_sil)
12:                     if result is SAT then
13:                         solution ← parse(result)
14:                         store solution to UniqueSolutions
```

is created which is solved by the Z3 SMT solver. The $checkSAT()$ function formulates the problem in Z3 syntax, where the parameters $nodes$, $memory$ and $upper\_bound\_sil$ defines the upper limits for assertions used in node usage, memory per node and total sil constraints, respectively . Line 5 checks if the constraint satisfaction problem is satisfiable. In case it is satisfiable, the output of the Z3 solver is parsed to fetch *number_of_nodes*, *memory_per_node*, *total_sil* and *e2e_latency*. These variables are combined into a $solution$ (line 6) and stored into a set of unique solutions (line 7), as we do not want to duplicate the set of valid deployments. We then try to optimize the *total_sil* by reducing it in steps of 1 (line 10) and check the satisfiability of new constraint satisfaction problem (line 11). The reduction of *total_sil* continues until Z3 cannot find a solution to the constraint satisfaction problem (line 9).

In the last step, the algorithm eliminates the solutions which are dominated by other solutions w.r.t. number of nodes, memory per node, total ASIL and thereby builds a Pareto front. As mentioned earlier our approach chooses a solution with the lowest end-to-end latency from the set of valid solutions, which were produced in step 2. For that reason we call this solution not optimal but optimized.

D. Integration in AutoFOCUS3

In scope of this work we integrated our synthesis approach together with the constraints and the optimization algorithm (introduced in the previous sub-sections) in the AutoFOCUS3 development methodology.

A typical development process in AF3 could look as follows. An engineer, designing a safety-critical system in AutoFOCUS3, ideally starts with the definition of requirements. Among other things in this phase, he will identify a set of safety goals which the system under development will have to fulfill. One of the safety goals could be that safety-critical software should not be compromised due unsafe hardware it is running on, which corresponds to the constraints 1) and 2) from IV-B.

In the next step, the engineer starts to design the structure and the behavior of the system in the logical architecture (cf. III-A). After that he can assign certain properties (cf. IV-A), such as Safety Integrity Level, to the elements of this level of abstraction. At this step it possible either to continue with the design of the technical architecture (i.e. the technical platform of the system is already fixed) or use the optimization approach from IV-C to find an optimized technical architecture together with the corresponding deployment and schedule. Regardless which option the engineer takes at some point he will end up with a logical and a technical architecture. Now he can use either the constraints from IV-B or the approach from IV-C to find either valid or optimized deployments and schedules w.r.t. the constraints he is interested in (cf. fig. 2).

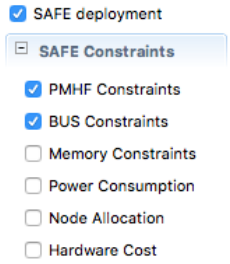

Fig. 2: Choosing constraints in AF3

The deployments and schedules which are found during this Design Space Exploration activity are valid towards certain constraints (such as 1) and 2) from IV-B). Therefore, the results of this DSE run can now be used as an evidence in a safety case to prove that the safety goal is reached by design. The integration of safety cases into AF3 development process was demonstrated in [31].

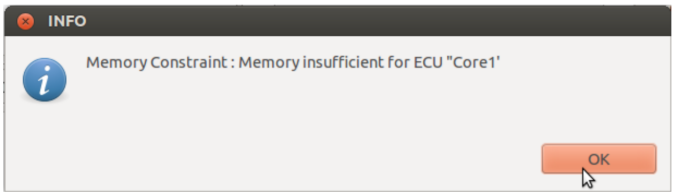

Fig. 3: Finding the unsatisfiable constraints

Given the potential complexity of the optimization problems addressed by our approach, it is possible to define the constraints so tightly that no solution can satisfy them all. For instance, the provided pool of ECUs – even if all of them were used in combination – does not provide enough memory to run all the software components. It is thus just as useful to support the user by providing guidance as to which constraints could not be satisfied. For this case we integrated the Z3s UnSAT Core functionality (cf. sec. II-A), which identifies unsatisfiable constraints for the current DSE problem. As seen in figure 3, this feature points the user exactly to the constraint which is not satisfiable and, hence, needs to be adjusted.

V. Evaluation

A. Description of the Case Study

We evaluated our approach using an industrial-like case study, which was created in the scope of the SAFE Project. This case study was modeled in AutoFOCUS3 and consists

ASIL Level	Components
ASIL D	c1, c2, c3, c4, c5
ASIL C	c6, c7, c8, c9, c10, c11, c12, c13, c14, c15, c16, c17
ASIL B	c18, c19, c20, c21, c22, c23, c24, c25, c26, c27
ASIL A	c28, c29, c30, c31, c32, c33, c34, c35, c36, c37, c38, c39
QM	c40, c41, c42, c44, c45, c46, c47, c48

TABLE II: Logical Architecture of the Case-Study

of both a logical and technical architecture. The logical architecture consists of 48 components, connected by 65 channels. The components have different ASIL levels, as described in table II.

The technical architecture consists of 9 nodes connected by a bus. Each hardware component was assigned a predetermined PMHF value (and ergo ASIL compliance) and cost (cf. table III). These values can, of course, be modified by the user to represent different hardware characteristics.

ECUt	1	2	3	4	5	6	7	8	9
PMHF	0.1	0.1	0.3	0.4	0.6	0.6	0.8	0.9	1.0
ASIL	D	D	C	C	B	B	A	A	QM
Cost	10	10	8	8	5	5	3	3	1

TABLE III: Technical Architecture of the Case-Study

B. Cost versus Safety

In our investigations, we analyzed how manipulating the criteria - both individually and collectively - affects the deployment and schedule generation process. The investigation is too large to list in detail here; more results are provided in [22]. To demonstrate the approach, we will focus on one example: how cost reduction affects safety-oriented deployments; an optimization problem reflecting design trade-offs, under the assumption that hardware with more stringent failure rates is more expensive to build. The results are shown in chart form in the next figure. The chart (cf. fig. 4) shows possible

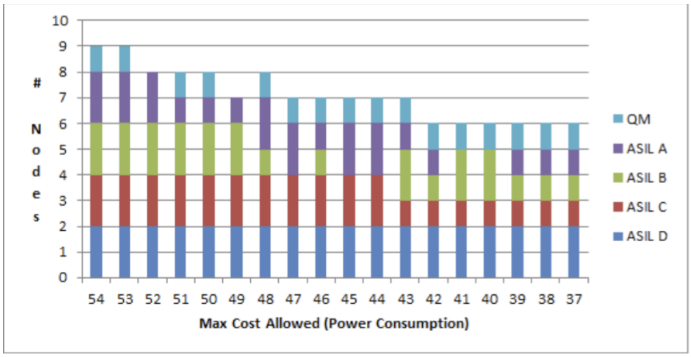

Fig. 4: Effect of Cost Constraint (non-uniform cost) on deployment (ASIL-D BUS)

deployments to satisfy the multi-criteria problem, while optimizing for power consumption, total cost of hardware, and total number of nodes. The results show that it is possible to satisfy the problem with various configurations and allows the user to choose which criteria are more relevant, and hence which deployment – among the multiple *valid* ones – is *most suitable* to their needs.

C. Evaluation of the Optimization Approach

In our second experiment, we investigated how long our approach would take to find a set of non-dominated, i.e., optimized, solutions. As input we used the case-study from section V-A. The search space was restricted using the following intervals: 1 to 9 nodes; 1 to 36 as total ASIL; and 10 to 100 MB memory per node (using a 10 MB iteration step). As discussed in section IV-C, the end-to-end latency does not have to be restricted because our approach picks the solution with the best end-to-end latency from all the solutions it finds. The total ASIL constraint represents the hardware cost factor, by assigning a high number to more capable hardware nodes.

Number of nodes	Memory per node	E2E latency	TotalSIL
5	100	263	10
7	70	292	14
6	80	248	16
6	80	257	14
6	90	210	11
8	60	269	17
6	80	281	13
8	90	192	13

TABLE IV: Pareto deployments for industry-like use case

In our experiment we used the Z3 solver with a 6 hours time limit for each evaluation. This means, that after this time period the Z3 solver times out with the best solution so far, which might be not the overall optimal solution. The result of this experiment, presented in table IV, was calculated in 126 hours. Given the size of the case study and the degrees of freedom under consideration (deployment, schedule, memory, SIL distribution, timing) we think that this time period is reasonable and feasible in a practical setting.

Most importantly, the approach allows for the explicit optimization for selected criteria, the impact of which can be seen in the last tables. One can have different deployments on the same as well as different numbers of nodes with different effects on resource usage, system attributes, and hardware costs. For instance, if the number of nodes is the absolute limiting factor, then one can choose the first deployment with 5 nodes. If however, one can accommodate a design with 6 hardware nodes, then it is possible to choose a solution optimized for Memory consumption and hardware costs, which also provides a comparatively low latency value. More details on the different effects of optimizing for different criteria are given in [22].

Above all, these outcomes are based on formal methods, they are deterministically reproducible, and form a solid foundation for safety assurance and correct/safe-by-construction solutions.

VI. Conclusion and Future Work

In the engineering of reliable and safe automotive systems, the process of mapping software to hardware is an

essential design step. In this paper we presented a design space exploration approach for multi-criteria optimization of the deployment problem, formalizing constraints relevant for system development according to ISO26262.

Not limiting our interest to generating merely valid solutions, but in good ones, we developed a multi-objective optimization algorithm, which synthesizes solutions pareto-optimized for safety, resource usage, timing and any other constraints the user defines. A state-of-the-art SMT solver is used in conjunction with the formalized constraints to find valid solutions, which satisfy the constraints, while a meta-search on top of the SMT solver provides the optimized solutions. In our example, we derived 6 constraints and implemented an optimization algorithm for 4 criteria, with ongoing work to give users more freedom to define their own constraints and criteria. Above all, the generated results are based on formal methods, they are deterministically reproducible, and form a solid foundation for safety assurance and correct/safe-by-construction solutions.

Our approach demonstrates the feasibility and effectiveness of using formal methods to generate correct solutions for safety-critical applications under real-world scenarios, increasing the confidence and validity of safety evidence for certification.

Solutions to software engineering problems are most useful when bolstered by tool-support; we integrated our work in a model-based framework, called AutoFOCUS3. Due to the seamless nature of AF3, our approach demonstrates how an integrated model-based framework can pragmatically provide support for practical problems in the development and certification of safety-critical systems in compliance with the relevant standards, and just as importantly, how formal methods can easily be made more accessible to practitioners without a formal background.

REFERENCES

[1] S. Voss, S. Zverlov *et al.*, "Design space exploration in autofocus3 - an overview," in *IFIP First International Workshop on Design Space Exploration of Cyber-Physical Systems*. Springer, 2014.

[2] S. Voss and B. Schaetz, "Deployment and scheduling synthesis for mixed-critical shared-memory applications," in *Engineering of Computer Based Systems, 2013 20th IEEE International Conference and Workshops on the*, 2013, pp. 100–109.

[3] M. Breuer, "Recent developments in the automated design and analysis of digital systems," *Proceedings of the IEEE*, vol. 60, no. 1, pp. 12–27, Jan 1972.

[4] E. Kang, E. Jackson *et al.*, "An approach for effective design space exploration," in *FOCS'10*. Springer-Verlag, 2011, pp. 33–54.

[5] S. Kuenzli, "Efficient design space exploration for embedded systems," Ph.D. dissertation, ETH Zurich, April 2006.

[6] E. B. Fernandez and B. Bussell, "Bounds on the number of processors and time for multiprocessor optimal schedules," *Computers, IEEE Transactions on*, vol. 100, no. 8, pp. 745–751, 1973.

[7] S. Prakash and A. C. Parker, "Synthesis of application-specific heterogeneous multiprocessor systems," in *ACM SIGARCH Computer Architecture News*, vol. 20, no. 2. ACM, 1992, p. 434.

[8] M. Thompson, H. Nikolov *et al.*, "A framework for rapid system-level exploration, synthesis, and programming of multimedia mp-socs," in *Proceedings of the 5th IEEE/ACM International Conference on Hardware/Software Codesign and System Synthesis*, New York, NY, USA, 2007, pp. 9–14.

[9] A. Metzner, Franzle *et al.*, "Scheduling distributed real-time systems by satisfiability checking," in *Embedded and Real-Time Computing Systems and Applications, 2005. Proceedings. 11th IEEE International Conference on*. IEEE, 2005.

[10] S. Kugele, L. Dieudonné, R. Popa, and H. Eckardt, "On the Deployment Problem of Embedded Systems." *In 13th ACM-IEEE International Conference on Formal Methods and Models for System Design (MEMOCODE'15)*, no. i, 2015.

[11] T. Saxena, "A generic framework for design space exploration," Ph.D. dissertation, Vanderbilt University, 2012.

[12] F. Ortmeier and W. Reif, "Safety optimization: a combination of fault tree analysis and optimization techniques," *International Conference on Dependable Systems and Networks, 2004*, pp. 1–8, 2004.

[13] M. S. Dhouibi, L. Saintis, M. Barreau, and J.-M. Perquis, "Safety driven optimization approach for automotive systems," in *Reliability and Maintainability Symposium (RAMS), 2015 Annual*. IEEE, 2015, pp. 1–7.

[14] R. Hilbrich and L. Dieudonné, "Deploying Safety-Critical Applications on Complex Avionics Hardware Architectures," *Journal of Software Engineering and Applications*, vol. 06, no. 05, pp. 229–235, 2013.

[15] E. Kang, E. Jackson, and W. Schulte, "An approach for effective design space exploration," *Foundations of Computer Software. Modeling, Development, and Verification of Adaptive Systems*, pp. 33–54, 2011.

[16] S. Kugele and G. Pucea, "Model-based optimization of automotive E/E-architectures," *Proceedings of the 6th International Workshop on Constraints in Software Testing, Verification, and Analysis - CSTVA 2014*, no. MAY 2014, pp. 18–29, 2014. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2593735.2593739

[17] M. Walker, M. O. Reiser, S. Tucci-Piergiovanni, Y. Papadopoulos, H. Lönn, C. Mraidha, D. Parker, D. 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.

[18] B. Selic and S. Gérard, *Modeling and Analysis of Real-Time and Embedded Systems with UML and MARTE: Developing Cyber-Physical Systems*. Elsevier, 2013.

[19] a. Aleti, S. Bjornander, L. Grunske, and I. Meedeniya, "ArcheOpterix: An extendable tool for architecture optimization of AADL models," *Model-Based Methodologies for Pervasive and Embedded Software, 2009. MOMPES '09. ICSE Workshop on*, pp. 61–71, 2009.

[20] M. Broy and K. Stølen, *Specification and development of interactive systems: focus on streams, interfaces, and refinement*. Springer Science & Business Media, 2012.

[21] S. Z. B. Schaetz, S. Voss, "Automating design-space exploration: Optimal deployment of automotive sw-components in an iso26262 context," in *Design Automation Conference (DAC), 2015 52st ACM/EDAC/IEEE*, 2015.

[22] SAFE-E, "Deliverable d4.4b: Final version of plug-in for safety and multi criteria architecture modeling and benchmarking," The SAFE-E Consortium, Tech. Rep., 2014.

[23] J. Gu, "Local search for satisfiability (sat) problem," *Systems, Man and Cybernetics, IEEE Transactions on*, vol. 23, no. 4, pp. 1108–1129, 1993.

[24] L. De Moura and N. Bjørner, "Z3: An efficient smt solver," in *Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 2008, pp. 337–340.

[25] F. Van Harmelen *et al.*, *Handbook of knowledge representation*. Elsevier, 2008.

[26] M. Ehrgott, *Multicriteria optimization*. Springer, 2005, vol. 2.

[27] A. Bauer, M. Broy *et al.*, "Automode - notations, methods, and tools for model-based development of automotive software," in *SAE International*, 2005.

[28] S. Zverlov and S. Voss, "Synthesis of pareto efficient technical architectures for multi-core systems," in *Computer Software and Applications Conference Workshops (COMPSACW), 2014 IEEE 38th International*. IEEE, 2014, pp. 366–371.

[29] N. Eén and N. Sorensson, "Translating pseudo-boolean constraints into sat," *Journal on Satisfiability, Boolean Modeling and Computation*, vol. 2, pp. 1–26, 2006.

[30] P. Diebold, C. Lampanosa *et al.*, "Practitioners' and researchers' expectations on design space exploration for multicore systems in the automotive and avionics domains – a survey," in *EASE Proceedings*. ACM Digital Library, 2014.

[31] S. Voss, B. Schätz, M. Khalil, and C. Carlan, "Towards modular certification using integrated model-based safety cases," in *proc. VeriSure: Verification and Assurance Workshop*. Citeseer, 2013.