Automatic Transition Between Structural System Views in a Safety Relevant Embedded Systems Development Process

Christian Ellen
OFFIS e.V.
Oldenburg, Germany
ellen@offis.de

Christoph Etzien
OFFIS e.V.
Oldenburg, Germany
etzien@offis.de

Markus Oertel
OFFIS e.V.
Oldenburg, Germany
oertel@offis.de

***Abstract*—It is mandatory to design safety relevant embedded systems in multiple structural system views. A typical example is the usage of a functional and technical system representation.**

A transition between these system views not only comprises the allocation of components but also copes with multiple design aspects and constraints that need to be transferred to the target perspective. Optimization goals regarding arbitrary design artifacts complicate this problem.

In this paper we present a novel comprehensive approach integrating common allocation techniques together with a partial design generation in a system wide process to optimize complex system view transitions. We demonstrate our approach using the CESAR design methodology. The original system models and requirements are used as input for our procedure and the results are directly applied to the same models.

I. Introduction

Safety standards like the ISO26262 [1] require a strict separation between the functional and the technical safety concept resulting in the necessity for multi-view system design processes. Solutions exist to allocate components from one view to another, suitable for constraints like real-time [2] or resource consumption [3]. For these approaches the system model has to be transferred manually in a processable format creating difficulties to display the results with respect to the design model. In contrast to this method we propose in this paper an integrated solution directly using the design models.

In the ARTEMIS project CESAR [4] a domain independent design approach for embedded systems has been developed. Different analysis and process guidelines are part of this approach. Systems developed according to the CESAR design approach are separated into multiple perspectives, representing the system at different structural stages.

In this paper we describe an approach how to assist the engineer in the transition from one perspective to another using automated techniques. We demonstrate how formal methods can be applied to the CESAR design approach. The approach is exemplified on a set of complex transformations of an automotive use case, in which we allocate logical components to tasks hosted on ECUs (Electronic Control Units). Furthermore, each task requires special equipment as indicated by the logical design. This equipment needs to be connected to the ECUs in a way that the cable length is minimal. System constraints, expert knowledge, and resource limitations are considered as well.

The current work is a proof of concept to demonstrate the feasibility of system wide optimizations on industrial relevant processes. The formalization for all features is still in development, but first results are discussed in this paper.

This paper is organized as follows: in section II we give an introduction to the CESAR design approach, in section III we describe our approach using a example of an redundant airbag system, and in section IV the corresponding formal model of the encoding is defined.

II. CESAR Design Approach

The CESAR design approach covers a component-based and multi-perspective development process. One part of the CESAR interoperability standard is its homogeneous view to development items of different tools and formats. The homogeneous view is based on a meta model which includes all generic concepts required in different domains (e.g. automotive and avionics) and defines the granularity of this view.

During the development process a system is specified and modelled at different levels of abstraction. Within these levels a set of basic viewpoints, called perspectives, have been identified by the CESAR and SPES2020 [5] projects. Perspectives are disjoint models of development items representing different structural stages of the system at the same level of abstraction. The five generic perspectives are:

- **Operational Perspective:** Customer needs are expressed in terms of capabilities and activities that should be provided by the system to the environment.
- **Functional Perspective:** The system functionality is modelled using functions and sub-function decomposition.
- **Logical Perspective:** In contrast to the functional perspective components on the logical perspective are

The research leading to these results has received funding from the ARTEMIS Joint Undertaking under grant agreement n°100016 and from specific national programs and/or funding authorities.

system oriented and represent system elements and their interaction.
- **Technical Perspective:** In this perspective the partitioning of components in software and hardware is performed. The system is expressed with all its physical components like sensors or computing units that host the software.
- **Geometrical Perspective:** Geometric components are closely related to the technical components. They define the dimensions, positions, and other physical properties of components are considered.

Part of the transition between two adjacent perspective is an allocation of elements in the source perspective to elements in the target perspective. Cross-perspective aspects like safety-concerns relate to items on different perspectives and abstraction levels.

In the use case presented in section III we evaluate the impact of different kinds of design constraints to the transition between the logical and the technical perspective. These constraints may originate from a safety assessment and result also in a redundant implementation of a component.

Initially the transition requires a relation between elements of the two perspectives. This defines which element of the source perspective can be allocated to which element (so called type) of the target perspective. Often, only an implicit type-concept between both perspectives is created. In our approach this relation is expressed explicitly as allocation constraints defining possible allocations.

We use the RSL [6] (Requirement Specification Language) developed in CESAR to define these type of allocation constraints:

"**`Logical element`** `LIO Accel. Front` **`shall be allocated to`** `Accel. Sensor Front Type`."

This specific constraint defines the allocation of "LIO Accel. Front" to an instance of "Accel. Sensor Front Type" (see example in section III).

These constraints emerge for example from expert knowledge or system requirements and are part of the system model. The entire allocation-relevant information can be encoded and is visible in the overall development system model.

III. Transition between Perspectives

The transition between the logical and technical perspective is illustrated by the example of the redundant airbag system. This simple example rises all challenges considered by our approach. It is a snapshot of the development process in which all requirements of the system are captured and the logical design is already decomposed. During this decomposition new constraints evolve restricting the allocation. For example if an early safety assessment identifies the need of a redundant implementation of a logical component, this could be the case performing a ASIL (Automotive Software Integrity Level) decomposition w.r.t. the ISO26262 [1]. The allocation of redundant components to the same ECU is avoided by the introduction of allocation constraints. For example, those allocation constraints postulate that redundancy (or diversity) in one perspective is preserved in another one.

Furthermore, in our example the technical and geometrical perspective are partially defined in the system model. In detail we assume that a fixed number of sensors and ECUs (Electronic Control Unit) with different types are used. The location of these elements is specified in the geometric perspective.

A. Transition Objectives

We define the following objectives for a transformation between two adjacent perspectives.

Objective 1. *All allocation constraints must be satisfied by the transition.*
The first objective demands that all constraints must hold in a transition, especially that every logical element is allocated according to their possible target elements under the given restrictions.

Objective 2. *Existing connections in each perspective must be preserved.*
The perspectives may already contain connected elements (e.g. due to expert knowledge or previous process steps). The goal of the second objective is to ensure that a transition maintains all existing connections.

Objective 3. *Allocated elements must fulfill their communication needs.*
If two logical elements are connected, their allocated technical elements must be directly or indirectly connected, too.

Objective 4. *New connections must respect resource capacity limitations.*
A transition creates a number of new connections in the technical perspective. Each of these connections must have a source and target element, and occupies some of its available resources (e.g. interfaces on an ECU). The third objective avoids dangling connections and connections which cannot fulfill their resource requirements.

Objective 5. ***(optional)*** *The transition has to be optimized w.r.t. the given minimization goal.*
Objectives 1–4 ensure that a transition is consistent and are therefor mandatory. The last objective is considered optional because non-optimized transitions are still consistent, if they obey the other constraints.

B. Application of the Approach

In the following we present our approach applied to a redundant airbag system depicted in figure 1 and show how the objectives are implemented.

a) Logical Perspective: On the left side of the figure the decomposition of the logical perspective is illustrated. The redundant airbag system is decomposed into the three subsystems "Left Side Impact," "Right Side Impact," and "Front Impact." These subsystems trigger the individual airbags of the vehicle depending on the side of impact. Each of the systems is implemented by two redundant subsystems (e.g. FI1 and FI2), containing several other components. We distinguish

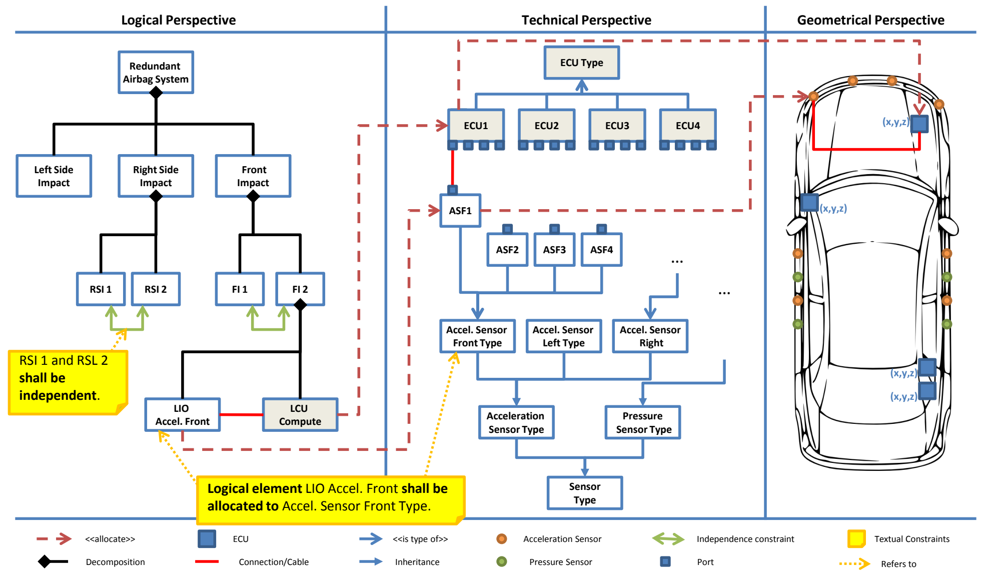

Fig. 1. The Redundant Airbag System example illustrates an allocation of two connected components within a homogeneous view of the three perspectives. All information required for the transformation are contained in the view.

these components in processing units (allocated to tasks) and input/output-components (allocated to sensors/actuators). The former we call LCUs (Logical Computation Units) and the latter LIOs (Logical Input Output components). Therefore a subsystem must be allocated to several different components in the technical perspective.

The independence constraints –a special kind of allocation constraint– are presented as green arrows between the redundant subsystems. These constraints restrict the allocation of tasks to the same ECU (objective 1). They are derived form constraints of the logical perspective by translating them to the technical perspective. Allocated tasks require connections to sensors/actuators according to connections in the logical perspective. Consequently the ECU hosting the task must be connected to suitable equipment (objective 3).

b) Technical Perspective: In the center of figure 1 the technical perspective is shown which focuses on technical components, interfaces, and connections. As mentioned before the sensors and actuators must be connected to the ECUs with respect to the allocation of the logical components. In addition to the technical components (ASF1, ASF2, etc.), their types (Accel. Sensor Front Type, Accel. Sensor Type, etc.) are modelled.

A possible allocation between a logical and technical component can be specified using these component types (see figure 1).

Two technical components are connectable if their interfaces are of the same type. Interface types are for example analog, digital and others.The types and capacities of the interfaces on ECUs and sensors create further restrictions on the allocation (objective 4).

According to objective 2 sensors which are already connected to an ECU cannot be reconnected to another ECU.

c) Geometric Perspective: The focus of the geometric perspective is on the position and dimension of the physical elements (ECUs and sensors/actuators). The length of the cables connecting the equipment with the ECUs is expressed here. The minimization of the overall cable length is used as cost function for this allocation problem (objective 5).

IV. Formalization

In order to automatically find a valid and optimized solution to the allocation problem we defined a formal specification of the transition. To evaluate the validity of such a transition we use the Satisfiability Modulo Theories (SMT) [7] solver HySAT [8]. The formalization itself is implementation-independent and not bound to a specific kind of solver. This kind of encoding is not entirely new (see [2], [3]), but we applied it in the context of our integrated approach. In this section we present only the basic ideas of the formalization.

A. Allocation Function

In our formal allocation model we abstract from the perspectives, LIOs, ECUs, etc. and define their relations based on more abstract sets. LCUs are a special kind of functions $f \in F$ which must be allocated as tasks running on ECUs. Since the real-time aspects of tasks are not part of our scope, we simplify this allocation to the function $alloc : F \rightarrow B$, which assigns every function f to an abstract box $b \in B$ representing an ECU. The solver has to find a valid solution of this function according to the transition objectives (section III).

Individual images of the function $alloc$ are restricted by the given constraints and resource limitations of the model (objective 1 & 4). This is encoded by embedding $alloc$ into first-order logic formulae and limiting its the image using allocation equations.

The semantics of an allocation equation over $alloc$ for a given $f \in F$ and $b \in B$ is defined as:

$$[\![alloc(f) = b]\!] = \begin{cases} true & \text{if f is allocated on b} \\ false & \text{otherwise} \end{cases}$$

This allows an intuitive way to formalize both, allocation and independence constraints. A typical example encoding of an allocation constraint is $alloc(f) \neq b$ avoiding f to be hosted on b. In case of independence constraints the images of two functions f_1 and f_2 can be compared directly: $alloc(f_1) \neq alloc(f_2)$

B. Resource Occupation and Connection

According to objective 4 two kinds of resource limitations are considered in our transition. The number of interfaces provided by an ECU is the first limitation, while the available equipment (sensors/actuators) itself is the second one.

Every task allocated on an ECU must be able to communicate with its equipment. Therefore, it has to occupy some of the available interfaces. The exact numbers and types of interfaces depend on the type of the connected equipment.

In a nutshell, the idea is to define the function $occ : F \times B \times T \mapsto \mathbb{N}$ which represents the number of occupied interfaces for each function f, box b and interface type $t \in T$. The image of this function is restricted by a set of linear inequalities ensuring that every function allocated on an ECU occupies the correct amount of interfaces and that the total number of interfaces is not exceeded by the overall allocation.

The second kind of limited resource in our transition is the available equipment. Every equipment $e \in E$ has a given geometric position and can only be used by a single function. Therefore, we define the function $use : F \times E \mapsto \{0, 1\}$ which is only true iff the a given function f is using the specific equipment e. Again, linear inequalities are used to ensure that every function is attached to the correct amount of equipment (objective 3) and no two function share the same equipment.

Based on use and $alloc$ we define the function $con : E \times B \mapsto \{0, 1\}$ in order to formalize the direct connection between an equipment and a box. The semantics of con is defined as:

$$con(e,b) = \begin{cases} 1 & \text{if } \forall f \in F : \\ & use(f,e) = 1 \implies alloc(f) = b \\ 0 & \text{otherwise} \end{cases}$$

Existing connections between equipment and ECUs in the technical perspective can be preserved (objective 2) by setting $con(e, b) = 1$ and thus assuring that any function using the equipment e is allocated on the connected box b.

C. Optimization Goal

The optimization goal in the example use case (objective 5) is the minimization of the overall cable length. We use a function $dist : E \times B \mapsto \mathbb{N}$ to model the individual distances between equipment and ECUs. Its values are statically calculated based on the positions of the elements in the geometric perspective. The minimization of the resulting cable length is computed based on the con and $dist$ functions:

$$\min_{con} \sum_{e \in E} \sum_{b \in B} con(e, b) * dist(e, b)$$

Using this formalization we are able to implement an automated procedure to solve and optimize perspective transition.

V. Conclusion and Future Work

In this paper we discussed the different aspects of the transition between design perspectives. We have shown that this problem covers more than just the allocation of components. As a result of our procedure missing connections are added to the target perspective.

The use case demonstrated how safety constraints and functional aspects, like the need for specific equipment, can be used to generate a complex transition between the logical and technical perspective. The solution is optimized using the overall cable length between ECUs and equipment.

In the CESAR project we applied our approach on an industrial scale example consisting of approximately 30 ECUs, 100 equipment units, and, 50 logical components using the same optimization criterion. The scalability has to be evaluated in more detail but some realistic examples are already feasible.

In our airbag example implicit knowledge is used to identify LIOs, ECUs, and equipment limiting the flexibility of the approach. In our ongoing research we will generalize the approach with more flexible and more expressive features. Such generalizations are not limited to direct connections but can connect components transitive through multiple elements. Furthermore, the independence between elements, currently expressed only as allocation constraints, will be refined with a concept of diverse implementations. The impact of independence and diversity constraints can be limited by the introduction of a scope concept. In addition, we plan to extend the optimization goal with arbitrary metrics.

These extensions in turn yield a more comprehensive solution for automatic perspective transitions.

References

[1] Technical Committee ISO/TC 22 Subcommittee SC 3, Ed., *ISO/WD26262: Road Vehicles - Functional Safety*. Automotive Standards Committee of the German Institute for Standardization, 2009.

[2] M. Glaß, M. Lukasiewycz, J. Teich, U. D. Bordoloi, and S. Chakraborty, "Designing heterogeneous ECU networks via compact architecture encoding and hybrid timing analysis," in *DAC*. ACM, 2009, pp. 43–46.

[3] F. Reimann, M. Glaß, C. Haubelt, M. Eberl, and J. Teich, "Improving platform-based system synthesis by satisfiability modulo theories solving," in *Proceedings of the 8th International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*, Scottsdale, USA, Oct. 2010, pp. 135–144.

[4] The CESAR Consortium, "CESAR Project," 2009. [Online]. Available: http://www.cesarproject.eu/

[5] W. Damm, H. Hungar, S. Henkler, I. Stierand, B. Josko, P. Reinkemeier, A. Baumgart, M. Büker, T. Gezgin, G. Ehmen, and R. Weber, "SPES2020 Architecture Modeling," SPES2020, Tech. Rep., to be published 2011.

[6] A. Mitschke, N. Loughran, B. Josko, M. Oertel, P. Rehkop, S. Häusler, and A. Benveniste, "RE Language Definitions to formalize multi-criteria requirements V2," The CESAR Consortium, Tech. Rep., 2010. [Online]. Available: http://cesarproject.eu/fileadmin/user_upload/CESAR_D_SP2_R2.2_M2_v1.000.pdf

[7] C. Barrett, R. Sebastiani, S. Seshia, and C. Tinelli, *Satisfiability Modulo Theories*, ser. Frontiers in Artificial Intelligence and Applications. IOS Press, February 2009, vol. 185, ch. 26, pp. 825–885.

[8] M. Fränzle, C. Herde, T. Teige, S. Ratschan, and T. Schubert, "Efficient solving of large non-linear arithmetic constraint systems with complex boolean structure," *Journal on Satisfiability, Boolean Modeling and Computation*, vol. 1, pp. 209–236, 2007.