Model-based Analysis and Synthesis Methods for Dependable Embedded Systems

András Balogh, András Pataricza, György Csertán, Balázs Polgár, Melinda Magyar
Budapest University of Technology and Economics, Department of Measurement and Information Systems
Magyar Tudósok krt. 2., Budapest, H-1117, Hungary
Phone: (36)-(1)-463-3594, Fax: (36)-(1)-463-2667, {abalogh,pataric,csertan,polgar,mmagyar}@mit.bme.hu

Abstract - The increasing complexity of distributed embedded systems, as found today in airplanes or cars, becomes more and more a critical cost-factor for their development. Model-based approaches have recently demonstrated their potential for both improving and accelerating (software) development processes. Therefore, in the project DECOS\(^1\), which aims at improving system architectures and development of distributed safety-critical embedded systems, an integrated, model-driven tool-chain is established, accompanying the system development process from design to deployment.

Beside the tool-chain a powerful verification and validation framework has been developed both for checking the correct behavior of the tools and for assuring the consistency, completeness and correctness of the related metamodels and developed models.

This paper gives an overview of the DECOS concepts and tool-chain and presents novel methods and tools for the analysis and synthesis of the system models.

I. INTRODUCTION

Distributed system composition is the main trend in creating safety-critical (SC) real-time systems like automotive, aerospace, and industrial control systems. Their growing complexity (e.g. tens of control units in a modern car) led to an integrated architecture concept. It supports the sharing of hardware resources between different sub-applications for the sake of cost reduction, but it still maintains the overall system safety by properly isolating jobs from each other.

Validation and certification of SC systems are a key problem. They are especially hard, if not impossible at all, if the behavior of the system is non-deterministic. The time triggered (TT) paradigms (such as TTP/C and FlexRay) use a strictly deterministic, static, design time generated schedule for both the computation jobs in the processing nodes and the inter-node communication tasks.

Several functional and extra-functional requirements have to be fulfilled during the development of applications running over an integrated architecture. The validation of the system design against those requirements has to be highly automated in order to keep the development process productive. Furthermore, automated, Quality-of-Service driven synthesis methods should also be used assuring a proper design quality.

![Model-Driven Development](Fig. 1: Model-Driven Development)

The traditional model-driven architecture (Fig. 1) approach defines the hardware-software integration step as an atomic, automatic mapping between platform-independent and platform-specific models. However, this solution can be used in enterprise software development it may not be suitable in highly constrained embedded environments. In order to overcome this problem we developed an interactive, iterative method that automates the mechanic steps of the integration and incorporates engineer decisions at the critical steps of the development process.

The involvement of the developer into the decisions results in a process that can incorporate non-formal decision constraints, such as physical arrangement of the nodes, supplier-related considerations, and so on. This helps to customize the process and to enforce existing company policies during system development.

We present novel methods and tools supporting analysis of the system models from multiple aspects as an integral part of the model-driven development process, and especially during the integration of hardware and software components. As this integration process is by its very nature an iterative and interactive series of model refinements, it is important to check the consistency of the intermediate models with the initial requirements, in order to indicate compliance problems as early, as possible.

\(^1\) DECOS (Dependable Embedded Components and Systems) is an integrated project partially funded by the EU within priority “Information Society Technologies (IST)” in the sixth EU framework program (contract no. FP6-511 764)
The tools - checking temporal and dependability related properties - are integrated into an open framework [1]. Thanks to its openness, further extensions by new analysis tools implementing checks for other extra-functional aspects, like power consumption or physical arrangement is explicitly supported.

In addition to the advanced analysis methods embedded into the tool chain we also suggest a new, optimization based synthesis method. This generates both the software-hardware mapping (resource allocation) and the scheduling of the intra and inter-node communication. The set of constraints embedded into the optimization algorithm guarantee that all the functional and extra-functional system requirements will be met by the resulting system design.

Our system design optimization tool includes several different possible objective functions that cover design aspects as system cost, throughput, extensibility, and robustness. The tool can be used to discover the design space by calculating the optimal solution for the simple objectives, and after that the designer can position the final solution between these solutions by setting a composite (combined) objective function for the optimization process. This enables the balancing between the contradictory optimization aspects.

The optimization tool can also be used to evaluate existing system designs. Using the same interface as in case of schedule generation the developer can validate the design, furthermore the tool is able to rank the design compared to the absolute optimum solution. This gives exact measure about the value of a given design.

The tool chain presented in this paper [2] is implemented based on the open, industry-standard Eclipse framework that offers strong infrastructure for tool development and enables the seamless integration of different products in a common, multi-platform environment.

The methods and tools presented are implemented in the framework of the EU FW6 IP DECOS (Dependable Components and Systems) project. The approach was validated by small industrial pilot applications from the automotive, aerospace, and industrial process control domains.

II. DECOS OVERVIEW

A. Concepts and Architecture

The basic principles for achieving dependability in a DECOS system [3] are strong fault encapsulation, fault tolerance by means of replication and redundancy, and separation of safety-critical from non-safety-critical functionality. These principles, in particular redundancy, lead to functional distribution. Additionally, supporting hard real-time applications requires guaranteed response times. Therefore, the functional structuring of a DECOS system comprises a number of (nearly) independent Distributed Application Subsystems (DASs), each realizing a part of the overall system service. DASs can be further subdivided into jobs, which represent the smallest executable software fragment in the DECOS model. Jobs communicate with each other by the exchange of messages via virtual networks.

![DECOS cluster](image)

**Fig. 2:** DECOS cluster with four nodes and three DASs: one with jobs A, B, C, another one with P and Q, and a third one with U, V, W, X, and Y. A, B, and Q have two replicas each, while P has three.

Furthermore, a DECOS cluster is physically structured into a set of distributed node computers interconnected by a time-triggered network. Each node computer comprises several encapsulated partitions, which serve as the protected execution environment for jobs. During the development of a DECOS system, a mapping of jobs to partitions has to be established as indicated in Fig. 2. Due to the guaranteed non-interference of individual partitions, a DECOS node is able to host multiple jobs belonging to different DASs, even exhibiting different levels of criticality. Conceptually, all jobs could be allocated to a single processing node, as long as resource limits and hardware fault-tolerance do not require distribution.

Taking the achievements of research in the area of dependable system architectures into consideration, DECOS does not intend to design the complete system architecture from scratch. Instead, it presumes the existence of a core architecture, providing the core services:

- Deterministic and timely transport of messages.
- Fault tolerant clock synchronization.
- Strong fault isolation.
- Consistent diagnosis of failing nodes.

Any architecture that provides these core services can be used. It has been demonstrated that the Time-Triggered Architecture (TTA) [4] is appropriate for the implementation of applications in the highest criticality class in the aerospace domain according to RTCA DO-178B [5] and consequently meets the DECOS requirements as core architecture.

On top of the core services, DECOS provides a set of architectural (or high-level) services:

- Virtual Networks (VN) and Gateways.
- Encapsulated Execution Environment (EEE).
- Diagnostics.

VN represents the communication system for DASs, embedded on the physical cluster network. Gateways provide means to exchange information between VNs, as well as with external networks, in a controlled way. The EEE is a partitioning real-time operating system that enables the execution of jobs from different DASs and of different criticality on the same hardware with guaranteed fault encapsulation. This is achieved by housing each job in its own partition with strong spatial and temporal protection [6]. Compared to other partitioning operating systems (e.g.
ARINC653 LynxOS, AUTOSAR Tresos), the EEE is very small in terms of code size (< 1MB) and can run on comparatively simple hardware [7]. The diagnostics service of DECOS both assists fault encapsulation and supports prognoses of hardware breakdowns.

Architectural services are implemented in a form influenced by the underlying (HW-)platform. In order to minimize dependency of application programmers on a certain implementation of these services, the Platform Interface Layer (PIL) provides a platform independent interface of the architectural services for application jobs.

An important DECOS feature is the support of both time- and event-triggered messages. Time-triggered (TT) messages transmit state values like the current speed periodically, while event-triggered (ET) messages transmit changes, e.g. the difference to the previous speed value whenever that change passes a certain threshold. So, while a transient transmission error of a TT message can be compensated with the next one, this is not the case for ET messages. Therefore, the latter cannot be utilized for transmission of safety-critical information. TT messages are also denoted as state messages, and ET messages as event messages. Essentially, state messages realize the parallel computing concept of a conflict-free distributed virtual shared memory.

B. The DECOS Tool-Chain

![DECOS Tool-Chain Diagram]

Fig. 3: DECOS tool-chain overview. Feedback loops, e.g. for failure reporting, are not shown.

III. THE ANALYSIS AND SYNTHESIS METHOD

A. The Allocation and Scheduling Task

As part of the PIM-PSM mapping process the software jobs have to be allocated to hardware nodes. During allocation several constraints have to be taken into account:

- the total memory consumption of the jobs allocated to a single node should not exceed the memory capacity of the node (both for code and data memory regions)
- CPU utilization of the jobs and middleware modules allocated to a single node should be less than 100%
- Replicas of a single job should not be mapped on the same node (with respect to the fault-tolerant requirements)
- Jobs handling special peripheral units (A/D converters, PWM units, DSP cores, and so on) should be mapped to a node that has the specific unit
- Jobs may have compatibility restrictions as specified by the user. All job instances should only be mapped to a node that is compatible with the specific job.

The job allocation algorithm tries to map the software components (jobs) to a minimal set of nodes while fulfilling all the allocation constraints. The DECOS allocation tool [11] implements a heuristics based search strategy for the allocation of software units.

As part of the PIM-PSM mapping process the scheduling of the messages and real-time operating system (RTOS) tasks have also to be performed. Based on the result of allocation, the PIM-PSM mapping tool synthesizes the communication map of the system that includes message instance to network segment mappings, gateway configurations and message to job mappings. This is the
input of the message scheduler tool [12] that generates the configuration for the communication subsystem.

The RTOS task scheduling is also done statically, in design time. This step is executed after the message scheduling, and the communication schedule is used as an input for it. The task scheduler generates the schedule table of the operating systems, and the skeletons of the tasks for each node separately.

The last step of the hardware-software integration process is the generation of the so-called Platform Interface Layer (PIL) course code that bridges between application level and operating system level function calls (message sending, timing, etc.) The PIL also contains the implementation of the DECOS high level services (diagnostics, virtual networks, gateways).

B. Limitations of the DECOS tools

If allocation and scheduling are performed in separate steps (Fig. 4) it can happen that a particular allocation is not schedulable. In this case backtrack is needed: another candidate allocation is generated and is tried to be scheduled. This process goes on until a feasible schedule is generated for a feasible allocation. This is the usual way and this was implemented first in the DECOS project.

However, a one-step solution method that generates the allocation and the schedule together would increase the productivity through decreasing the development time and it
could provide a solution that is closer to the optimum according to some criteria. In the next section such a solution method is discussed.

C. The Solution Method

In order to overcome the limitations of the multi-step heuristics based allocation and scheduling process, a new, single-step optimization based synthesis method has been developed. The method relies on mixed integer programming (MILP) that is widely used in solving optimization problems of different kinds.

The first step of the optimization is the generation of the MILP equation system. This is based on (part of) the system model, namely the job, job instance, node, message and application objects (and their respective attributes), is extracted from the PIM-PSM tool and is converted to the input format of the solver tool.

Besides the application specific information, there are some generic and communication specific constraints that are also part of the MILP problem. Generic constraints include the definition of task-node, and message-network segment relationships (mutual exclusions). Communication protocol specific constraints represent special criteria (like the minimal and maximal communication schedule length, etc.).

Based on the requirements, additional constraints, representing extra-functional requirements, can be defined [10]. These elements can be added to the original set of constraints and the solver will automatically process them.

The resulting MILP model contains all information and constraints that present application related (functional and extra-functional), application independent (generic safety, generic limitations), and protocol specific constraints.

In order to be able to find the optimal solution, formalization of the optimality criteria should be done. In case of critical embedded system design several different aspects should be assessed:

- **Total system cost** is important in mass production (like in automotive industry). The goal is the minimization of production costs while fulfilling all the other design criteria.
- **System throughput** (or end-to-end delay) can be also an important aspect. Maximum delay is a design requirement in several cases, but the minimization of delay can also be a design goal.
- **Extensibility** of the system has several aspects. The design goal can be to maintain a large CPU and memory reserve on a single node in order to be able to extend the system with a new job. An other scenario can be the equalization of reserves between node that enables the future integration of several small tasks or a distributed application.
- **Robustness** is also a critical design factor. In integrated system design, each application has its own availability criteria that must be fulfilled by the implementation. Our high-level objective serves the minimization of simultaneous application brake downs. For instance, if a car has electrical brake and steering subsystems (each duplicated), in case of a given number of simultaneous hardware node faults (2 in this case) some of the applications will fail. In worst case, both applications go down, but in case of a more robust system design, only one of them will actually fail.

The five different goals described here have been

![Fig. 6: The Control Panel of the PSM analysis and synthesis tool](image-url)
implemented as possible objective functions for our optimization tool. The advantage of these criteria is that the details of the mathematical optimization problem are hidden from the engineer and he can work with higher level concepts.

In real-life scenarios a single objective may not be sufficient. Our method also supports the creation of composite objectives consisting of the basic ones (with different weights).

The MILP-based method uses the ILog CPLEX [13], an industrial grade optimization tool, for model solving. A graphical front-end tool has also been developed that offers higher usability by visualizing the parameters and results of the process.

IV. THE ANALYSIS AND SYNTHESIS TOOL

The PSM Analysis and Synthesis tool consists of two Eclipse plug-ins. One of them is an EMF (Eclipse Modeling Framework) plug-in which contains the PSM models. These models are imported from the existing DECOs PIM-PSM tool. The other plug-in is responsible for the user interface management. For this purpose we use SWT to build the graphical user interface (GUI).

A. The scheduler generator plug-in

The scheduler generator plug-in is used to import the PSM models which are visualized on the graphical user interface ad to interface with the solver tool.

Fig. 5 illustrates a part of the PSM metamodel. A system contains nodes, applications, job instances, and messages. A job instance may be part of a replica set (if the original job is replicated). Jobs and messages belong to applications. Messages have a single sender and several receiver jobs. The parameters of the elements are stored in the attributes of the corresponding objects.

B. The scheduler GUI plug-in

The scheduler GUI plug-in contains the visualization part of the tool. Based on SWT, the parameters of the optimization can be easily configured before the optimization. This information can be set in a window called Control Panel. The developer can start the optimization by pressing a button. If all data are valid, the synthesis of the systems starts. The Results Panel is responsible for drawing the solutions to a radar chart. This panel contains a progress bar to show the status of the ongoing process and an allocation and schedule chart to show the details of the optimal solution.

The Control Panel – This is the panel for entering the optimization criteria for the synthesis process. These parameters form three groups:
- optimization objectives,
- execution settings,
- additional constraints.

The optimization objectives (as described earlier) are high-level criteria for the optimization. In order to support complex objectives, the developer can select multiple objectives and can assign different weights to them.

Additional settings for the execution can be defined by which for instance the execution time or the maximal number of solutions to be found can be limited. This can be used for feasibility checking of a model before doing the real (and most probably time consuming) optimization.

In some cases it is hard to give absolute constraint values for some of the extra-functional criteria. The additional constraints section of the GUI can be used to specify relative values for some of the system properties.

In this case, a first optimization run is executed that searches for the optimal value(s) of the specified constraint(s), and after that – during the final optimization – the relaxed values will be used as constraints.

For example, the goal of the designer can be the development of a “fast” system (using throughput as objective). In most cases this results in huge hardware cost. After the first results, the question can be: “How much would the system cost if we relax the throughput by 30%”. These types of design questions can be easily specified through the user interface, and answered by the solver tool. Fig. 6 shows all the possible parameters that can be set.

As the first step of the optimization the validation of the control parameters is performed. In case of wrong or mistyped values an error message is shown to the user. If parameters are validated the optimization engine is called.
The Results Panel – This panel visualizes the result of the optimization process (Fig. 7). Since the process can take a long time the intermediate results are shown on the fly. The panel consists of three main parts: the progress bar, the radar chart and a schedule chart.

The progress bar shows the actual status of the optimization; the status of the process, the number of the solutions found till the moment and the time elapsed since the beginning of the process.

A single radar chart shows the properties of all the solutions that are already found. These properties are the previously defined and weighted objectives of the optimization like throughput and the number of nodes (i.e. hardware cost). The properties of the last found solution are listed on the right side of this section. By visualizing the properties on a radar chart, the different solutions can be compared easily and the design decisions are supported this way.

The third part of this panel contains an allocation and schedule chart which illustrates the physical implementation of the result of the synthesis. The nodes of the system and the jobs on these nodes are visualized time dependently. One cycle of the system is illustrated with the periodic jobs of the different nodes. The usage of the system network is also illustrated with the messages between the jobs. The parameters of these system elements can be listed below the chart.

V. USAGE SCENARIOS

The main intended usage scenario of the optimization based tool is the synthesis of system PSM-s using various objectives during the system design process. There are however some additional possible scenarios to be considered.

The tool is also capable of doing analysis and assessment of existing system models. After importing a complete model (with existing allocation and scheduling maps) the solver can a) validate the mapping information based on the built-in constraints, b) calculate the optimal objective values for the given design (by re-scheduling it) and rank it using the radar chart.

The chart is a compact visualization of the model properties along the five different objectives (coordinates). Each schedule appears as a separate polygon on the chart and the designer can compare them easily.

This kind of visualization points out the differences between possible solutions and helps to find the trade-off between various, often contradicting objectives.

VI. RELATED WORK

Several commercial and academic tools support allocation and scheduling ver the TT principle. TTTech [12] offers tools for scheduling the communication and task execution in TTP/C [18] and FlexRay [15] based clusters. These tools generate a single solution based on internal heuristics, moreover, they require user interaction in several steps of the scheduling.

Mentor Graphics also offers a network design tool (called Volcano Network Architect) [14] able to synthesize the communication architecture for in-vehicle communication networks. However it only supports (event-triggered) CAN [16] and (time-triggered) LIN [17] protocols. As CAN (treated as the core communication protocol) no scheduling tables are generated, the tool controls the communication scheme by setting the message ids (that represent priorities) in order to fulfill all the timing requirements.

Research results complementing the industrial tools are presenting various scheduling methods for a variety of communication protocols [19,20,21]. These methods consider only the functional and timing requirements of the system, without other aspects. The software-hardware allocation is also out of scope of these techniques.

Shariful et al. [22] present a multi-variable optimization approach for the hardware-software allocation problem. The solution is similar to our method presented in this paper, as it handles multiple aspects of requirements, but it does not target scheduling problems.

All of the methods described in the related works use heuristics based solutions for solving part of the allocation-schedule problem set. Our method uses mathematical optimization supported by powerful solver tools in order to be able to handle models of practical size. The growing computing power of the desktop computers allows the application of such methods in the engineering process.

VII. CONCLUSION

In the DECOS project a novel analysis and synthesis method have been developed that solves in one single step the allocation and scheduling problem in the design of real-time, safety-critical embedded systems in a time-triggered environment. The unique strength of the approach is the tunableness of the optimization objectives, i.e. the system designer can weight the different, often contradictory criteria such as hardware cost, throughput, extensibility and robustness.

In addition a tool has been developed that implements this method and visualizes the possible results on a single radar chart. This way the design decisions are supported: the system designer can compare the different solutions and choose the one that fits the best to the specific requirements of the system to be designed.

REFERENCES


