

From Model-Based Design to Deployment of Integrated, Embedded, Real-Time Systems: The DECOS Tool-Chain

Wolfgang Herzner, ARC Seibersdorf research, wolfgang.herzner@arcs.ac.at
Martin Schlager, TTTech Computertechnik AG, Martin.Schlager@tttech.com
Thierry Le Sergent, Esterel Technologies, Thierry.LESERGENT@esterel-technologies.com
Bernhard Huber, Technical University of Vienna, huberb@vmars.tuwien.ac.at
Shariful Islam, Technische Universität Darmstadt, ripon@informatik.tu-darmstadt.de
Neeraj Suri, Technische Universität Darmstadt, suri@informatik.tu-darmstadt.de
András Balogh, Budapest University of Technology and Economics, abalogh@mit.bme.hu

Abstract

The integrated EU-project DECOS aims at the provision of enabling technologies to move from federated to integrated distributed architectures in order to reduce development, validation and maintenance costs, and to increase the dependability of embedded applications in various application domains. A major concern is the development of an integrated tool-chain accompanying the system development process from design to deployment. This paper gives an overview of this tool-chain and outlines important design decisions and features.

1 Introduction

Over the past decades, development of embedded systems has followed a customised design approach, resulting in rather isolated applications, reinvention of system design concepts, and minimal reuse of code across diverse application domains. Consequently, in modern cars a number of services like the body electronic subsystem or the power-train control subsystem co-exist, each equipped with its own electronic hardware, communication cabling etc. While this approach eases system composition and supports fault encapsulation (a failure in one subsystem will not affect others), it implies at least increased hardware costs, weight, and power consumption.

In contrast to such "federated" system architectures, DECOS¹ targets the development of an "integrated" system architecture, where the same hardware infrastructure (computational resources, communication network) is shared by several subsystems. For extending its applicabil-

ity, mixed-criticality applications, including those with hard real-time requirements, shall be supported.

When integrating (safety-critical) subsystems into one platform, it has to be guaranteed that each of them can be executed in a protected environment that resembles the environment of the federated architecture, i.e. it would not share resources with other subsystems. In particular, subsystems must not disturb each other, and faults emanated in one subsystem must not propagate to others.

A dedicated system architecture (relying on strong fault encapsulation and fault tolerance by means of replication, separating safety-critical from non-safety-critical functionality or encapsulated execution environments) is essential for achieving the intended goal and hence a central outcome of DECOS. For successfully exploiting the benefits of the system architecture, tool support is also crucial. Therefore, a tool-chain is developed for design, development, configuration and integration of DECOS applications, which is presented in this paper.

After the next chapter, which shortly outlines DECOS and the tool chain, system modelling, configuration establishment, and behaviour modelling are addressed by chapters 3 to 5, respec-

¹ DECOS (Dependable Embedded COmponents and Systems) is an integrated project (IP) funded by the EU within priority "Information Society Technologies (IST)" in the sixth EU framework programme (contract no. FP6-511 764).

tively. After putting all these pieces together in chapter 6, the last chapter draws a conclusion and depicts a short outlook.

2 Overview

2.1 DECOS Concepts and Architecture

Here, only a summary can be given. See [10] for more details.

Supporting safety-critical applications almost enforces distribution in order to achieve sufficient redundancy and tolerance against hardware faults. Additionally, supporting hard real-time applications requires guaranteed execution and transmission times. Therefore, a DECOS system consists of a number of (nearly) independent *Distributed Application Subsystems* (DAS), running on a set of hardware *nodes* connected via a *communication network*. DASs can be subdivided into *jobs* representing the smallest executable software fragment in the DECOS model, and which communicate through the exchange of *messages*. Nodes and network together establish a *DECOS cluster*, as indicated in Fig. 1.

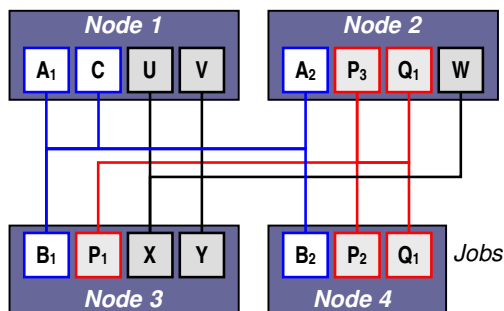


Fig. 1: DECOS cluster with four nodes and three DASs: one with jobs A, B, C, another one with P and Q, and a third one with U, V, W, X, and Y. A, B, and Q have two replicas each, while P has three.

In order to benefit from existing achievements, DECOS 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) [9] is appropriate for the implementation of applications in the highest criticality class in the aerospace domain according to RTCA DO-178B [13]

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

VNs represent 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 enables execution of jobs from different DASs and criticality on the same hardware with guaranteed fault encapsulation. This is achieved by housing each job in an own partition with strong spatial and temporal protection. And the diagnostics service both assists fault encapsulation and supports prognoses of hardware breakdowns.

In order to minimise dependency of application programmers on a certain DECOS implementation, 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.

2.2 DECOS Tool-Chain

For supporting development of certifiable DECOS applications by integrating several DASs on one cluster, a dedicated tool-chain is defined and currently under development. As shown in Fig. 2, it adopts a model-driven approach [12]. First, the *Platform Independent Models* (PIMs) of the DASs are created, which serve two purposes.

On the one side, together with the specification of the target cluster (Cluster Resource Description - CRD) and other information (job size etc.), it is used to derive configuration and scheduling information, as well as to generate the PIL, by transforming the PIMs into the *Platform Specific*

Model (PSM). In Fig. 2, "Candidate PSM" is denoted rather than "PSM", because if scheduling fails, another allocation has to be chosen.

On the other side, PIMs are used to guide the development of jobs, by modelling their behaviour, which is addressed in section 5. Finally, the results of both activities are integrated to achieve the target executables, which can then be downloaded to the application cluster.

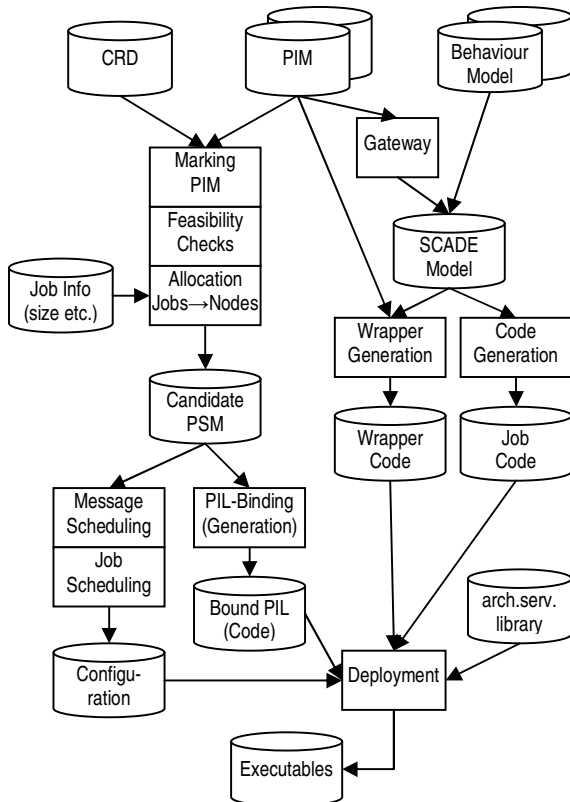


Fig. 2: DECOS tool-chain overview. Feedback loop, e.g. for failure reporting, are not shown.

3 System and Configuration Modelling

This chapter addresses generation of relevant input for the PIM→PSM transformation process, namely the PIMs and the description of the cluster hardware and resources (CRD).

3.1 PIM and its Generation

The purpose of the PIM [16] is to formalize the functional, dependability, and performance requirements of the DAS in an implementation platform independent manner. It is the place of the first steps of system architecture conceptualization. DECOS platform services - both at

core and high-level - are handled in an abstract form that is easy to use and understand at this level of design.

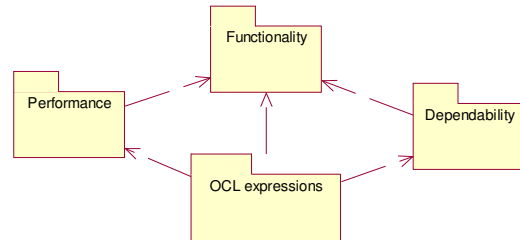


Fig. 3: PIM meta-model

According to the concepts of MDA [12], the PIM has its own meta-model. It is a UML model composed of four packages, each containing its respective object diagrams: (I) Functionality package: the basic functional elements like DAS, Job, Interface, Port, Message, Sensor/Actuator, State Variable, etc. (II) Dependability package: the dependability attributes of functional elements: reliability, availability, SIL, redundancy degree, etc. (III) Performance package: the performance attributes of functional elements: WCET, period, phase, deadline, latency, etc. (IV) OCL expression: additional semantic constraints: multiplicity dependability etc.

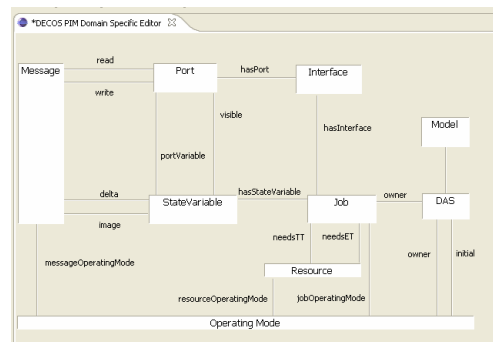


Fig. 4: DSE for the PIM under VIATRA

Since the data model of the PIM – XML – is hard to edit even with XML editors, two solutions are provided in order to ease the generation of a PIM: (i) to use the same UML tool as for high-level system design. In this case, the DECOS-PIM-XML file is generated from the XMI output of the UML editor. Currently Rational Rose 2003 and Rational Software Modeller are supported. (ii) to use a Domain Specific Editor (DSE) (see Fig. 4), which would allow for creating only meta-model compliant PIMs. Currently a DSE is implemented under the Eclipse technology². It runs

² <http://www.eclipse.org/>

directly inside of VIATRA [1] which is the selected tool for PIM-PSM mapping. This way importing PIMs is not needed any more.

3.2 CRD and its Generation

It is the purpose of the so-called *Cluster Resource Description (CRD)* to capture the relevant characteristics of the platform for the software-hardware integration in the DECOS design flow. These characteristics include amongst others computational resources (e.g., CPU and memory), communication resources, and dependability properties.

In order to ease CRD creation, a graphical, domain-specific modelling environment is developed, using the Generic Modelling Environment (GME). GME is a configurable framework for creating domain-specific modelling environments [11]. The configuration of GME is performed via the *Hardware Specification Model (HSM)*, a meta-model which formally describes the targeted modelling domain, i.e. it describes the entities, its attributes, the relationships, and constraints that can be expressed with and that are validated by the resulting modelling environment.

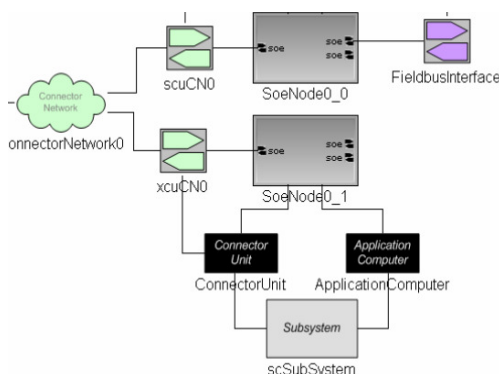


Fig. 5: Example (cut-out) of a CRD built with GME

One aim of the HSM is to facilitate reuse and hierarchical composition of the resource model. This is achieved by separating the resource modelling process into two phases: the resource capturing and the resource composition.

Resource Capturing: The specification of reusable hardware entities of a DECOS platform, so-called *resource primitives*, is addressed by the resource capturing phase. Resource primitives are the smallest physical hardware units whose characteristics are captured. Examples for resource primitives are: processors, memory elements, communication interfaces. However, the HSM provides mechanisms to extend the set

of resource primitives that can be modelled in order to be flexible and extensible with respect to the types of resource primitives [6].

Resource Composition: This second phase of the modelling process is concerned with the composition of an entire CRD (including the internal setup of DECOS components and their interconnection) out of the previously modelled resource primitives. The composition is guided by the DECOS component model [10], which is fully captured by the HSM.

The interface to the subsequent tools in the development process realising the software/hardware integration is specified using the extensible markup language (XML).

4 Deriving the Infrastructure

Based on these inputs (bound PIL will be addressed in 4.3), the next steps in the DECOS tool-chain are described:

- PSM-generation (mapping jobs to nodes)
- scheduling
- PIL-generation

4.1 PSM-Generation

Being (still) a model, the main purpose of the PSM is to precisely specify which application jobs are to be assigned to which cluster nodes, under consideration of all constraints defined in the PIMs of the DAs and the available resources described in the CRD. Fig. 6 depicts a small part of its meta-model in UML-notation.

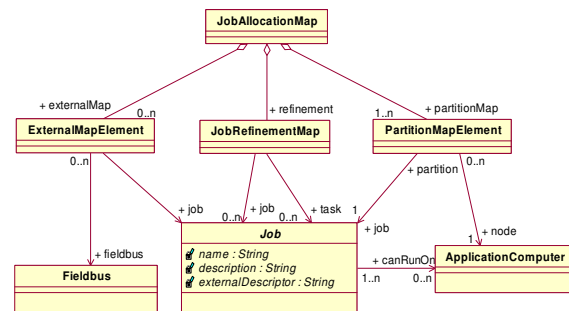


Fig. 6: Part of PSM meta-model

The PSM generation process encompasses a number of steps like PIM marking, feasibility checks, and the allocation process. The significant part of the mapping process is to allocate jobs with different criticality to a shared HW platform (HW nodes) subject to constraints and requirements of fault-tolerance and real-time.

It is sometimes necessary to add manually information to the PIM. This step is called PIM marking. It is an interactive step, where information like specific middleware requirements as for CAN-protocol support, special message lengths, or pre-mapping hints are added. The result is a modified PIM containing elements and associations reflecting the additional information.

The main step is to assign jobs to nodes under the considerations of the functional and non-functional (i.e. performance and dependability) constraints given in the PIMs. Examples for such constraints are

- resource requirements (e.g. memory, CPU, sensors, actuators, bandwidth)
- dependability constraints (e.g. replicas must be assigned to different nodes)

A dual-track approach is taken in DECOS to generate the PSM. In the first phase, a transformation based mapping process is developed which deals constraints one-by-one. It finds a feasible solution for resource allocation while satisfying different constraints. We have explicated a heuristics based systematic resource allocation approach for this and presented it in [8]. Considering dependability and real-time as prime drivers, we presented a schedulable allocation algorithm for the consolidated mapping of SC and non-SC applications onto a distributed platform.

Although the allocation problem is NP-hard [4], exploiting symmetry (job replicas, identical nodes, ...) can improve the performance of such approaches [15], but it is difficult to assess solutions with respect to certain criteria like reliability maximisation or cost minimisation.

Therefore, in the second phase, a Multi-Variable Optimization (MVO) approach is proposed where multiple objectives are optimized together with satisfaction of constraints. Here, a so-called MVO function is used, which associates a scalar-valued function $v(q)$ to each point q in an evaluation space, representing the system designer's preferences, provided that choosing a feasible alternative from a set of contenders such that v is maximized or minimized.

See [7] for more information about the generation of the PSM.

4.2 Scheduling

The DECOS Integrated Architecture consists of several nodes that communicate via a time-triggered physical network. Virtual time-triggered or event-triggered networks are built upon this time-triggered physical network, implying that all information transfer takes place via messages of the underlying time-triggered physical network. Thus, each DECOS node must be able to send/receive messages (cf. Fig. 7). Furthermore, DECOS nodes that share the same communication medium are required to coordinate the transmission of messages, i.e., at each point in time, exactly one node is allowed to send a message.

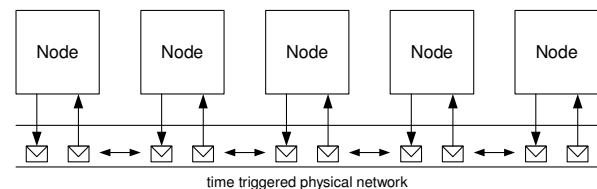


Fig. 7: DECOS nodes communicating via a time-triggered physical network

In addition to the scheduling of messages, the scheduling of EEE-partitions (see 2.1) and of applications tasks (DAS jobs may consist of several tasks sharing the same spatial and temporal resources) that run in the partitions must be provided.

For that purpose, the TTTech tool suite³ has been adapted to cope with partitions, and integrated into the DECOS tool chain.

These tools (DECOS TTPplan and TTPbuild) provide a graphical front-end for the specification of partitions as well as a programming interface that is used for importing the PSM. From that, they compute valid schedules and generate configuration data structures, i.e. MEDLs (Message Descriptor Lists) for the communication controller, C-MEDLs for the hardware implemented fault-tolerance layer, optimized fault-tolerance layer tasks, and the respective configuration files for the operating system configuration including the protection parameters of the partitions.

4.3 PIL Generation

As mentioned in 2.1, the PIL provides a technology invariant interface to the DECOS architec-

³ <http://www.tttech.com/products/software.htm>

tural services for application jobs. Following "native" services are offered by PIL:

- generic message transfer (TT and ET),
- global time service,
- membership service (to get information about health states of nodes and jobs).

In addition, the usage of domain-specific application *middleware* (AppMW) like for CAN-support is possible.

Since C is still the most common programming language for embedded systems, including generated code like that from SCADE (see next section), a C-binding for the PIL API is provided by default. One means to improve safety in a C-environment is to make intensive use of types and names, and to avoid the usage of error prone types like `void*` or `char*` for parameters and return values. Hence, for each job/message combination an own set of functions is provided, forcing to generate the PIL individually for each job.

For instance, if a job X receives a state message S of type `t_S` and may send event messages E of type `t_E`, then essentially the following C-API will be generated for it:

```
DCS_ReturnCode DCS_get_S(
    t_S *out_S,
    DCS_bool *out_validityIndicator);
DCS_ReturnCode DCS_queue_E(
    const t_E * in_E,
    const DCS_Time* in_timeout);
```

('DCS' stands for DECOS.) Some more functions are generated, e.g. for providing some control on data access, but according to the same principle.

In order to achieve platform and programming language independency, PIL generation can be controlled by the *bound PI* (see Fig. 2), which contains, for instance, coding templates for the respective platform/language pair.

5 Behaviour Modelling and Design

In addition to modelling system structure, configuration, and scheduling as described so far, specification of behaviour is another issue. Though using C-code for that purpose is of course possible, in safety-critical environments safer approaches are highly recommended. Therefore, SCADE [14] has been chosen to be the primary DECOS tool for behaviour modelling

and development, which is described in this chapter.

5.1 SCADE

Based on a formally defined data-flow notation [5], SCADE offers *strong typing*, *explicit initialization*, *explicit time management* (delays, clocks, etc), and *simple expression of concurrency* (data dependencies). By means of a graphical data flow graph editor, it supports model-based development. This not only allows for simulation at model level, accompanied by dedicated testing [3] or formal proof techniques by the SCADE Design Verifier [2] to prove safety properties, it also enables qualified code generation, using the KCG code generator. KCG has been certified against DO178B level A [13] and IEC 61508 at any SIL level.

The basic SCADE modelling elements are pre-defined *operators* and user-definable *nodes*. Both have input and output parameters, through which they are connected with other nodes and operators (see Fig. 8). Of course nodes can be nested. Fairly obviously, nodes will be used to represent DAS jobs.

In order to assure that "job nodes" adhere to their interface definition in PIM, SCADE's UML gateway is used to import PIMs into SCADE. And in order to enable usage of behaviour models developed in Simulink⁴, another SCADE gateway can be used to import Simulink models into such nodes 5.3. In the following, these gateways are addressed.

5.2 PIM-Import

The SCADE UML gateway is a flexible tool for import and reuse of software architectures specified in other modelling languages as SCADE node skeletons. It allows for easy addition of extension modules to support any modelling language that has similar architectural concepts as UML: Architectures consist of functional blocks, with one or several levels of hierarchical decomposition; and blocks are connected at specific interaction points, with corresponding communication protocols defined.

The SCADE UML gateway is being extended to support PIM-based modelling, by adding a new module specifically tailored for DECOS PIM, which realizes the following mapping:

- Each DAS represent a namespace for its contained Jobs.

⁴ <http://www.mathworks.com/>

- Jobs of a DAS are the architectural blocks of the DAS.
- Ports (grouped in Interfaces) of a Job are the interaction points of the corresponding block.
- Event-triggered and time-triggered messages form the protocols at the interaction points.

Note that in SCADE, a node can be instantiated several times. Therefore, a node is a kind of "block type". In the PIM on the contrary, Jobs are instances already. There is no notion of job-type to be instantiated several times. Anyway, a job instance can be assimilated to its implicit "singleton" job type whenever the concept of "block type" is needed.

Using the mapping realized by this PIM gateway, the core of the gateway builds an internal representation of the corresponding SCADE model, together with traceability information that points back to the original PIM model.

The PrettyPrint module of the gateway then serializes this internal representation into a valid SCADE model containing skeleton nodes for all jobs present in the PIM model.

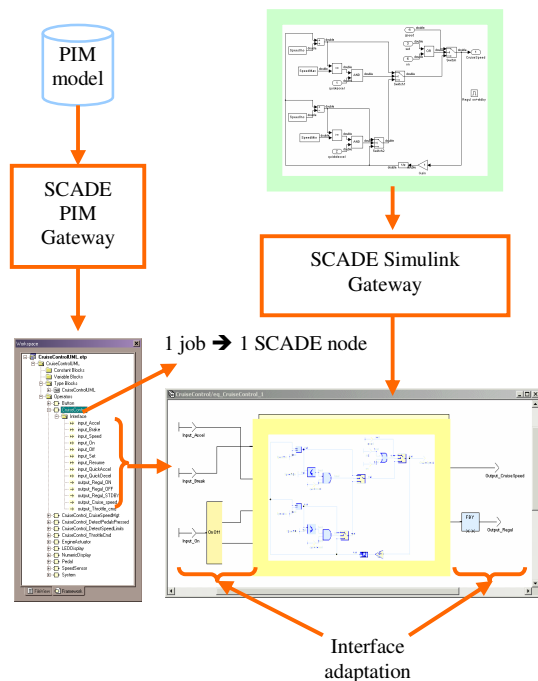


Fig. 8: Importing PIM and Simulink models into SCADE

5.3 Simulink-Import

If the behaviour of the DAS jobs is originally modelled in Simulink, it has to be imported into SCADE, to "fill the contents" of the nodes cre-

ated by the PIM to SCADE gateway. The DAS jobs behaviour can be defined in several Simulink models, and/or in several parts of a Simulink model. The current version of the Simulink translator only allows one Simulink import (one part of one Simulink model), so it is upgraded to support this requirement. When a modification of a Simulink job model is made, thanks to this "modular translation" feature, only the respective part needs to be re-imported.

Since re-translation may introduce incoherencies in the SCADE model, semantic inconsistencies (on names, types, or used clock units) are automatically checked by the SCADE checker.

5.4 Code Generation

As already mentioned, KCG is used to generate the code from each individual SCADE node job. To get the code of the complete distributed application, the job code must be linked to the middleware code via the PI API.

Since SCADE nodes do not access DECOS services via the PI API directly, but instead work on 'context objects' which contain their input and output parameter, they have to be embedded into so-called *wrappers*, which per execution

- fetch received messages and put them into the input fields of context objects,
- activate the node code,
- forward output fields of context objects to PI API as send messages.

Since node code may not produce requested output, a specific flag mechanism is implemented to inform the wrapper which messages to send.

6 Putting it all Together

As shown in the previous chapters, a rather wide variety of tools comprise the DECOS tool-chain. In order ease the handling of all these tools, VIATRA [1] is used as "backbone" with respect to PIM→PSM transformation. It not only allows for developing model transformation conveniently, it is also possible to generate code with it (e.g. PIL), as well as to develop domain-specific editors, which e.g. ease PIM creation. Although CRDs could conceptually also be developed with VIATRA, the taken approach (see 3.2) is very appealing. So, basically four tools constitute the DECOS tool-chain – GME, VIATRA, SCADE, TTPplan/build, as indicated in Fig. 9, which also

shows the interchange formats among these tools. Boxes denote interactive activities.

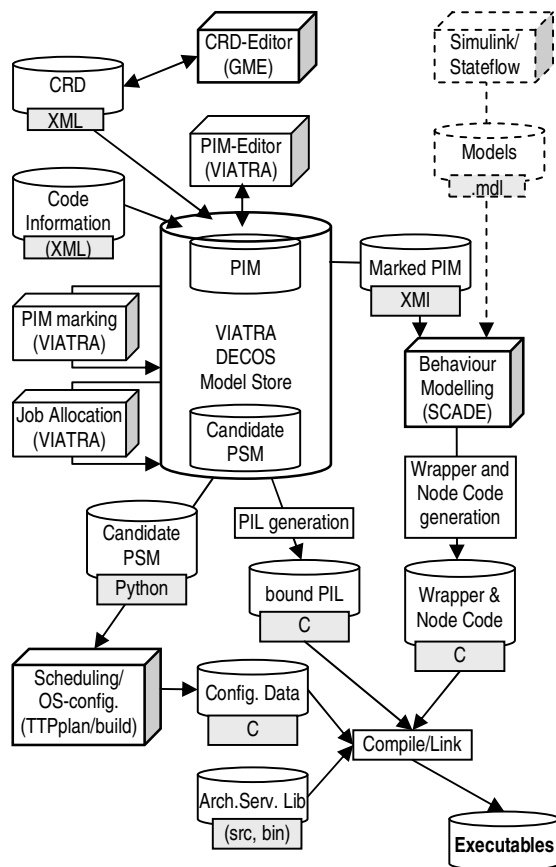


Fig. 9: DECOS tool-chain: involved tools and interfaces

7 Conclusion

A tool-chain has been presented, being under development in the integrated EU-project DECOS, which enables efficient and convenient configuration and code development for integrated embedded real-time systems, based on a model-based approach.

Besides completing the tool-chain itself, further features are intended to support simulation of complete DASs in SCADE, and easing PIM creation by means of PIM-specific patterns.

References

- [1] G. Csertan, G. Huszerl, I. Majzik, Z. Pap, A. Pataricza, and D. Varro: VIATRA: Visual automated transformations for formal verification and validation of UML models. In Proc. of the 17th IEEE Int. Conf. on Automated Software Engineering (ASE 2002). IEEE (2002), 267–270
- [2] A. Bouali, B. Dion, K. Konishi,; Using Formal Verification in Real-Time Embedded Software Development. Japan SAE, Yokohama, 2005
- [3] B. Dion, J. Gartner; Efficient Development of Embedded Automotive Software with IEC 61508 Objectives using SCADE Drive. in Proc. of VDI Conference Electronic Systems for Vehicles, Baden-Baden, Oct. 2005;
- [4] D. Fernandez-Baca, Allocating Modules to Processors in a Distributed System. IEEE Trans. on Softw. Eng., 15(11), pp. 1427–1436, 1989.
- [5] N. Halbwachs, P. Caspi, P. Raymond, and D. Pilaud. The synchronous dataflow programming language lustre. Proceedings of the IEEE, 79(9):1305–1320, September 1991.
- [6] B. Huber, R. Obermaisser, and P. Peti. MDA-Based Development in the DECOS Integrated Architecture – Modeling the Hardware Platform. Proc. of the 9th IEEE Int. Symp. on Object and Component-Oriented Real-Time Distributed Computing (ISORC). 2006.
- [7] S. Islam, G. Csertan, W. Herzner, T. LeSergent, A. Pataricza, N. Suri; A SW-HW Integration Process for the Generation of Platform Specific Models. ME'06, Oct. 2006
- [8] S. Islam, R. Lindström and N. Suri; Dependability Driven Integration of Mixed Criticality SW Components, In the proc. of the 9th IEEE International Symposium on Object and Component-oriented Real-time distributed Computing (ISORC), p 485-495, 2006.
- [9] Kopetz, H. and Bauer, G. (2003). The time-triggered architecture. IEEE Special Issue on Modeling and Design of Embedded Software
- [10] H. Kopetz, R. Obermaisser, P. Peti, N. Suri; From a Federated to an Integrated Architecture for Dependable Real-Time Embedded Systems; Technical report 22/2004, TU Vienna, July 2004.
- [11] A. Ledeczki, M. Maroti, A. Bakay, G. Karsai, J. Garret, C. Thomason, G. Nordstrom, J. Sprinkle, and P. Volgyesi; The Generic Modeling Environment; Proceedings of WISP, Budapest Hungary, May 2001
- [12] OMG. Model driven architecture, A technical perspective. Technical report, OMG Document No. ab/2001-02-04, Object Management Group.
- [13] RTCA (1992). DO-178B: Software Considerations in Airborne Systems and Equipment Certification. Radio Technical Commission for Aeronautics, Inc. (RTCA), Washington, DC.
- [14] SCADE Suite Technical and User Manuals, Version 5.0.1, June 2005, Esterel Technologies
- [15] G. Weißenbacher, W. Herzner, E. Althammer; Allocation of Dependable Software Modules under Consideration of Replicas. ERCIM Workshop on Dependable Software Intensive Embedded Systems, Porto, Portugal. Sep.2005
- [16] A. Pataricza: Report about decision on meta model and tools for PIM specification, DECOS deliverable D 1.1.1, Dec 2004.



Wolfgang Herzner received his doctor degree in Computer Science at the Vienna University of Technology in 1984. Initially working in the fields of computer graphics and multimedia, he was engaged in corresponding international standardisation within ISO/ IEC JTC1 until 1996.

After co-designing and developing large industrial systems, e.g. for digital video surveillance, he now works in the area of safe software engineering and related areas like model-driven software development or (design) patterns, with emphasis on dependable embedded real-time systems. With that focus, he is currently involved in projects like DECOS. Herzner is with the ARC Seibersdorf research since 1984. He gives lectures about component-based software engineering at the Johannes Kepler University of Linz, and is member of ACM, IEEE Computer Society, the Austrian Computer Society (OCG), and reviewer of IEEE Software Engineering.



Martin Schlager received two master degrees from the Vienna University of Technology, a Master of Science (2002) and a Master of Social and Economic Sciences (2005). In the past four years, he has been involved in several national and European research projects with a

focus on dependable real-time systems. Martin Schlager has been affiliated with TTTech Computertechnik AG since 2001. Between 2003 and 2004 he was granted a one-year scholarship by the Austrian Academy of Sciences. In that time he worked as a researcher in the Real-Time Systems Group at the Institute of Computer Engineering at the Vienna University of Technology. In 2004 he joined TTTech as an R&D project manager. His current assignment includes the coordination of TTTech's activities in the EU FP6 project DECOS.



Thierry Le Sergent received his PhD on Computer Science from the "LAAS-CNRS" in 1993. He has been involved for 7 years in the development of software engineering methods and tools (ASA+, ObjectGEODE) at Verilog and CS-Verilog. Years 2000 and 2001, he

managed the IST SafeAir project for Telelogic Technologies Toulouse. In 2002, he joined Esterel Technologies as R&D project manager leading the SCADE Gateways team. Activities of the team includes the design and development of SW tools that link SCADE environment to other formalisms, e.g. UML/SysML and Simulink/Stateflow. He is managing IST projects for Esterel Technologies: RISE (FP5), DECOS (FP6), SPEEDS (FP6).



Bernhard Huber received his diploma degree (MSc) in Computer Science from Vienna University of Technology in 2004. He is currently affiliated as research and teaching assistant with the Real-Time Systems Group of Vienna University of Technology. His research interests

focus on dynamic reconfiguration and integrated resource management of time-triggered systems as part of the EU-funded IST research project DECOS.



Shariful Islam received the B.Sc. degree in Electrical and Electronic Engineering from Khulna University of Engineering and Technology, Bangladesh in 2000 and the M.Sc. degree in Information Technology [Study Model: Embedded Systems Engineering] from the University of

Stuttgart, Germany in 2004. He is currently a PhD candidate at the DEEDS group in the Department of Computer Science at TU Darmstadt, Germany. His research interests include design and assessment of integrated dependable real-time embedded systems, system design optimization, SW-HW integration - main issues are dependability, real-time and power. He received the 'Prime Minister Gold Medal' award in recognition of his outstanding academic performance in B.Sc. He is a student member of the IEEE. More information on his interests and activities is provided at <http://www.deeds.informatik.tu-darmstadt.de/ripon>.



Neeraj Suri received his Ph.D. from the University of Massachusetts at Amherst. He currently holds the TU Darmstadt Chair Professorship in "Dependable Embedded Systems and Software" at TU Darmstadt, Germany. His earlier academic appointments include the Saab Endowed Professorship and faculty at Boston University.

His research interests span design, analysis and assessment of dependable and secure systems/software. His group's research activities have garnered support from DARPA, NSF, ONR, EC, NASA, Boeing, Microsoft, Intel, Saab among others. He is also a recipient of the NSF CAREER award. Suri serves as an editor for IEEE Trans. on Dependable and Secure Computing, IEEE Trans. on Software Engineering, ACM Computing Surveys, IJNS, and has been an editor for IEEE Trans. on Parallel and Distributed Systems. He is a member of IFIP WG 10.4 on Dependability, and a member of Microsoft's Trustworthy Computing Academic Advisory Board. More details are available at: <http://www.deeds.informatik.tu-darmstadt.de/suri>



András Balogh holds an MS in Technical Informatics received from the Budapest University of Technology and Economics in 2004. Currently he is a Ph.D. student at the Fault Tolerant Systems Research Group at BUTE.

His professional interest includes model-driven development of dependable systems and e-business systems, and formal analysis of system models. He has many years of experience in architectural system design, and in software and transformation development. He holds courses on Eclipse-based development, and UML modeling.