The Aerospace demonstrator of DECOS

Hauke Fuhrmann and Jens Koch and Jörn Rennhack and Reinhard von Hanxleden

***Abstract*— A goal of the *Dependable Embedded Components and Systems* (DECOS) project, an Integrated Project within the EU Framework Programme 6, is to explore the integrated distributed time-triggered architecture paradigm. A high-lift flap system serves to validate and demonstrate this paradigm in the aerospace application domain. This paper gives an overview of the application and the system architecture, and describes the model-based development process.**

I. INTRODUCTION

A. Time-Triggered Architecture

With the growing complexity of distributed systems the intercommunication within and between subsystems is increasing. For receiving fault-tolerance in highly safety-critical systems much effort must be spent (*e.g.*, in system design and error containment) if event-triggered communication buses such as CAN [1] are used, due to the fact that these buses are mostly developed for (soft) real-time systems with flexible requirements concerning timing. In contrast, the time-triggered communication approach in the *time-triggered architecture* (*TTA*) [2] offers deterministic, fault-tolerant communication services with additional features that enable the developers to better manage complexity and to find design flaws earlier in the development process.

As TTA is a general notion, there are different implementations available. The *Time-Triggered Protocol* (*TTP*) [3] is a communication protocol for the TTA and specifies its communication subsystem. Other approaches are forthcoming, for example *Byteflight* [4], *FlexRay* [5] and *TTCAN* [6].

The TTP describes the communication scheme of the system. The set of nodes participating in communication via TTP is called the *TTP cluster*. The access to the bus follows the *time division multiple access* (TDMA) scheme as it is depicted in Figure 1. The global communication schedule is called the *Message Descriptor List* (*MeDL*) and has to be known by every communication controller in the cluster. In a TTA, not only the communication scheme is time-triggered, but the task execution at the individual nodes, too, in order to synchronize task execution with message transfer. The local task schedules of the nodes are called *Task Descriptor Lists* (*TaDL*).

H. Fuhrmann is with Faculty of Engineering, Inst. of Computer Science and Applied Mathematics, Christian-Albrechts University of Kiel, Germany haf@informatik.uni-kiel.de

J. Koch is with Airbus Deutschland GmbH, Hamburg, Germany jens.koch@airbus.com

J. Rennhack is with Airbus Deutschland GmbH, Hamburg, Germany joern.rennhack@airbus.com

R. v. Hanxleden is with Faculty of Engineering, Inst. of Computer Science and Applied Mathematics, Christian-Albrechts University of Kiel, Germany rvh@informatik.uni-kiel.de

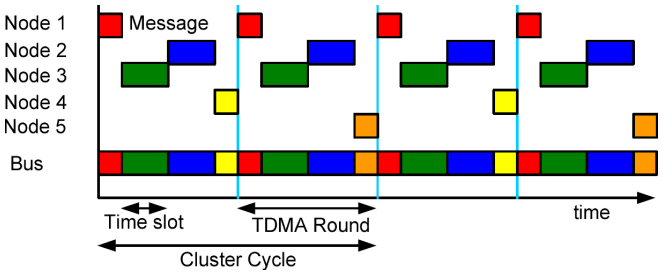

Fig. 1. The TDMA scheme of a TTP network.

B. Model-Based System Design

Another advancing key technology is the *model-based system design*. Modelling suites such as Matlab/Simulink/Stateflow [7], SCADE [8] or Mission Level Designer [9] allow to model a system prior to its physical implementation. This helps to find errors in the development process in early stages: Design faults in the specification can be revealed or the whole system can be simulated although some of the components do not yet exist physically. This helps to find errors as soon as possible and can save effort, time and money because in later design phases the costs of eliminating errors are likely to be much higher.

For best results in the industrial development process the two technologies, TTA and model-based system development, should be combined. Thus the development process leads to a model-based system design of the TTA.

C. The Dependable Embedded Components and Systems Project

The DECOS project (*Dependable Embedded Components and Systems*) [10], an Integrated Project within the EU Framework Programme 6, should deploy a process of developing embedded systems that are built on COTS (commercial-of-the-shelf) hardware and software components and nevertheless enable us to develop safe systems.

The key to the problem is to develop fundamental and enabling technologies which are independent of domain and technology in order to facilitate the paradigm shift from *federated* to *integrated* design of dependable real-time embedded systems. This shall lead to reduced development, validation and maintenance costs in both the software and the hardware domain.

The major objective is to research the compositional system framework and to develop a set of generic hardware and software components. As DECOS is platform independent, they are usable on various platforms that support the defined core services *deterministic and timely message transport*,

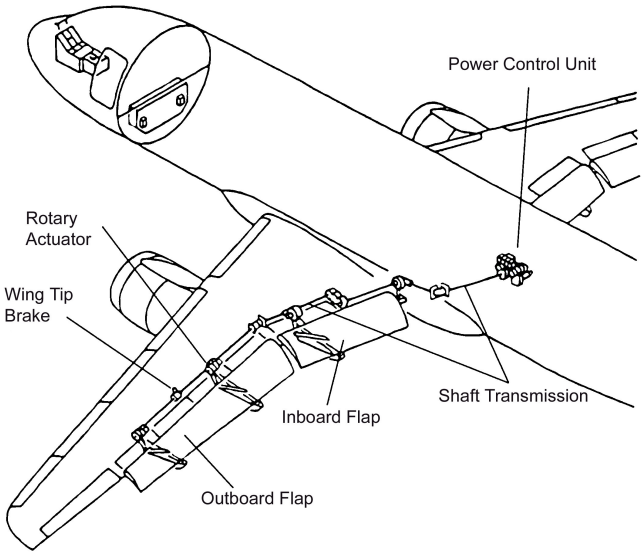

Fig. 2. The state-of-the-art high-lift flap system [11]

fault-tolerant clock synchronization, *strong fault isolation* and *consistent diagnosis of failing nodes*.

The TTA was chosen because it offers *strong composability*, *effective fault propagation barriers*, *fault-tolerance* by active replication, *strong diagnosability* and *formal analysis* of critical architecture functions. The particular choice within the aerospace subproject fell to TTP as it consequently follows the TTA paradigms and safety issues and is technically mature.

II. THE AEROSPACE SUBPROJECT

For demonstration of the research and development results, several test benches will be implemented employing the new technology. One is to validate the achieved results of DECOS and demonstrate their practicability in a highly safety-critical aerospace application environment.

The demonstrator will implement an electronically synchronized *high-lift flap system*. This system is motivated by the current state-of-the-art for such systems [11]. As depicted in Figure 2, a flap system consists of two flap panels at each wing.

The flap system can increase the concavity of the wing if activated and therefore increase the ascending force temporarily. This is used at low speed for landing and take off purposes only. Hence a flap system is safety-critical. If the system fails during the flight, it will not be available for landing, which is obviously a serious problem.

If the left and right flap panel are not perfectly synchronized, the ascending force on the two wings differ. This compromises the controllability of the plane and might ultimately lead to a crash.

To avoid the asynchronous state of the flap panels, they need to be synchronized. In the state-of-the-art flap system this is done mechanically: A tight mechanical shaft physically connects the left wing flap panel with the right wing flap panel. This shaft lies across the whole aircraft

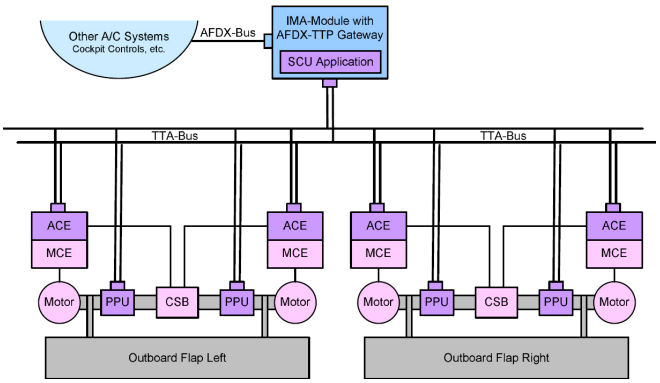

Fig. 3. The architecture of an electronically synchronized flap system

fuselage. A central power control unit actuates the shaft with the help of two electrical motors. If one motor fails, the other side will move the whole shaft with half the speed by a speed summing differential. This way both sides are always perfectly synchronized unless a shaft breaks or blocks, whereupon the shaft brakes freeze the system. However, this scenario is highly unlikely.

The drawback of this solution is obvious: The shaft across the fuselage is very inflexible and the development of the tip-to-tip shaft transmission is very laborious and includes the internal construction of the fuselage into the development of the flap system, which makes the system neither modular nor reusable.

An electronically synchronized flap system could give more flexibility by introducing modularity and might even increase safety by exchanging central control with local controls enabling more sophisticated fault reaction strategies. An electronic system has not yet been tackled, as safety has highest priority.

The demonstrator is a test platform for the aerospace domain to realize the implementation, test and integration of tools, methods and components (hardware and software) being developed in the DECOS project and to validate the achieved results and demonstrate their practicability in this highly safety-critical application environment.

Figure 3 shows the architecture of the system. Only one flap panel per wing side is shown.

The left and right side of each flap panel are still connected via a mechanical shaft. Each shaft side is powered by a powerful electronic motor. The motor is controlled via the *motor control electronics* (*MCE*) whereas *actuator control electronics* (*ACE*) control the synchronization process and form an outer control loop. *Position pickoff units* (*PPU*) measure the current angle of the flap shaft and a hydraulic *cross shaft brake* (*CSB*) is able to fix the shaft in case of faults. The ACEs and PPUs communicate via a time-triggered bus in order to exchange information for the synchronization control loop. The ACEs control the CSB and only if both ACEs on each side indicate correct functionality, the CSB is released.

A central *system control unit* (*SCU*) monitors the behaviour of the system and commands the desired flap angles.

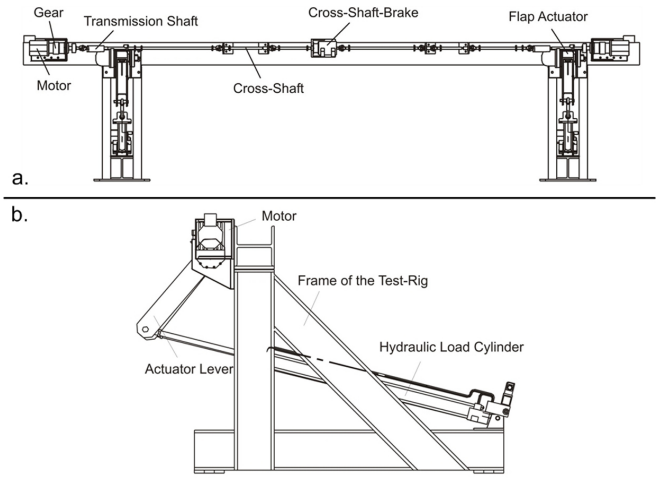

Fig. 4. The physical test rig of TUHH. a. front view, b. side view [13]

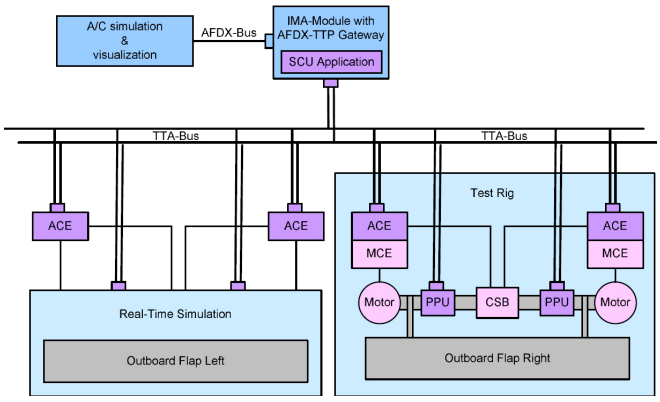

Fig. 5. The architecture of the flap system developed around the flap test rig

The SCU is implemented as an application on an IMA module which has the interface to the TTA bus on the one hand and the interface to the event triggered communication bus *AFDX* (Avionics Full Duplex Switched Ethernet) [12] on the other hand. Through the AFDX channel other aircraft systems are able to communicate with the SCU application and could even communicate directly with the TTA nodes by the AFDX-TTP-gateway functionality.

A. The test bench

A physical test rig of a flap panel exists at the Technical University Hamburg-Harburg (TUHH). The available part is depicted in Figure 4. The test rig represents the right outboard flap panel and comprises a hydraulic load cylinder to simulate the aerodynamic forces to the flap.

The rest of the system will be developed around this test rig. Hence the real system architecture will look as shown in Figure 5.

The test rig will implement the right flap side. The mechanical parts will be augmented by the position-pickoff unit and the electronic units as described above. The left flap side will be implemented as a real-time simulation: The ACEs will be available as electronic units but connect to a computer unit that simulates the behaviour of the left flap. The SCU will be implemented on a IMA module as described but the input commands come from an aircraft simulation and visualization application at a personal computer that connects it via AFDX interface to the IMA module.

III. MODEL-BASED SYSTEM DESIGN

The complexity of the systems obviously rises by introducing more and more electronic systems which intercommunicate. The system developer must address topics of safety, reliability, maintainability, availability, security and reusability.

One can try to cope with the complexity by using models in the development process. In general a model is a dedicated abstraction of the real world. It can help to explain certain real-world phenomena. Graphical representations of information play an important role in software engineering. They help to display the underlying data in a user-friendly way. Graphical modelling languages show the system on a very high level of abstraction. They can hide irrelevant data and clarify hierarchy in order to design the development process in such a way that it fits human understanding best. Good modelling suites join together capabilities of abstraction of human developers with the detailed implementation of machines. So they offer an intuitive model language to support human understanding and lead to an automated system code synthesis to support the target machines directly.

A. Purpose of the model

There are many reasons that justify to use modelling and to make a system model to a central part in the development process of the system:

1) System specification: Specifications are often written in natural language and are therefore of informal nature. Models typically have a precise semantics (depending on the modelling language used), hence transferring the textual descriptions into a model formalizes a specification. As models can be structured hierarchically, one would probably understand the essence of a system more quickly by studying a model than by reading a textual specification.

2) Computing the communication schedule: Before the different components of the TTA can finally be implemented, the communication schedule—in our case the TTP-specific Message Descriptor List (MeDL)—must be defined, because the component schedules are built according to the MeDL. Modelling tools, such as Matlab/Simulink and SCADE, offer interfaces to the TTA cluster compiling tools of TTTech. So the MeDL can be synthesized directly from the model. Modelling the TTA in an appropriate tool is much more intuitive and better to read than working with TTP-Plan directly.

3) Specifying logical behaviour: As an electronically synchronized high-lift flap system is a new concept, there are no synchronization algorithms and fault reaction strategies available from the shelf. So the development of the logical functionality of the different components must be done from scratch. Building a model and simulating its behaviour

can help to find possible failure modes and assist at the development of this logical behaviour.

4) Test, verification: Many modelling languages bestow models with such precise semantics that the model can be simulated. Hence the developers can test the system before any physical implementation is available. Therefore elimination of design faults might be done in early stages of the development process, which saves development costs. In addition, with SCADE and the *Design Verifier* the developer can prove predefined properties of the operators and *Model Coverage Analysis* might verify that every element of the model (which represents a software requirement) has been dynamically activated when the system requirements are exercised.

5) Code generation: Modelling suites like SCADE and Matlab are equipped with code generators that transform the models into a programming language, such as C. This code can be compiled with specific target platform compilers and loaded to the system components directly or with little hand coded additions. Having a model in such detail that it allows the synthesis of the program code of the components would be a great benefit of the modelling effort.

B. Esterel Technologies: The SCADE Suite

The *SCADE Suite* is the modelling environment of *Esterel Technologies* [8]. The semantics of the models are given by *synchronous languages*: A very common model in software process control is *cycle-based reaction*. The implementation cyclically repeats a sequence of three actions: reading the inputs, computing the reaction and producing the corresponding outputs. Input events occurring during a reaction are queued for the next reaction, which makes the reaction atomic and deterministic. There are two different synchronous programming styles: The data-flow style and the imperative style. These lead to the graphical modelling techniques of the SCADE Suite. An informal introduction to these synchronous languages gives [14].

1) Data flow models: Lustre [15] is the synchronous language with data-flow style which is well-adapted to steady process-control applications and to signal processing. A SCADE model is a graphical representation of such a Lustre program.

2) Safe State Machines (SSMs): State machines can be embedded into a SCADE model. Esterel Technologies uses its *Safe State Machines* (*SSM*) which were formerly known as *SyncCharts*. They were conceived in the nineties [16] as a graphical notation for the synchronous *Esterel* language [17]. Hence SSMs inherit their semantics through the mapping to Esterel. This mathematical semantics is explained in a technical report [18] and an informal presentation of the model and its semantics is given in [19], [20].

3) Code generation: SCADE provides Ada, standard C and qualified C code generators.

The SCADE Editor exports a textual description of the SCADE model which is fed into a *SCADE-to-Lustre* filter that transforms the model into Lustre code. There are three different *Lustre-to-Code* transformators, that accept Lustre code as an input and produce code in the corresponding target language, *i.e.*, Ada, ANSI C or C qualified with respect to DO-178B level A [21]. That is the most constraining level of one of the most constraining development processes, which is mandatory for safety critical systems in the aerospace industry.

C. TTP-Coupling

The Time-Triggered Protocol, TTP, has been developed during the past 25 years by Vienna University of Technology (TU Wien) [22]. In 1998 *TTTech* emerged as a company from the TU Wien and EU-funded research projects such as TTA and X-By-Wire. From then on it has commercially been using the results by developing tools for the TTA system design.

The target group are mainly the aerospace and automotive industry and comparable industries, which show a strong distribution of the development process into two parties: The *system integrator* and the *subsystem suppliers*, which often differ from the system integrator.

Therefore the TTTech tools build a two-level design framework taking this relationship into account: *TTP-Plan* is an overall cluster design tool, which specifies global cluster properties and generates a global communication schedule—the Message Descriptor List (MeDL). Whereas *TTP-Build* is a node and task design tool. It specifies local node properties, generates the local task schedule—the Task Descriptor List (TaDL), configures the operating system and generates middle ware code. There are some other tools available used here, such as *TTP-Load*, *TTP-View*, *TTP-Matlink* and *TTP-SCADElink*.

The TTA-modelling interface to SCADE is called *TTPlink*. It provides a SCADE node library with nodes for TTP messages. There are nodes for modelling the sending of messages, the receiving of messages and the passing of local messages between tasks within one subsystem. Annotations to the message nodes add the message properties, such as the message names and the sending period. TTA subsystems and tasks are modelled by normal SCADE nodes which get enriched by annotations that include the subsystem resp. task properties. Forms allow the input of these properties into the annotations. One standard SCADE node represents the TTA cluster. It only comprises TTA subsystem nodes and the message nodes that specify the communication between the subsystems. Each subsystem node only includes TTA task nodes that may be connected with local message nodes. The TTA task nodes may contain other SCADE nodes that model the functional behaviour of the task.

An additional view is for specifying the hardware units, the TTP nodes, and the mapping of these nodes to subsystems.

The TTP-coupling for SCADE adds user interfaces to automatically generating the MeDL with TTP-Plan, the TaDLs for every node with TTP-Build, the wrapper code for the tasks and the code for the applications. The compilation of application binaries and the upload of these to the TTP nodes can be controlled by the user interface as well. Finally the

calibration of values through the monitoring node can be enabled by this user interface.

IV. THE AEROSPACE VIRTUAL TEST-BENCH

The Virtual Test-Bench that is to be developed will be an abstract simulation model representing the physical Aerospace Test-Bench. It permits to simulate the overall test-bench behaviour, integrate and test the functional components (HW/SW) of all project partners in this model and enables to validate the design at a high level of abstraction. The architecture design tools and validation methods developed in DECOS will be applied, if applicable. With aid of the simulation model a concept validation of the physical test-bench will be done before developing the individual components. Finally, the physical test-bench will be validated and tested against the virtual simulation model.

A. The Virtual Test-Bench

1) Scope of the model: The main scope of the modelling is the time-triggered communication. So all components that have access to the TTA bus are modelled at first. These are

- The four *Actuator Control Units* (*ACE*) which control the motor control electronics and therefore directly influence the motor motion which moves the flap panel.
- The four *Position Pickoff Units* (*PPU*) that detect the current position of the flap panel and provide the values on the TTA bus.
- The *System Control Unit* (*SCU*) that gets positioning commands from the cockpit resp. the aircraft simulation via the AFDX bus.

For simulation traces that provide significant results, the model has to be enriched by more detailed information. As the main purpose of the system is to synchronize the left and right flap panel, it appears reasonable to model the flap panels, too. Although the mechanical functionality is not the main scope of this model, it would be very helpful to have such information about the panels. This includes the functionality of the *Motor Control Electronics* (*MCE*) and the mechanical, electric and hydraulic parts of the flaps, *i.e.*, the main shaft, beds, the *cross shaft brake* (*CSB*), the electric motors and the mechanical connection of the PPUs.

The cockpit commands coming via the AFDX bus to the SCU will be implemented by textual flight scenario traces that can be fed into the simulation directly to the SCU. The virtual test-bench uses the same flight scenario traces for the simulation of the model as for the simulation of the real test bench.

Figure 6 shows the scope of the model.

2) Contents of the model: The virtual test-bench comprises three levels with different kinds of information:

1.) The first level contains the global cluster information. That is the specification of the different subsystems and the TTP messages, incoming and outgoing. A TTP message may be produced by exactly one but consumed by multiple subsystems. Additionally the hardware nodes of the cluster are specified. In our case each subsystem gets implemented by exactly one hardware node (called *host* in SCADE to avoid the name clash between *SCADE node* and *TTP node*). This level of information is enough to generate the Message Descriptor List (MeDL). The interface to the scheduling tools of TTTech allows to generate the MeDL with this information automatically.

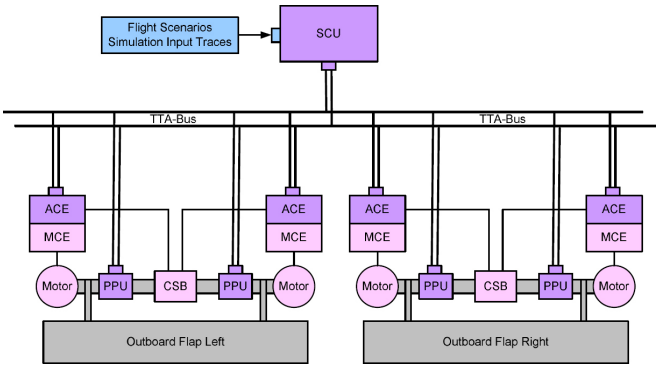

Fig. 6. Scope of the model

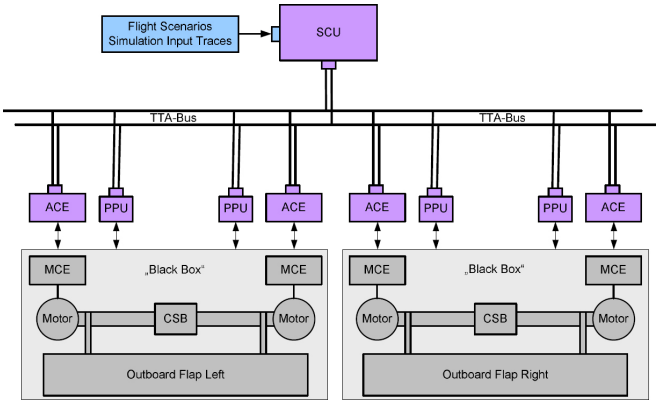

Fig. 7. The mechanical subsystem as a black box

2.) The second level contains the task information. Each subsystem only includes tasks and local messages between tasks. Task properties such as task name and time budget are specified here. SCADE can export this information to the task scheduler of TTTech which merges it with the global cluster information (*e.g.*, MeDL) in order to form the fault-tolerant communication level (which is not used here) and the task schedules of the nodes (TaDL).

3.) The third level comprises the functional behaviour of the tasks. The implementation is not as simple as in a purely functional model. The tasks have to be completely separated. No other data transfer between tasks is provided by SCADE apart from the TTP messages. Therefore modelling of the environment beyond task borders is hardly possible. Our solution will be to model the environment with the mechanical properties of the flaps in Matlab Simulink (done by TU Hamburg-Harburg) and import this model to SCADE as a binary C library in order to protect the intellectual property of the Simulink model. The architecture of the virtual test-bench is depicted in Figure 7.

V. RESULTS AND OUTLOOK

For the aerospace industry as well as for other industrial sectors, the results of the DECOS project are needed to provide a fundamental and comprehensive basis for future innovations within utility systems, communication architectures and appropriate avionics applications for the next generation of aircraft and re-designs.

Smart and distributed components, gateways to connect asynchronous and synchronous communication architectures, and high-speed communication combined with the strong fault-tolerance, reliability and deterministic behavior of the time-triggered architecture (TTA), are the necessary basis for the development of complex systems in highly safety-critical environments.

In order to reduce the effort during qualification, modification and customization, appropriate methodologies and tools are required, which also support diagnostic and system configuration to reduce maintenance and operating costs for safety-critical utility systems.

The deterministic, fault-tolerant philosophy of the time-triggered architecture fits the deterministic philosophy of synchronous languages. The synchronous languages that underlay SCADE and SSMs give precise mathematical semantics to the graphical models and therefore mathematical analysis of such models is possible and the behaviour is fully deterministic, which is essential for safety-critical applications.

In DECOS appropriate architectures, methodologies, associated COTS hard- and software components and comprehensive tool chains will be developed which fulfil the requirements of the aerospace industry. These components and tools will cover: cluster design, middleware and code generators, validation and certification as well as systems-on-a-chip (SoCs) for high dependability applications. Furthermore technology invariant software interfaces and encapsulated virtual networks with predictable temporal properties such that application software can be transferred to a new hardware and communication base with minimal effort (legacy reuse) will be developed.

REFERENCES

[1] *ISO 11898. Road Vehicles—Interchange of digital information—Controller area network (CAN) for high speed communication*, International Standards Organisation, 1993.

[2] H. Kopetz and G. Bauer, "The time-triggered architecture." *Proceedings of the IEEE*, vol. 91, no. 1, pp. 112–126, 2003.

[3] *Time-Triggered Protocol TTP/C High-Level Specification Document Protocol Version 1.1*, 1st ed., TTA-Group, November 2003.

[4] J. Berwanger, M. Peller, and R. Griessbach, "byteflight - A new protocol for safety critical applications," in *Proc. FISITA 2000 Fédération Internationale des Sociétés d'Ingénieurs des Techniques de l'Automobile*, June 2000.

[5] R. Belschner, R. Mores, G. Hay, J. Berwanger, C. Ebner, S. Fluhrer, E. Fuchs, B. Hedenetz, A. Krüger, P. Lohrmann, D. Millinger, M. Peller, J. Ruh, A. Schedl, and M. Sprachmann, "FlexRay: The communication system for advanced automotive control systems," in *SAE 2001 World Congress*. Detroit, USA: Society of automotive Engineers, Mar. 2001.

[6] T. Führer, B. Müller, W. Dieterle, F. Hartwich, R. Hugel, and M. Walther, "Time triggered communication on CAN," http://www.can-cia.org/can/ttcan/fuehrer.pdf, 2000.

[7] The Mathworks, Inc., "Company homepage," http://www.mathworks.com.

[8] Esterel Technologies, "Company homepage," http://www.esterel-technologies.com.

[9] MLDesign Technologies, "Company homepage," http://www.mldesigner.com.

[10] DECOS - Dependable Components and Systems, "Research project homepage," https://www.decos.at/.

[11] T. Neuheuser, B. Holert, and U. B. Carl, "Elektrische Antriebssysteme für ein zentrales Landeklappenelement," in *Deutscher Luft- und Raumfahrtkongress*, vol. DGLR-JT 2002-192, Stuttgart, 2002.

[12] *ARINC 664, Aircraft Data Networks, Part 7 — Deterministic Networks*, ARINC, Annapolis, Maryland, USA. [Online]. Available: http://www.arinc.com

[13] H. Geilsdorf, "DECOS - SP6 Test-Bench Design," 2005.

[14] G. Berry, "The foundations of Esterel," *Proof, Language and Interaction: Essays in Honour of Robin Milner*, 2000, editors: G. Plotkin, C. Stirling and M. Tofte.

[15] N. Halbwachs, P. Caspi, P. Raymond, and D. Pilaud, "The synchronous data-flow programming language LUSTRE," *Proceedings of the IEEE*, vol. 79, no. 9, pp. 1305–1320, September 1991. [Online]. Available: http://citeseer.nj.nec.com/halbwachs91synchronous.html

[16] C. André, "Representation and Analysis of Reactive Behaviors: A Synchronous Approach," in *Computational Engineering in Systems Applications (CESA)*. Lille (F): IEEE-SMC, July 1996, pp. 19–29. [Online]. Available: http://www.i3s.unice.fr/~andre/CAPublis/Cesa96/SyncCharts_Cesa96.pdf

[17] F. Boussinot and R. de Simone, "The ESTEREL language. another look at real time programming," *Proceedings of the IEEE*, vol. 79, no. 9, pp. 1293–1304, September 1991.

[18] C. André, "SyncCharts: A Visual Representation of Reactive Behaviors," I3S, Sophia-Antipolis, France, Tech. Rep. RR 95–52, rev. RR (96–56), Rev. April 1996. [Online]. Available: http://www.i3s.unice.fr/~andre/CAPublis/SYNCCHARTS/SyncCharts.pdf

[19] C. André, "Semantics of S.S.M (Safe State Machine)," I3S, Sophia-Antipolis, France, Tech. Rep., 2003. [Online]. Available: http://www.esterel-technologies.com/v3/?id=50399\&dwnID=48

[20] R. v. Hanxleden, "Modellierung Reaktiver Systeme - Statecharts und Synchrone Sprachen," in *Software Engineering für Eingebettete Systeme*, P. Liggesmeyer and D. Rombach, Eds. Spektrum Akademischer Verlag, 2005.

[21] *Software Considerations in Airborne Systems and Equipment Certification*, RTCA/EUROCAE, Dec. 1992.

[22] H. Kopetz and G. Grünsteidl, "TTP - a time-triggered protocol for fault-tolerant real-time systems," Institut für Technische Informatik, Technische Universität Wien, Treilstr. 3/182/1, A-1040 Vienna, Austria, Tech. Rep., 1992.