Network of Excellence

IST-004527 ARTIST2:
Embedded Systems Design

Activity Progress Report for Year 1

JPRA-Cluster Integration:

Component Modelling and Composition

Cluster:

Modelling and Components

Activity Leader:

Bengt Jonsson (Uppsala University)

The development of a general framework for component-based engineering of complex heterogeneous systems is one of the grand challenges in information sciences and technologies. This JPRA will combine the efforts and skills of world-leading researchers in Europe to address this challenge.

Table of Contents

1. Introduction

1.1 Activity Leader

Team Leader: Bengt Jonsson (Uppsala)
Areas of his team's expertise: compositional modelling.

1.2 Policy Objective

The development of a general framework for component-based engineering of complex heterogeneous systems is one of the grand challenges in information sciences and technologies. This JPRA will combine the efforts and skills of world-leading researchers in Europe to address this challenge.

1.3 Industrial Sectors

Relevant for most industrial sectors.

The automotive sector is particularly important, given the particular market pressure. Example forecasts indicate a 150% growth rate for electronic automotive components during the next 10 years. An example of an ongoing effort in the European automotive industry is the project EAST-EEA with participation of all major European car manufacturers, suppliers and software-tool providers, as well as research organizations and universities with connections to the automotive industry. Another currently pursued initiative is AUTOSAR, involving key OEMs and supplier companies. These projects expect to produce output by 2005-06.

Also in other sectors, including Aerospace, Telecommunication, and Industrial Automation, the problems addressed by this activity are central for managing the complexity of building heterogeneous embedded systems.

2. Overview of the Activity

2.1 Artist Participants and roles

Team Leader: Noël Plouzeau (INRIA)
Areas of his team's expertise: extra-functional properties specification and testing.

Team Leader: Bengt Jonsson (Uppsala)
Areas of his team's expertise: compositional modelling.

Team Leader: Joseph Sifakis (VERIMAG)
Areas of his team's expertise: composition of timed systems.

Team Leader: Jean-Pierre Gallois (CEA)
Areas of his team's expertise: testing component modelling.

Team Leader: T. Coupaye (FTR&D)
Areas of his team's expertise: Component-based middleware.

Team Leader: Tom Henzinger (EPFL – affiliated partner)
Areas of his team's expertise: modelling and interfaces for real time systems.

2.2 Affiliated partners, Roles

Team Leader: Marius Minea (Timisoara)
Areas of his team's expertise: extraction and abstraction of interfaces.

Team Leader: Alan Moore (ARTiSAN Software)
Areas of his team's expertise: UML standard evolutions.

Team Leader: Matthias Grochtmann (DaimlerChrysler)
Areas of his team's expertise: Specification, Design and Implementation of automotive systems.

Team Leader: Christer Norstrom (ABB)
Areas of his team's expertise: non-functional properties in industrial control systems.

Team Leader: Ivica Crnkovic (MdH)
Areas of his team's expertise: Component models for embedded systems.

Team Leader: Julio Medina (Univ. of Cantabria)
Areas of his team's expertise: connection with scheduling.

2.3 Starting date, and expected ending date

Starting date September 1st, 2004

Expected ending date: When a coherent framework for handling non-functional properties in design of component-based systems has been developed.

2.4 Baseline

Existing component models and frameworks do not adequately support essential properties of real-time systems, such as heterogeneity, resources, behaviour, timing, and quality of service. This problem is only partly addressed in existing collaborations between partners, e.g., within projects ARTIST, AIT-WOODDES, OMEGA, QCCS, SAVE, Families, EAST-AEE and Trusted Components.

Partners are developing important contributions towards a framework for component-based development, including

- Techniques and tools for specifying and analyzing timing properties of components and systems (timed automata with variations, IF, Kronos, UPPAAL, etc.)
- Building blocks of a general theory for building heterogeneous systems (e.g., IF)
- Technology for contracts and their monitoring

2.5 Technical Description

Develop concepts and techniques for:

- representing properties of resource usage, behaviour, timing, quality of service;
- composing components while guaranteeing such properties;
- checking whether components are compatible with each other with respect to such properties.

Develop methods for automatically generating abstract timed interfaces from system models with more detailed timing

Develop a general theory for building heterogeneous systems

2.6 Work achieved in the first 6 months

The first cluster meeting (Jan. 2005) **identified and agreed on several key issues** in techniques for modelling and analyzing QoS properties of components:

1. The principle to separate, as much as possible, different types of properties, e.g., to separate timing properties, CPU resource properties, power consumption, consumption of other types of shared resources, etc., from each other. Then each type of property can be handled by a suitable modeling and specification technique. This is a central message in the work by OFFIS [Damm, Votintseva et al. 05], where specification and analysis of components are considered in the context of automotive applications.

   Such a separation of concern is already provided in the Real-Time Calculus, presented at the meeting by L. Thiele (ETHZ). In this calculus, computation resources are “first class citizens” and specified separately from workload and timing properties.

2. To be able to separate, and express a flexible relation between, required and provided properties (of different types). This is also a central message in the work [Damm, Votintseva et al. 05] by OFFIS.

   The work on *timed interfaces* (by de Alfaro, Henzinger, Stoelinga), presented by the two latter persons at the January meeting, introduce such a clear separation.

3. In order to model and analyze timing properties using general timed automata techniques, techniques must be developed for simplifying a complex timed automaton specification into one of smaller (possibly user-defined) complexity. Otherwise timed automata descriptions of systems of components will grow in complexity and become unmanageable. Marius Minea (Timisoara) took on the task to survey techniques for simplifying timed automata-based specifications.
4. Tools for analysing systems of components specified as timed automata, such as *UPPAAL/TIMES* and *IF/Kronos* can be used for analysis also of less general forms of specifications. Thus, the work should be focused on the problem of finding appropriate ways to *specify* components. Analysis can then be provided through a translation from such specifications to timed automata.

Surveys were prepared for and presented at the January cluster meeting

1. Marius Minea (Timisoara) presented a survey on results for timed models that support compositionality and separation between required and provided properties [Minea 2005a]. Identified issues were the selection of an appropriate timing formalism, giving a trade-off between expressivity and analysis efficiency, and analysis of compatibility between provided and required properties. None of the presented frameworks really offered a general technique for avoiding the scalability problems in analysing systems of timed automata.

3. Activity Progress Report

3.1 Work achieved in months 6-12

Surveys have been prepared, some of which were presented at the June meeting.

- MdH has made a classification of different types of extra-functional properties [Crnkovic, Larsson, Preiss], based on the way in which properties of systems can be predicted from corresponding properties of its components.
- Timisoara presented a survey on existing techniques for simplifying a complex timed automaton specification into one of smaller (possibly user-defined) complexity, as agreed at the January meeting [Minea 2005b]. A potential direction ahead was identified as combining two previous approaches: one based on timed language inclusion, and another on predicate abstraction for timed zones.
- I. Crnkovic (MdH) has filed (together with M. Chaudron, Eindhoven) a proposal with a publisher to write a textbook on "*Component Based Software Engineering", [Chaudron, Crnkovic, 2005]* which will cover different aspects of component models and component-based software development.

As a result of the January 2005 meeting, the team of L. Thiele (ETHZ) extended their Real-Time Calculus approach to express assumptions and guarantees, allowing constraints on timing and CPU consumption to be propagated between requirements and platform resources, with some impressive benchmark results. Future collaboration with the "Execution Platform" cluster will be maintained.

The June 2005 meeting provided several important insights into the problems of how to build a theory and tools for design of heterogeneous systems, by means of comparing and contrasting the approaches of Metopolis and of IF.

In view of the many existing proposals for expressing timing properties of components and systems, to some extent arising from different needs, it was proposed (at the June meeting) to produce a catalogue of QoS properties. Such a catalogue would contain a classification of QoS properties, related to timing, with associated techniques for expressing and reasoning about them. A draft of the catalogue would be produced during Autumn 2005, organized by J. Medina (Cantabria).

3.2 Recommendations

The conducted meetings have shown that the problems addressed in this activity are central to several activities, also in other clusters, of ARTIST2. Another observation is that solutions to the challenges are better proposed in a specific design setting (such as synchronous language, specific system architecture, etc.). It is therefore important that the activity develop stronger links to other ARTIST2 activities, which may also motivate some restructuring of ARTIST2 organization.

3.3 Milestones

As a result of several ARTIST2 meetings, the fact that the topics of this activity are central to many areas in embedded system design has become apparent. Several research links have been established that have already generated interesting research publications, and will continue to do so.

3.4 Main Funding

The activity will be partly funded by the CARROLL initiative, a common research program between Thales, CEA and INRIA. In addition, support will be provided by partners projects on component modelling such as: Families (for CEA, INRIA), ITEA European project on component based modelling of product lines; STACS (for CEA), French national RNTS project on validation and testing of component based models.

Uppsala research is funded through national programs (Science Council) and basic university funding.

3.5 Indicators for Integration

The topic should become a focus of research for the involved teams.

A common understanding of the important issues should be developed. This will be evidenced by significant scientific papers jointly written by partners, and by transferring research results back into the modelling platform.

3.6 Evolution

The connections established during the first year of ARTIST2 operation will result in joint research results, significantly advancing the state of the art, during the next 6-12 months.

4. Detailed Technical View

4.1 Brief State of the Art

Future design processes for embedded applications must provide drastic levels of re-use and support extensive use of library components in order to meet market productivity targets, while levelling or even decreasing development costs. Key requirements for component models that support such level of reuse include widely adopted component models that cater for the multitude of functional as well as non-functional constrains. The non-functional constraints range from resource constraints (including processor resources, execution time, communication bandwidth, memory, and power) to QoS and real-time constraints.

This ARTIST2 activity addresses the challenge of developing component technology for embedded systems that supports specification and prediction of real-time and QoS-properties. It is widely recognized that such technology should be based on a **rich component model** [BBB+00], which allows to model, specify, and predict timing, QoS, and resources properties of components and of systems composed from components. Support should be provided for

- **Rich component specifications,** which include specifications of timing and QoS properties of components, as well as constraints and requirements on resources,
- **Composability,** answering questions such as *"will a new component cause deadline violations?"* and allowing incremental V&V of incrementally built systems,
- **Compositionality,** inferring global properties from component properties.

In the landscape of currently adopted component technologies, those who are widely adopted, such as EJB [JF00], .NET [Mic01], COM [Mic95], etc. contribute to structuring of system development, offer infrastructure, middleware, and tool support that solve tricky problems of component composition and communication. They allow a separation between the component development and system development processes. However, they do **not** give adequate support for handling QoS and resource properties, in a way that they can alleviate integration problems or support system predictability.

Within the embedded systems domain, more specialized technologies have been developed, which provides some limited support for handling QoS and resource usage, but only in rather limited situations. A typical such technology can be exemplified by the *Rubus* component model [IN02]. This model is designed for a specific kind of operating system and a specific type of platform, in this case a Fixed Priority RTOS executing on a single CPU. Specification of component properties is constrained to be of a form that suits a specific scheduling analysis technique: in this case components are specified as periodic tasks, with parameters that can be used by a specific analysis technique for fixed priority scheduling. A solution like this is a help for the specific development environment for which it is designed, but does not support interoperability between different component models, nor does it support change in system architecture, e.g., moving to a distributed architecture.

There are several languages that are specifically designed to express QoS and resource properties of components. Examples include QML [FK98], QuO/CDL (http://quo.bbn.com), AQuA, and AQML [Nee91]. The definitions of these languages do not include a precise semantics. Restricted forms of analysis can be performed, based on an intuitive understanding (e.g., to infer that a component with the QoS property *"at least 500 operations per second"* implements a requirement stating *"at least 300 operations per second"*). Several research efforts use these languages for the expression of QoS properties (an example being the QCCS project), but the interpretation of the syntax (e.g., to generate QoS monitors) remains ad hoc. Consider, e.g., the problem of interpreting the QoS requirement *"the component has a mean capacity of 500 operations per second".* How much slack should be allowed over how short a period when interpreting such a specification? this type of questions is not answered by existing semantics, although a specific implementation must of course resolve it in some ad hoc way.

An approach which avoids the limitations of specific technologies, like *Rubus,* and can give more precise semantics to QoS properties, is to use establish semantic models for specification of QoS properties. To specify timing properties, different variants of timed automata can be used. This has been done, e.g., in the work of *timed interfaces* (de Alfaro, Henzinger, Stoelinga) [dAHS02], and in the work on the *Omega* component model [DJPV05], which have developed a semantics in terms of the IF language, supported by timed automata. For other types of properties, e.g., relating to queuing and performance, models based on queuing networks, Markov chains, etc. have been used. These approaches offer a precise mechanism for specifying and analysing QoS properties. A potential problem is that analysis may not always scale to systems with large numbers of components. For instance, standard schedulability analysis for simple fixed priority scheduled systems typically scales better to large numbers of components than does analysis of systems whose components are specified in detail by timed automata.

Still missing is a framework that allows to use established semantic models, which

- allow a clear separation between required (assumed) and provided (guaranteed) properties, while also addressing the composability and compositionality problems.
- allow to express dependencies between different kinds of non-functional properties (e.g., between available CPU power and latency),
- allow analysis of system properties to scale well with increasing system size.

A key characteristic of component-based embedded systems is **heterogeneity** of component models. This heterogeneity concerns different execution models (synchronous, asynchronous, vs. timed), communication models (synchronous vs. asynchronous), as well as different scheduling paradigms. Technology must be provided to allow designing heterogeneous embedded systems from diverse types of components, and allow predicting and optimizing functional and non-functional properties of the designed systems. There are design tools, in which systems are designed by putting together pieces that might be termed components. Examples are MetaH (http://www.htc.honeywell.com/metah), Ptolemy (http://www.ptolemy.eecs.berkeley.edu), and Metropolis (http://www.gigascale.org/metropolis). The functions of these tools are in some sense analogous to, e.g., MATLAB/Simulink. The advantage is that they support a variety of design notations. However, "components" can be assembled only in the supporting tool, meaning that different developments must all be developed in the same environment.

For the design of heterogenous systems, design environments are being developed, intended to support the composition of systems from a variety of heterogeneous systems. An example is Metropolis, which decouples the specification of orthogonal system aspects (computation vs.

communication, functionality vs. architecture, behaviour vs. performance), and achieves integration by mapping different specification formalisms to a common Model of Computation, by means of which designs can be evaluated by simulation. However, in order to address, e.g., the issues of composability and compositionality we need to develop a coherent **theory for building complex heterogeneous systems.** Such a theory is missing today, thereby making it difficult to understand how to build systems that combine, e.g., synchronously and asynchronously executing components and reason about non-functional properties.

4.2 Industrial Needs and Experience

The technology for rich component models and heterogeneous system is crucial to many sectors in embedded systems industry.

Automotive industry In the automotive industry, systems becomes more complex as the number of ECUs increases. The system functions, controlling particular aspects at the system level (for example cruise control) require input and output control of many components. This requires sharing different types of resources (time, communication, memory and CPU consumption). With increasing complexity, system reliability and safety become major problems. A satisfactory handling of safety-critical functions, such as emerging brake- and steer-by-wire systems, will require the integration of methods for establishing functional and temporal correctness for each component, as well as system-wide attributes such as safety and reliability.

An example of an ongoing effort in the European automotive industry is the project EAST-EEA (http://www.east-eea.net) with participation of all major European car manufacturers, suppliers and software-tool providers, as well as research organizations and universities with connections to the automotive industry. The goal of EAST-EEA is to develop a structure for the next generation of electronic automotive features. There are two main activities to achieve this goal. (1) Specification of middleware suitable for the automotive industry, and (2) development of an Architecture Description Language (ADL).

The separation of functional architectures and deploymenet architectures is also central to the approach currently pushed by the AUTOSAR initiative (http://www.autosar.org), involving key OEMs and supplier companies.

Industrial automation The domain of industrial automation comprises a large area of control, monitoring and optimization systems. Many control systems are manufactured in rather large volumes, and must be configurable to suit a variety of customer contexts. The core part of a control system or a robot is typically a real-time control system that runs on a simple RTOS, or even without any OS. Other parts, such as I/O and communication protocols are in many cases provided by suppliers. The system has to be open to allow easy integration of new functionalities. Since the software usually survives many generations of hardware, it must be easy to port. Component-based development has been practised for many years, using standards such as IEC 61131.

In comparison with the situation in the automotive domain, one can roughly say that the lower layers of industrial control systems are similar in structure but, at least until now, the interoperability requirements are higher and the life cycles longer. The problems of growing system complexity together with the requirements on open, upgradeable, highly dependable and distributed systems pose many challenges which are in fact the central issues of component-based development in general.

- System integration is today a central problem in development. Component models including the possibility to specify predictabiliey, quality of service, and reliability related properties as well as tools supporting them, are lacking.
- An efficient use of components requires tools for system development with components; in particular tools for predictable component composition.
- Systems at the process control level must be able to communicate to different types of field devices and use different protocols. For this reason, it is important to define standards that contain more information than general purpose standards or tools. This means that interoperability between different application domains and different component models are required.

Consumer electronics Consumer electronics products are developed and delivered in form of *product families* which are characterized by many similarities and few differences and in form of *product populations* which are sets of products with many similarities but also many differences. The diversity of products is achieved by inclusion of different components into a common architecture. Similarly as in the automotive industry, product development is integration-oriented; that is, products are built by integration of components and new features (i.e. products) are achieved by integration of new components.

Presently, the component models used in consumer electronics support only rudimentary analysis and prediction of extra-functional properties of the components and systems. There are increasing requirements for developing methodologies for reasoning about system properties derived from the component properties. Typical requirements are prediction of QoS, memory, CPU, and power consumptions.

Telecommunication Software Telecom applications involve several domains, such as commercial information systems, network management, service management and real-time network and execution platforms. A main requirement in the telecommunication domain is that service design and development needs to be fast. Components play and have played a crucial role, and the majority of these components are embedded in core network platforms or several types of devices: mobile, fixed, etc. Components may be shared between different applications. These applications are in general not deployed at the same time but are continuously added and modified, inducing a large amount of work for functional integration, but also and even mainly, for performance integration. Furthermore, components - or their specifications - are reused when new services are developed in order to reduce the development cycle and to allow a large commercial diversity with a relatively small technical diversity, the interest of components goes beyond interoperability. Service components have individual requirements that might be violated when composed and deployed with other service components. This problem, well-known in the telecommunication world as the *service interaction* problem, must be tackled taking into consideration including real-time and performance aspects.

Mobile devices have to tackle several critical constraints (memory size, energy consumption, time constraints, etc.). They require continuous adding, removing or modification of components, and different service negotiation procedures. Security and availability are requirements in any kind of environment (unreliable environment, different kinds of communication modes, different performance properties).

Avionics and Aerospace applications are highly safety- and mission-critical and must be able to satisfy very hard real time constraints. Some of theses systems have an extremely long lifetime (over 20 years for an airplane) and will undergo several generations of upgrades and platform migrations. Also the amount of software in this kind of systems has been dramatically increasing. For example, in 1974, an Airbus 300B embedded just 500 Kbytes, tomorrow Airbus

380 will embed 64 Mbytes and in 2015, a Gbyte of embedded software is probably not a limit. In space applications, the trend is similar.

Since flight testing is extremely costly, model-based approaches with simulation and verification are more advanced and applied than in other domains, e.g., as witnessed by the prominence of the avionics application domain in many advanced technology projects (e.g., SafeAir http://www.safeair.org/project/, Mobies http://www.liacs.nl/marcello/ mobij.html, and others).

Presently, the approach for building a flight controller is a synchronous approach. The verification of the integrated system is a major problem. Emphasis is placed on predictability of global system properties and global system architecture. There is a prominent desire is to continue the trend towards model-based development, supporting it by integrated tool chains that can perform analysis of properties like fault tolerance, timing, utilization, quality of service, etc. on models, and thereafter generate optimized code for target platforms.

A major challenge in the domain is the adoption of a truly component based approach. The encapsulation of functionalities concerning distribution, security, replication, in a middleware consisting of components with guaranteed non functional properties will be the key for making existing validation methods (applied today to the synchronous model of the control) applicable to an integrated system. In order to make this vision a reality, appropriate formalisms for representing high level views of a given system architecture, including properties of components need to be built. There is an ongoing new development of a standard called AADL (Avionics Architecture Description Language, http://www.aadl.info), which has emerged from MetaH. In AADL, there exists a notion of connector, which need to be made general enough to represent a middleware component guaranteeing secure or timely communication, etc.

Summary

Component-based development allows integration problems to be handled in the earlier phases of system design. Component properties that have global system impact, notably properties of timing and resource consumption, can be specified in interfaces in such a way that global resource usage can be predicted *a priori*, avoiding hard problems in system integration. Ther are several technical problems that must be overcome to make this technology successful.

- Composition and integration of component-based systems requires technology for specification of interfaces to be developed to the point that it can *a priori* guarantee component interoperability. This is important, e.g., in the telecommunications domain
- Embedded systems are typically resource constrained. This is a further motivation why component technology must support specification of extra-functional properties (resources), so that system resource needs can be predicted and managed early in system development.
- Interoperability between different component technologies is important. The integration of heterogeneous systems, comprising components designed in different paradigms, should be supported.
- Embedded systems are typically developed over a long time, implying that support for maintenance of system evolution is an important consideration. The appropriate level of specification of component and system properties should allow system hardware and platforms to be exchanged and upgraded, as well as allowing components to be reused in different contexts. This motivates an increased interest in model based approaches to specification and development.

References

[BBB+00] F. Bachmann, L. Bass, C. Buhman, S. Comella-Dorda, F. Long, J. Robert, R. Seacord, and K. Wallnau. Technical Concepts of Component-Based Software Engineering, Volume II. Technical Report CMU/SEI-2000-TR-008, Software Engineering Institute, Carnegie-Mellon University, May 2000.

[DJPV05] Werner Damm, Bernhard Josko, Amir Pnueli, Angelika Votintseva A discrete-time UML semantics for concurrency and communication in safety-critical applications. In *Science of Computer Programming*, 2005

[dAHS02] L. de Alfaro, T.A. **Henzinger**, M. **Stoelinga**. Timed Interfaces, in Proc. EMSOFT 2002. LNCS 2491, pp. 108-122.

[FK98] S. Frolund and J. Koistinen. Quality-of-Service specifications in distributed object systems. Distributed Systems Engineering, 5(4):179–202, Dec. 1998.

[IN02] D. Isovic and C. Norström. Components in real-time systems. In Proc. RTCSA 2002, 8th International Conference on Real-Time Computing Systems and Applications, Tokyo, Japan, March 2002.

[JF00] H. Jubin and J. Friedrichs. Enterprise JavaBeans by Example. Prentice Hall, New Jersey, 2000.

[Mic95] Microsoft Corporation. The component object model specification, 24th ed., Oct. 1995

[Mic01] Microsoft Corporation. .NET framework developer's guide. http://msdn.microsoft.com/library/default.asp, 2001

[Nee91] S. Neema. System-Level Synthesis of Adaptive Computing Systems. PhD thesis, Vanderbilt University, May 1991.

4.3 Ongoing Work in the Partner Institutions

The work in this activity addresses the two challenges

- Developing the foundations for a **rich component model**, in particular with respect to **timing and QoS properties** and resource usage properties. Support should be provided for component specifications, composability, and compositionality,
- Developing a theory for composing **heterogeneous** systems.

4.3.1 Support for Timing and QoS Properties

The goal of this activity is to develop principles for specifying and analyzing timing and QoS properties of components and systems of components. The developed principles should combine the following desiderata.

- Mechanisms to define precisely a wide variety of timing properties
- Mechanisms to specify availability and requirements on resources, and the relationship between resource utilization and timing properties
- Mechanisms for clearly relating *required* properties, i.e., those resources or properties that a component assumes from its environment or platform, and *provided* properties, which are those resources or properties that a component guarantees in return.
- Scalable techniques for analyzing properties of systems, e.g., by mapping specifications to available analysis tools.

The motivation for these desiderata can be derived from the previous state-of-the-art accounts. They are also indirectly supported by observations and proposals brought forward in two research works by partners

- In the context of specification and analysis of components for automotive applications, OFFIS [Damm, Votintseva et al. 05] has elaborated on the principle of the separation of concerns, developing an approach for 3-dimensional (de)composition and analysis of component systems: horizontal, vertical, and inter-viewpoint dependencies. The approach considers explication of assumptions to be fundamental for the success of component-based development.
- MdH has (together with ABB) made a classification of different types of extra-functional properties [Crnkovic, Larsson, Preiss], based on the way in which properties of systems can be predicted from corresponding properties of its components. A conclusion of the classification is that rather few properties (e.g., static memory footprint) can be directly predicted in this way. Most properties (e.g., end-to-end latencies) are a consequence of complicated interplay between component properties (e.g., timing), other component properties (e.g., resource usage), environment properties (e.g., usage patterns) and system architecture.

Elements of partner contributions that together can realize the goals of the activity are

- *Uppsala* and *VERIMAG*: tools (*UPPAAL/TIMES* and *IF/Kronos*) for analyzing systems specified as timed automata,
- *Univ. of Cantabria:* tool set (*MAST*) that can perform classical schedulability analysis, e.g., for fixed-priority scheduling.
- *MdH* and *OFFIS:* Extensive experience from collaborating with (e.g., automotive) industry, to import requirements for a developed technology
- *EFPL* and *Timisoara* and *Uppsala* and *VERIMAG:* Expertise on timed automata specifications, compositionality, and how to separate required from provided properties.
- *ETHZ (Platforms cluster):* Expertise on techniques for specifying resources, CPU usage.
- *CEA* and *INRIA:* Expertise on model based development and UML.

Initial efforts to provide specification and analysis of timing properties in the context of model based design embed fixed-priority schedulability analysis into a framework of model-based design.

- Within the context of the SAVE Swedish national project, Uppsala and MdH are developing *SaveCCM* (the SaveComp component model) [Hansson, Åkerholm, Crnkovic, Törngren 04]. *SaveCCM* can be seen as a slight extension of the *Rubus* component model. The timing properties for a task are restricted to bounds on the execution time in each invocation. Timing properties of a system of components can be analyzed using fixed-priority analysis techniques, using e.g., the MAST schedulability modeling and analysis environment developed by the Univ. of Cantabria. An innovation in the *SaveCCM* work is that the component model has also been given a formal semantics through a translation to the timed automata formalism of UPPAAL [Carlson, Håkansson, Pettersson 05]. This opens up the possibilities for less restricted forms of timing specifications of components, although this possibility has not yet been exploited in the context of *SaveCCM.* The *SaveCCM* component model has been employed in industrial case studies, e.g., at CCM Systems.

- The Univ. of Cantabria, which has developed the MAST schedulability modeling and analysis environment, is extending it towards specifying systems of components. Within the context of ARTIST2, a modelling environment is being developed, in which scenarios and corresponding end-to-end requirements are specified, which can then be analyzed by the existing MAST tools.

These efforts demonstrate how a technology for model-based design can be developed on the basis of existing specification and analysis principles. However, this ARTIST2 activity also (and perhaps more importantly) aims at providing more general specification and analysis techniques, which are not constrained by, e.g., the assumptions made by standard fixed-priority scheduling analysis techniques for periodic tasks. A major effort in this activity is therefore directed towards relaxing these assumptions, while still making the analysis tractable and scalable.

A major goal of the January 2005 meeting was to review potential approaches towards this goal. Several promising lines of continued work were reviewed

- One promising point of departure is the real-time calculus, developed by the group of Lothar Thiele (coordinator of the ARTIST2 Platforms cluster), who participated at the January meeting. The real-time calculus can specify components under less constraining assumptions, and represent many different kinds of properties (period, jitter, bursts) in a uniform way. A further advantage activity is that it supports separation of concerns, since computation resources are treated as first-class citizens along-side with functional and timing properties; the available computation resources are specified explicitly in a uniform representation.
- A more general technology for specifying and analyzing timing properties is offered by (variants of) timed automata. At the January meeting, several aspects of this technology were reviewed.
    - The work on *timed interfaces* (by de Alfaro, Henzinger, Stoelinga, where the two latter persons were present at the January meeting) introduce a clear separation between required (assumed) and provided (guaranteed) properties.
    - Marius Minea (Timisoara) presented a survey on results for timed models that support compositionality and separation between required and provided properties. The presented frameworks extend the timed automaton-framework by concepts that assign modalities to certain elements. A typical example is Time I/O-automata, in which input and output are distinguished by semantically means. None of the presented frameworks really offered a general technique for avoiding the scalability problems in analysing systems of timed automata.
    - The *TIMES* tool, developed at Uppsala, allow to model and analyze timing and resource properties in a similar context as classical schedulability analysis. Similar capabilities are offered by the *IF/Kronos* toolset. The Uppsala team was working to find a technique for specifying component-based systems, which is more general than thos considered in classical schedulability analysis, but not too general in order to avoid the scalability problems of analysis of systems of arbitrary timed automata. A driver for this research is a case study of a robot controller, inspired by ABB.

Conclusions and induced work of the January 2005 meeting, regarding this set of problems were

- In order for general timed automata-based specifications to be a basis for component specifications, techniques must be developed for simplifying a complex timed automaton specification into one of smaller (possibly user-defined) complexity. Otherwise timed automata descriptions of systems of components will grow in complexity and become unmanageable. The January meeting also found that there are currently no obvious existing solutions to this problem. Marius Minea (Timisoara) took on the task to survey techniques for simplifying timed automata-based specifications.
- Tools for analysing systems of components specified as timed automata, such as *UPPAAL/TIMES* and *IF/Kronos* can be used for analysis also of less general forms of specifications. This had been demonstrated, e.g., in the work on *SaveCCM.* Thus, the work should be focused on the problem of finding appropriate ways to *specify* components. Analysis can then be provided through a translation from such specifications to timed automata.

  Subsequently, the Uppsala team took up this idea, and started working on translations between the real-time calculus and timed automata, with the goal to evaluate this form of specification on case studies.
- It is identified as essential to develop techniques for component specification which allow a clear separation between
  - required (assumed) and provided (guaranteed) properties,
  - different kinds of non-functional properties (e.g., between available CPU power and latency), while still being able to relate between them.

  This position was supported ed by, among others, OFFIS, reporting on extensive experience in collaboration with the automotive industry, in their elaboration on the principle of the separation of concerns, reported above.

  As a result of the January 2005 meeting, the team of Lothar Thiele took up the idea of separating between required and provided properties, and started work on incorporating it into their real-time calculus.

At the June 2005 meeting, progress on these items was reported, including:

- The team of L. Thiele (ETHZ) extended their Real-Time Calculus approach to express assumptions and guarantees, allowing constraints on timing and CPU consumption to be propagated between requirements and platform resources. This allowed them to treate scheduling problems in a compositional way, with some impressive benchmark results. Future collaboration between the “Modeling and Components” cluster with the “Execution Platform” cluster will be maintained.
- First results on translation between real-time calculus specifications and timed automata were presented by Uppsala. This translation covered mainly the timing aspects, whereas some difficulties in representing resources remained. The following discussion also revealed scepticism as to how general is the real-time calculus approach in handling phenomena like resource conflicts, non-static scheduling policies, and others. The discussion provided valuable feed-back for the continued work on this topic.
- Marius Minea (Timisoara) presented a survey on techniques for simplifying timed automata-based specifications (agreed on at the January 2005 meeting). These techniques are primarily aimed at making verification algorithms involving timed automata more efficient, and can not obviously be used to simplify specifications of systems of component in a good way. The Timisoara team is continuing to work on the problem of simplifying timed specifications, based on ideas in the surveyed work.

Following the June 2005 meeting, Timisoara, ETHZ, and Uppsala are collaborating in an effort to use these elements to develop technology for combining expressive specification with scalable analysis of component-based sytems. A first approach is to use the real-time calculus as a specification language, and timed automata (using *UPPAAL/TIMES*) as a tool for analysis. This approach is being implemented. In a longer term perspective, there may also be other suitable techniques for specifying timed systems (other than the real-time calculus), which can be handled in a similar manner. The intended result should be a toolset for analyzing timing properties of component-based systems, which alllows separation of concerns in the sense that computation resources are treated as first-class citizens along-side with functional and timing properties. The next synchronization point for this development is the workshop on performance and predictability analysis, organized by E. Deprettere and L. Thiele in Leiden, nov. 2005, where different approaches will be reviewed and tested on benchmark examples.

At the June 2005 meeting, INRIA also reported on their ongoing work to provide semantic and analysis support for their existing component model by means of translation to a timed automata model, and use of existing tools (in this case *IF/Kronos*). The goal of this work is the transformation of INRIAs existing QML based contract specification language (developed in the QCCS project) to the event and action model of the IF platform. Thus, this work follows the same general strategy as the work on *SaveCCM* described above.

An analogous effort, but for a different purpose, has been pursued by FTR&D. They have developed a translation from component-based systems to performance analysis tools (in this case, based on the theory of queueing networks), in order to be able to evaluate system performance at design time for component-based systems. In spring 2005, FTR&D has initiated a new work on WEB service composition and orchestration coping with QoS attributes. Currently, FTR&D continues to works also on contracts and Service Level Agreement for QoS attributes (performance and dependability).

In view of the many existing proposals for expressing timing properties of components and systems, to some extent arising from different needs, it was proposed (at the June meeting) to produce a catalogue of QoS properties. Such a catalogue would contain a classification of QoS properties, related to timing, with associated techniques for expressing and reasoning about them. A draft of the catalogue would be produced during Autumn 2005.

Dortmund and Uppsala are collaborating on establishing automata learning techniques for automatically deriving behavioural models of components from legacy code or observations of system behavior. Part of the work concerns extending these techniques to derive timed models. Potential applications are to derive timed models of environments of component-based system for modelling and analysis. This can potentially lead also to techniques for simplification of timed specifications. In order to complement conceptual investigation by experimentation, Dortmund has constructed LearnLib, a library for automata learning and experimentation. Its modular structure allows users to configure their tailored learning scenarios, which exploit specific properties of the envisioned applications, in order to make automata learning applicable to realistic scenarios.

CEA has studied how to adapt symbolic execution mechanisms used in the frame of the AGATHA tool to generate test cases for component systems. There were two main difficulties:

- to take into account the different data theories (one theory per component specification) in a consistent manner.
- To correctly flatten specifications written in the CEA framework (which is by nature hierarchical) in order to be able to treat them with the AGATHA tool.

Current work includes to study how to treat hidden components (that can not be directly stimulated from the environment) during testing.

An effort of general interest for component-based system development is the ongoing effort of M. Chaudron (Eindhoven) and I. Crnkovic (MdH) to write a textbook on "*Component Based Software Engineering",* which aims to cover different aspects of component models and component-based software development, and use the *SaveCCM* component model as a running illustrating example.

Some case studies are also performed in he context of this activity

- INRIA, OFFIS, and Aalborg are collaborating on an industrial case study with MAN B&W on diesel engine control.
- The *SaveCCM* component model has been employed in industrial case studies, e.g., at CCM Systems.

4.3.2 Theory for Composing Heterogeneous Systems

VERIMAG is developing a framework for heterogeneous system composition, which allows considering architectures and their elements as first class entities independent from the component's individual behaviour. Such a decoupling between behavior and architecture is essential for correctness-by-construction theory depending mainly on architectural properties. In particular, the framework introduces the notion of *glue* operators which transform sets of components into composite components. Two independent classes of glue operators are

- operators based on interaction models with provide a general mechanism for modelling the interactions between a set of components,
- priority orders which provide a general mechanism for restricting the behavior of a set of components by preserving deadlock-freedom.

Currently, these concept are being incorporated into the IF platform, thereby forming *IFCO* or *IF for components*. This tool provides means for model based simulation for any concrete component model expressed in terms of the primitives of our framework. The work in [GS04, GS05] provides necessary conditions for guaranting deadlock freedom of a component system by analysing only its interaction model. Presently, VERIMAG is working on weakening these conditions by conditions relying also on abstractions of the behaviours of components, but not of the global behaviour.

The June 2005 meeting provided an interesting opportunity to assess and discuss the relationship between the effort of *IFCO* and other existing efforts with similar aims, notably *Ptolemy* and *Metropolis.* Interesting points at which these efforts differ turned out (in the discussion) to be

- *IFCO* aims at a total separation between behavior and architecture by requiring that any element of behavior be encapsulated in a component. Communication between components can only be attained by synchronization over ports. In contrast, tools like *Metropolis* allow (perhaps more pragmatically) also communication elements (channels) to encapsulate behavior (buffers, stacks, etc.).
- IFCO also provides rather restricted means to apply control in order to satisfy QoS properties. Essentially only easily implementable primitivies (like priority) are supported. In contrast, *Metropolis* offerss a rich construct, called *Quantity managers* do specify in a rather declarative way, QoS requirements and mechanisms. This construct opens for richer system descriptions but also obscures or hides problems in realizing control mechanisms to enforce QoS properties. The philosophy of *IFCO* is that complex QoS requirements can be stated in some declarative formalism (e.g., a temporal logic), but are *not* part of the system model (they are requirements); the system model itself should use only constructs that have a clear mechanism of implementation.

A smaller effort at platforms for component modelling and execution, is the effort by Uppsala to develop a prototype virtual machine (named TICK) for timely execution of automata-based component models. Currently the kernel of the virtual machine has been implemented. On top of the kernel, component models may be developed, compiled, simulated and executed symbolically.

References

M. Chaudron, I. Crnkovic: *Component Based Software Engineering, Fundamental Concepts and Engineering Practices.* Book proposal filed with publisher. 2005

I. Crnkovic, M. Larsson, O. Preiss (2005): Concerning Predictability in Dependable Component-Based Systems: Classification of Quality Attributes, Architecting Dependable Systems III,, p pp. 257 – 278, LNCS 3549, Springer, 2005

M. Minea. *Compositionality for Timed Models – a partial survey.* Draft presentation at the cluster meeting, Jan 24, 2005.

M. Minea. *Refinement for Timing Properties.* Draft presentation at the cluster meeting, June 28, 2005.

4.4 Interaction, Building Excellence Between Partners

As a result of the series of ARTIST2 meetings held during the first year (organized by the Modeling and Components cluster, but also by the Hard Real-Time cluster), fundamental philosophies underlying the participating partners' work have been disseminated, discussed. A result is that the ARTIST2 teams have a more coherent view on basic issues. Fundamental principles that will more coherently guide future work of partners include

- the principle to separate orthogonal properties in specification of non-functional properties, e.g. to separate the specification of timing, processing resources, power, and find suitable representations for each type of property, and mechanisms to express their interdependencies,
- the importance of a clear separation between required (assumed) and provided (guaranteed) properties, while also addressing the composability and compositionality problems,
- the importance of developing a theory for design of heterogeneous systems by component composition.

The problems addressed in this activity have also poved to be central for the work of other ARTIST2 clusters, notably **Hard Real-Time** and **Execution Platforms.** Collaborations with these clusters have been initiated as a result of ARTIST2 meetings.

Joint publications by activity partners

T. Berg, O. Grinchtein, B. Jonsson, M. Leucker, H. Raffelt, B. Steffen: On the Correspondence Between Conformance Testing and Regular Inference. In Prof. FASE 2005, Fundamental Approaches to Software Engineering, 8th International Conference, Edinburgh, UK, April 4-8, 2005, Lecture Notes in Computer Science 3442, pp 175-189,

T. Berg, H. Raffelt, B. Steffen: LearnLib: A Library for Automata Learning and Experimentation FMICS 2005, 10th Intl. Workshop on Formal Methods in Industrial Critical Systems.

J. Carlsson, J. Håkansson, P. Pettersson. SaveCCM: An analyzable component model for real-time systems. Proc. Int. Workshop on Formal Aspects of Component Software, Oct. 2005.

Ivica Crnkovic, Jakob Axelsson, Susanne Graf , Magnus Larsson, Rob van Ommering , Kurt Wallnau, COTS Component-Based Embedded Systems - A Dream or Reality4th International Conference, ICCBSS 2005, Springer, Bilbao, Spain, February, 2005

J. Elmquist, S. Nadjm-Tehrani, M. Minea. Safety Interfaces for Component-Based Systems. Proceedings of the 24th International Conference on Computer Safety, Reliability and Security (SAFECOMP), 2005

Huáscar Espinoza, Hubert Dubois, Sébastien Gérard, and Julio Medina. A General Structure for the Analysis Framework of the UML MARTE Profile",To appear in the Proccedings of the International Workshop "MARTES: Modeling and Analysis of Real-Time and Embedded Systems" hold on October 4,2005 in Montego Bay, Jamaica in conjunction with 8th International Conference on Model Driven Engineering Languages and Systems, MoDELS/UML 2005.

4.5 Spreading Excellence

ARTIST2 teams of this cluster are involved in other national and international projects, where the issues addressed in this activity are central.

The modelling and components cluster is, together with two other clusters, organizing the ARTIST2 Summer school in 2005.

ARTIST2 members have organized several tutorials and workshops at conferences, including

- Tutorial “component-based approach embedded systems” presented at Automated Software Engineering (ASE) conference, Linz, Sept. 2005,
- Organisation of a panel “component-based development for embedded systems, held at Intl. conference for commercial components (ICCBSS, Bilbao, February 2005)