INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Project-Team aoste

Models and Methods for the Analysis and Optimization of Systems with Real-time and Embedded Constraints

Sophia Antipolis - Rocquencourt

THEME COM

Activity Report

2005

Table of contents

1. Team

Aoste is a joint project with UNSA (university of Nice/Sophia-Antipolis) and CNRS, through their I3S laboratory. Aoste is also spread between the two INRIA sites of Sophia-Antipolis and Rocquencourt. Aoste is a follow-up of the former INRIA Tick and Ostre teams, and of the I3S Sports team.

Head of project team
Robert de Simone [Senior Researcher, INRIA]

Vice-head of project team
Yves Sorel [Senior Researcher, INRIA]

Administrative assistant
Sophie Honnorat [Sophia-Antipolis, part-time]
Nelly Maloisel [Rocquencourt, part-time]

Staff member INRIA
Dumitru Potop [Researcher, from october]
Alix Munier [On temporary leave from University Paris XII]

Staff member UNSA
Charles André [professor]
Frédéric Mallet [associate professor]
Marie-Agnès Peraldi-Frati [associate professor]

PhD. students
Julien Boucaron [ST Microlectronics contractual funding]
François Lagarde [Funded by CEA-List, from November]
Patrick Meumeu Yomsi [INRIA scolarship, from December]
Jean-Vivien Millo [BDE regional scholarship with ST Microelectronics, from december]
Nicolas Pernet [INRIA scolarship]
Mickaël Raulet [INSA/MITSUBISHI ELECTRIC scholarship]

Post-doctoral fellow
Arnaud Cuccuru [from October]
Benoît Miramond [from January to August]

Specialist Engineer
Gérard Cristau [part-time, from October]

Technical Staff
Cyril Faure [Project engineer]
Arnaud Rouanet [Software development staff, until September]
Christophe Gensoul [Software development staff, from September]

Student interns
Omar Kermia [University Versailles/Saint-Quentin]
Patrick Meumeu Yomsi [Engineer School ECE Paris]
Jean-Vivien Millo [University Marne-la-Vallée]
Cédric Piasco [University Nice/Sophia-Antipolis]
Quentin Quadrat [Engineer School EPITA Paris]
Céline Roussel [Engineer School ESIEE Noisy-Le-Grand]

External collaborators
Liliana Cucu [Post-doctoral fellow Polytechnic Institute or Porto, Portugal]
Thierry Grandpierre [Assistant Professor ESIEE Noisy-Le-Grand]

2. Overall Objectives

2.1. Overall Objectives

Keywords: *UML*, *adequation*, *architectural models*, *automatic verification*, *code distribution*, *code generation*, *codesign*, *compilation techniques*, *formal semantics*, *hardware synthesis*, *high-level modeling*, *mapping*, *multiprocessors*, *optimization*, *program analysis*, *real-time embedded systems*, *scheduling*, *synchronous hypothesis*, *synchronous reactive formalisms*, *systems-on-Chip*.

The main goal of Aoste is to provide innovative approaches for the design of real-time and embedded systems, based on powerful algorithmic methods applied to well-defined models with sound mathematical semantics (in short: formal model-based design). Here "design" means altogether:

High-Level Modeling
Model transformation and analysis
Implementation onto Embedded platforms

We shall try to promote our semantically sound, model-based approach at each of these three levels. In the case of *high-level modeling*, this translates into the need for offering appealing and familiar syntactic constructs to help shape up modular semantic model construction; for the *transformation and analysis part*, the goal is to provide "correct-by-construction" synthesis techniques to transform models progressively from higher-level to lower-level, and check otherwise their match against correctness properties; in the *implementation onto embedded platforms* part, we introduce an explicit architectural platform, and consider the optimized mapping in time and space of models-as-programs onto this platform.

To cover this vast spectrum of topics we need to specialize the type of formalisms we shall consider. We focus on *synchronous reactive languages*, such as Esterel/SyncCharts, and on the *Application/Architecture/Adequation* (AAA) methodology, as implemented in the SynDEx design environment. Typical application domains for our techniques are found in nowadays major embedded electronic fields, such as mobile telephones and similar appliances branded as "secure communicating objects", in mobile robotics, in automotive and aircraft transportation, and in digital system-on-chip (SoC) design.

3. Scientific Foundations

3.1. High-Level Modeling

Keywords: *Esterel*, *SyncCharts*, *UML*, *synchronous formalisms*.

Participants: Charles André, Julien Boucaron, Robert de Simone, Frédéric Mallet, Jean-Vivien Millo, Marie-Agnès Péraldi, Dumitru Potop, Yves Sorel.

3.1.1. Synchronous formalisms

Historically the so-called *synchronous reactive formalisms* [29], [11] were developed, mostly inside French research groups in the 1980's (Esterel, Lustre, Signal was the trilogy), as foundational study of semantically well-founded description languages for real-time embedded software systems. Meanwhile a number of modeling languages were also introduced in this field, aiming more specifically at system simulation with discrete time steps: HDLs for hardware circuits, Statecharts for embedded software modeling, Simulink for signal and image processing and control theory. It should be recognized in our view that synchronous languages brought exactly what these simulation formalisms lack: a clear sense of correct construction properties, under which the instantaneous behavior (the reaction) can always be provably scheduled safely in an intelligible way, and which makes executions deterministic and complete (because all valid scheduling are essentially causally equivalent). With such an assumption there is a guaranteed match between the simulation model and the executable code obtained through the implementation ("*what you simulate is what you execute*"), and this opens the way to many important design activities, such as synthesis, verification, and test generation activities, which are largely banned out of a setting where the simulation model may differ from the actual implementation. Also, the precise scheduling in the case of synchronous formalisms is not required from the designer, it is synthesized from the high-level correctness principles. Examples of such benefits are the *clock calculus* in Signal, and in our case the *constructive* semantics of Esterel, and the optimized mapping of application specified with these languages, onto architectural models of SynDEx.

Esterel [13], [16] was developed jointly at INRIA and École des Mines de Paris, in the Meije research team then headed by Gérard Berry. The language is of imperative nature, with syntactic features for precise description of reactive instants, conceptual parallelism (potential parallelism), signal broadcast and preemption. Its scope is the representation of control-dominated reactive systems as hierarchical automata. Under a strict correctness condition of *constructive causality* [14] (signal presence values should be determined at any instant before it is tested), it can be given formal interpretation, either in the form of synchronous circuits, or as Mealy finite state machines. These, being classical mathematical models, allow design transformation activities such as optimization and automatic verification based on model-checking, and can produce target C code directly simulating the behavior (of the circuit or the Mealy machine respectively).

SyncCharts were developed inside the SPORTS project-team at I3S (UNSA-CNRS) [2], [8]. As opposed to Statecharts (and UML State Machines), their graphical syntax respect a strict state containment hierarchy, and transitions cannot cross a macrostate boundary. But where SyncCharts greatly depart from UML State Machines is on semantics, which can only be asynchronous in UML due to the lack of time modeling, while it is carefully synchronous in our case. The execution semantics of UML State Machines is described in terms of operations of a hypothetical machine that implements a state machine specification. Events are dispatched and processed by the state machine, *one at a time*. This does not open the possibility to handle simultaneous occurrences of events, which are the rule in synchronous models.

The synchronous semantics of SyncCharts is based on constructive causality issue: in ecah instant a value should be produced prior to being consumed. Constructiveness is in fact a notion first studied by Sharad Malik (Princeton U.). Variants of Esterel borrowing syntax from general-purpose languages such as ECL (Esterel-C language) and Jester (Java-Esterel) were designed inside Cadence Berkely Labs in the context of the POLIS codesign/cosimulation project that ultimately led to the VCC product by Cadence. In Germany the SYNERGY project was conducted at GMD in Axel Poigné's group to build an environment merging features of Esterel, Lustre and Argos (a synchronous variant of StateCharts less expressive as SyncCharts, developed at VERIMAG). Work on foundational semantics of synchronous formalisms were also conducted in Germany (Quartz project of Klaus Schneider at Karlsruhe U.) and in the UK (Michael Mendler and Gerard Lüttgen, Sheffield U.). Work on optimized compilation schemes for Esterel were developed in parallel at France Telecom Grenoble (Etienne Closse, Daniel Weil *et al.*), at Synopsys and then Columbia U. (Stephen Edwards), and in the INRIA Tick project (PhD thesis of Dumitru Potop).

Esterel/SyncCharts and Lustre/SCADE are now developed and commercialized in an industrial context by Esterel Technologies, an INRIA spin-off founded partially by former members of the Tick research group (and scientifically headed by Gérard Berry). On the academic side we study new directions for compilation/synthesis, optimization and analysis/verification, based on languages extensions or erstrictions, and heterogeneous embedded compilation targets.

3.1.2. Globally-Asynchronous/Globally-Synchronous (GALS) extensions and latency-Insensitive design

Sometimes, due to system complexity and physical latencies due to long interconnections, the *synchronous hypothesis* becomes unrealistic for implementation, while remaining a major useful reference specification. The prospect of the latency-insensitive design approach, as pionnered by Luca Carloni *et al.*, is to start from a synchronous specification, then desynchronize it. This step involves propagating "absent" signal values and allowing unbounded buffering for "in-travel" communications, and produces a model close to that of Event/Marked graphs in classical Petri Net theory. Third, mandatory latencies are introduced to represent physical constraints (gathered elsewhere); they provide effective bounds for buffering resources, and back-pressure flow-control protocols are to be applied. In essence this amounts to reconstructing a new fully synchronous system, but this time with a new, smaller instant granularity, one that complies with the latency requirements imposed. Each of these steps involve the addition of extra protocol “wrappers” around the previously existing components, to allow semantic preservation of behavior in a precisely defined way. These components can themselves be expressed as synchronous processes.

Providing efficient bounds on the size of buffering storage elements required to accomodate the iimposed latencies is an important issue. It amounts to scheduling techniques as found in [17], [34], [22], but their application in the domain of SoC design raises new opportunities in optimization, such as those based on low-power dissipation criteria.

In Latency-Insensitive modeling the handling of absence is explicit (new signals have to be cast to notify it). This is generaly costly at run time, and work on extending the approach to *multi-clock* systems, where absence is truly dealt with as a primitive notion, are currently under investigation.

3.1.3. UML modeling diagrams for Real-Time Embedded applications

The UML consists of a variety of models (or “diagrams”), aiming at covering modeling concerns during the whole lifespan of software engineering. Of particular interest to us are some models of *structural* or *behavioral* nature, and in the later case *state* and *sequence* diagrams. State diagrams may represent components behaviors, in a way inspired from StateCharts. Sequence diagrams represent possible interaction scenarios between components, in a way inspired from Message Sequence Charts.

The only “semi-formal” semantics of models is usually given in natural language, with high risks of ambiguity and, even worse, inconsistencies; there is a uniform lack of clear relationship *between* the various models; currently, sequence diagrams have poor expressiveness. A number of research efforts address this demand for rigourous semantics of the UML models, such as the *Precise UML* group (http://www.cs.york.ac.uk/puml) or the Neptune (Nice Environment with a Process and Tools Using Norms and Example) project (http://neptune.irit.fr). *Umlaut* (J-M. Jezequel, INRIA Triskell team) is a UML transformation framework allowing complex manipulations to be applied to a UML model, where manipulations are expressed as algebraic compositions of reified elementary transformations.

The weak expressivity of sequence diagrams as models of interactions is currently tackled by researchers proposing extensions of *Message Sequence Charts* (MSC) to this end (thus outside the UML standardization community). Work on “High-Level MSCs” (for instance in INRIA Triskell and S4 teams) or on “Live Sequence Charts” (LSC, by D. Harel and W. Damm) fall into this category. With *LSCs* one can express possible, but also mandatory or even forbidden interaction scenarios. Prototypical tools at the University of Oldenburg/OFFIS provide semantic interpretation into timed automaton.

While the standard UML is aimed at general-purpose object-oriented software engineering, specific extensions (or “profiles”) have been proposed to deal specifically with real-time aspects. Just to mention a few such: UML-RT (B. Selic) used in the Rational Rose-RT development environment, RT UML (B. Douglass) used in Rhapsody (I-Logix), and ACCORD/UML (F. Terrier, CEA). UML-RT is based on the first success story of ROOM (Real-time Object-Oriented Modeling), introduced in 1994 by Selic, Gullekson, and Ward, and put on the market by ObjecTime. The RTAD (Real-Time Analysis and Design) working group of the OMG has been especially created to promote real-time issues within the OMG, and to specialize the UML to be suitable for different real-time domains. The newly adopted UML-SPT profile is a first visible result, enabling models that support Scheduling, Performance, and Time evaluation. However, it fulfils different needs than ours (quantitative performance analysis rather than executable specifications). We are taking part in the elaboration of a Request-For-Proposal (RFP) for a new profile in this field, named MARTE (Modeling and Analysis of Real-Time Embedded systems). In this context we shall try if possible to promote models relevant to our goals.

There is a growing interest in integrating synchronous concepts into UML (or UML profiles). The former PAMPA and Ep-Atr teams (IRISA) once proposed the BDL formalism, to study a mixed synchronous-asynchronous semantics. The UML here plays the role of a federator notation. The I3S SPORTS project has adopted another point of view [4]: the direct use of synchronous (imperative) models. The question is whether these enrichments are “lightweight” (stereotypes or tag values) or “heavyweight” changes to the UML. Heavyweight here means that synchronous hypotheses are at places incompatible with some basic current assumptions made in UML. A UML state machine, which is a variant of Statecharts, has a queue for incoming events, an event dispatcher that selects and de-queues event instances one at a time, and an event processor which processes dispatched event instances under a run-to-completion scheduling policy. This definitely excludes simultaneous occurrences. Interaction models raise similar difficulties. Introducing our synchronous models to UML (SyncCharts as state-based model, and SIB as interaction model) would need changes at the meta model level.

The new **UML2.0** standard should improve the ability and utility of the UML with respect to architecture and scalability (through its “Superstructure” RFP). In the new version *classes* can be structured and reuse other classes playing specific “*parts*” roles. *Ports* are introduced for architectural modeling, as (instantiable) connection points through which *part* instance export specific services or operations accross the class boundary. *Interfaces* should be also expanded to allow specification of a *required interface* (from the distant other end) in addition to the usual notion of (local) *offered interface*. Moreover, the specification of allowable sets of sequences of service invocations might be specified with “protocol state machine”. Almost all these new possibilities were present in the ROOM’s *capsule* notion, a major influence. Such a model can play the role of an ADL (Architecture Description Language). Other improvements are related to behavioral models: *sequence diagrams* might now be broken up into “interaction fragments”, with nesting capabilities and extended control constructs, making them closer to MSC and LSC. Last but not least, a form of *Data Flow Diagrams* should be introduced (since *activity diagrams* address only partially this issue).

To summarize, new UML trends meet our concerns about system architecture, components, and behavior. UML offers rich and standard notations, but lacks semantic rigor at places. This should not hinder our objectives of rigorous system design. Whenever an official model semantics will appear as not defined well enough, we shall feel free to adapt it: *strict UML compliance is not our goal !*

3.2. Transformations and analysis

Keywords: *Compilation, formal verification, synthesis.*

Participants: Robert de Simone, Dumitru Potop.

The fact that syntactic constructs in synchronous languages are fully defined in terms of corresponding semantic operations immediately pays off in that all kinds of compilation/synthesis, analysis and verification, or optimization methods can readily be caracterized as formal transformations of mathematical models [49]. Therefore we realized, far in advance, the announced programme of Model-driven Architecture, in our case with true meaningful algorithmic transformations. To this end we extensively use a number of well-defined mathematical models, such as (hierarchical) finite-state Mealy machines, synchronous circuits as Boolean gate netlists, or data-control flowgraphs. The various steps of compilation (or static analysis, or dynamic analysis seen as model-checking, or optimization at each model level) are always formally defined, and thus *correct-by-construction*, discarding altogether issues of “synthetizability subsets”, or discrepancy between simulation, that limit the application of syntactic formalisms without clear unambiguous semantics in the field.

3.2.1. Compilation/synthesis

Synchronous languages raise specific issues regarding efficient compilation of programs onto various targets (when producing hardware descriptions, one usually talks of “synthesis” rather than compilation). This is due to the strong demand on semantic preservation across models. The case of compilation onto distributed embedded architectures, where compilation requires an architecture model description and fancy mapping/scheduling techniques shall be further described as part of the AAA methodology.

Concerning Esterel/SyncCharts, compilation was first realized in the 1980’s as an expansing into flat global Mealy FSMs; this produces efficient, but often unduly large code size. Then in the 1990’s a translation was defined into Boolean equation systems (BES), with Boolean register memories encoding active control points. While such models are known in the hardware design community as Boolean gate *netlists*, they can be used in our context for software code production. Here the code produced in quasi-linear in size (worst-case quadratic in rare cases), but the execution consists in a linear evaluation of the whole equation system (thus each reaction requires an exceution time proportional to the whole program, even when only a small fragment is truly active). Thus in the early 2000's new implementation frameworks were thought, relying on high-level control-data flowgraphs selecting the active parts before execution at each instant. This scheme is both fast and memory-efficient, but cannot cope with all programs (as the full constructive causality analysis underlying the synchronous assumption cannot then be realized at "compile time", and this check is of utterly importance for program correctness). Correctnes is in this context ensured by a stronger, more restrictive *acyclicity* criterion, which provides a static evaluation order for signal propagation.

Some of the specific issues of Esterel/SyncCharts compilation are:
- the potential existence of *instantaneous loops*, diverging so that the instant never reaches completion ("non-zeno" behaviors);
- *shizophrenic* problems, where signals and other variables may assume several distinct values, and be instanciated multiply in the same reaction. This calls for program fragment *reincarnation* for efficient implementation (with single-static assignment techniques);
- *constructive causality* issues, when a static order to signal propagation cannot be defined, as this order may vary through time. This may call for the symbolic computation of the reachable state space (RSS) covering all potential control configurations, to establish computability of behaviors.

In fact these issues are sometimes *less* specific than can be thought on first glance, and the interplay between our specialized techniques, and more general compilation methods, is a subject of current studies. This si useful to help synchronous languages get rid of this false "niche topic" image. In particular, *static analysis* techniques are getting momentum in the justification of several compilation steps for Esterel, allowing to smoothly derive compiler specification from (extended) formal semantics, thus paving the way to true mathematical compiler certification.

One advantage of compilation schemes transforming programs from formal models to formal models is that one can benefit (and sometimes contribute) to a rich body of optimization and analysis techniques developed for these models. Circuit and FSM optimization/minimization, equivalence-checking and temporal logic model-checking are instances of this.

3.2.2. Dynamic analysis, automatic verification and model-checking

Formal finite state semantics paves the way to model-checking, which has reached its largest success in synchronous hardware verification. Here the most famous tool is SMV, a symbolic BDD-based model-checker developed at CMU (E. Clarke, K. McMillan), although a number of other similar tools exists, both in public domain or industrial proprietary versions. Recently a standardization of temporal property syntax named SUGAR was achieved at international level. In the past dedicated model-checkers were developed for all synchronous languages: XEVE for Esterel, LESAR/NBAC for Lustre, SIGALI for Signal. In all cases they are symbolic BDD-based model-checkers, avoiding temporal logic formalisms by stating generic properties, and adding synchronous parallel processes to test and monitor observed systems so that large classes of properties can be reduced to generic ones on combined systems. In the last decade model-checking techniques, mostly based on symbolic state space constructions, were extended to cover other problems than mere satisfaction of properties. In Esterel they were used to enforce "state-conscious" constructive causality, and in designing more efficient (but costly) optimization techniques on circuit descriptions. In Lustre, and then Esterel, they were used for automatic test pattern generation with the aim of covering specified portions of the state space (actually those techniques using explicit state spaces originated in the context of asynchronous languages, in the work of INRIA Pampa team for instance). In Signal they were extended to the problem of *controller synthesis*, in which a synchronous parallel supervisor is algorithmically built to keep the observed system from reaching pathological or forbidden states.

Recently, new techniques for "bounded" model-checking using efficient SAT-solvers were introduced (SAT being the well-known SATisfiability problem for propositional logic). They sacrify completeness (one must know "up to what depth" the property is meant to hold, or provide non-automatically some sort of recursion invariant assumption to be proved) to the sake of efficiency; in fact the economic pressure of hardware verification led to new advances in the algorithmic heuristics involved in SAT-solving methods, with the so-called Stalmark method in Sweden (by Prover Technologies), and iterative learning methods as implemented in Schaff (Malik, Princeton U.), BerkMin and predecessors.

3.3. Mapping onto Embedded platforms

Keywords: *RTL*, *Rapid prototyping*, *distributed*, *embedded*, *executive*, *fault tolerance*, *graph*, *hardware/software co-design*, *multiprocessor*, *off-line*, *on-line*, *optimization*, *parallel*, *partial order*, *partitioning*, *real-time*, *real-time operating system*, *real-time scheduling*, *specific integrated circuit*, *synchronous languages*, *system level CAD*.

Participants: Liliana Cucu, Dumitru Potop, Yves Sorel.

3.3.1. The AAA methodology

The AAA methodology (Algorithm-Architecture Adequation) allows to specify “application algorithms” (functionalities) and redundant “multicomponent architectures” (composed of processors and specific integrated circuits all together interconnected) with graph models. Consequently, all the possible implementations of a given algorithm onto a given architecture is described in terms of graphs transformations. An implementation consists in distributing and scheduling a given algorithm onto a given architecture. Adequation amounts to chose one implementation among all the possible ones, such that the real-time and embedding constraints are satisfied and the hardware redundancy is fully used. Furthermore, from the adequation results our graph models allow to generate automatically, as an ultimate graphs transformation, two types of codes: dedicated distributed real-time executives or configuration of standard distributed real-time executives (RTlinux, OSEK, etc) for processors, and net-lists (structural VHDL) for specific integrated circuits. Finally fault tolerance is of great concern because the applications we are dealing with are often critical, that is to say, may lead to catastrophic consequences when they fail. The AAA methodology provides a mathematical framework for rapid prototyping and hardware/software co-design taking into account fault tolerance.

From the optimization point of view, real-time systems are, first of all, “reactive systems” which mandatorily must react to each input event of the infinite sequence of events it consumes, such that “cadence” and “latency” constraints are satisfied. The latency corresponds to the delay between an input event consumed by the system and an output event produced by the system in reaction to this input event. The cadence corresponds to the delay between two successive input events, i.e. a period. The term event is used in a broad sense, it may refers to a periodic or to an aperiodic discrete (sampled) signal. When hard (critical) real-time is considered, off-line approaches are preferred due to their predictability and best performances, and when on-line approaches are unavoidable, mainly to take into account aperiodic events, we intend to minimize the decisions taken during the real-time execution. When soft real-time is considered off-line and on-line approaches are mixed. The application domains we are involved in, e.g. automobile, avionic, lead to consider scheduling problems for systems of tasks with precedence, latency and periodicity constraints. We seek optimal results in the monoprocessor case where distribution is not considered, and sub-optimal results through heuristics in the multiprocessor case, because the problems are NP-hard due to distribution consideration. Also, in addition to these timing constraints, embedded systems must satisfy technological constraints, such as power consumption, weight, volume, memory, etc, leading in general to minimize hardware resources. In the most general case architectures are distributed, and composed of several programmable components (processors) and several specific integrated circuits (ASIC[1] or FPGA[2]) all together interconnected with possibly different types of communication media. We call such heterogeneous architectures “multicomponent” [39].

The complexity, not only of the algorithms that must be implemented, but also of the hardware architectures, and also the multiple constraints, imply to use methodologies when development cycle time must be minimized from the high level specification until the successive prototypes which ultimately will become a commercial product. In order to avoid gaps between the different steps of the development cycle our AAA methodology is

[1]ASIC : Application Specific Integrated Circuit
[2]FPGA : Field Programmable Gate Array

based on a global mathematical framework which allows to specify the application algorithms as well as the hardware architecture with graph models, and the implementation of algorithms onto architectures in terms of graphs transformations. This approach has the benefit on the one hand to insure traceability and consistency between the different steps of the development cycle, and on the other hand to perform formal verifications and optimizations which decrease real-time tests, and also to perform automatic code generation (real-time executives for processors and net-list for specific integrated circuits). All these benefits contribute to minimize the development cycle. Actually, the AAA methodology provides a framework for hardware/software co-design where safe design is achieved by construction, and automatic fault tolerance is possible only by specifying the components that the user accepts to fail.

To summarize, we are interested in the optimization of distributed real-time embedded systems according to four research topics:

1. models for specifying, with graphs and partial orders, application algorithm, hardware architecture, and optimized implementation,
2. implementation optimization:
   - real-time scheduling algorithms in the case of monoprocessor,
   - real-time distribution and scheduling heuristics in the case of multiprocessor,
   - heuristics for resources minimization in the case of multiprocessor and specific integrated circuit,
3. automatic code generation for processor (dedicated or standard RTOS configuration) and for specific integrated circuit (net-list),
4. fault tolerance.

Beside these researches, we propose a tool implementing the AAA methodology. It is a system level CAD software called SynDEx (http://www.syndex.org). This software, coupled with a high level specification language, like one of the Synchronous Languages or Scicos, leads to a seamless environment allowing to perform rapid prototyping and hardware/software co-design while reducing drastically the development cycle duration and providing safe design.

We next describe in greater details the three main modeling aspects involved in the approach.

3.3.1.1. Algorithm model

Our algorithm model is an extension of the well known data-flow model from Dennis [64]. It is a directed acyclic hyper-graph (DAG) [63] that we call “conditioned factorized data dependence graph” [33], whose vertices are “operations” and hyper-edges are directed “data or control dependences” between operations. Hyper-edges are necessary in order to model data diffusion since a standard edge only relates a pair of operations. The data dependences defines a partial order on the operations execution [72], called “potential operation-parallelism”. Each operation may be in turn described as a graph allowing a hierarchical specification of an algorithm. Therefore, a graph of operations is also an operation. Operations which are the leaves of the hierarchy are said “atomic” in the sense that it is not possible to distribute each of them on more than one computation resource. The basic data-flow model was extended in three directions, firstly infinite (resp. finite) repetitions in order to take into account the reactive aspect of real-time systems (resp. “potential data-parallelism” similar to loop or iteration in imperative languages), secondly “state” when data dependence are necessary between repetitions introducing cycles which must be avoided by specific vertices called “delays” (similar to z^{-n} in automatic control), thirdly “conditioning” of an operation by a control dependence similar to conditional control structure in imperative languages. Delays combined with conditionings allow to specify FSM (Finite State Machine) necessary for describing “mode changes”, e.g. some control law is performed when the motor is the state “idle” whereas another one is performed when it is in the state “permanent”. Repetition and conditioning are both based on hierarchy. Indeed, a repeated or “factorized graph of operations” is a hierarchical vertex specified with a “repetition factor” (factorization allows to display only one repetition). Similarly, a “conditioned graph of operations” is a hierarchical vertex containing several alternative operations, such that for each infinite repetition, only one of them is executed, depending on the value carried by the “conditioning input” of this hierarchical vertex. Moreover, the proposed model has the synchronous language semantics [66], i.e. physical time is not taken into account. This means that it is assumed an operation produces its output events and consumes its inputs events simultaneously, and all the input events are simultaneously present. Thus, by transitivity of the execution partial order associated to the algorithm graph, outputs of the algorithm are obtained simultaneously with its inputs. Each input or output carries an infinite sequence of events taking values, which is called a “signal”. Here, the notion of event is general, i.e. signals may be periodic as well as aperiodic. The union of all the signals defines a “logical time”, where physical time elapsing between events are not considered.

3.3.1.2. Architecture model

The typical coarse-grain architecture models such as the PRAM (Parallel Random Access Machines) and the DRAM (Distributed Random Access Machines) [73] are not enough detailed for the optimizations we intend to perform. On the other hand the very fine grain RTL-like (Register Transfer Level) [71] models are too detailed. Thus, our model of multicomponent architecture is also a directed graph [25], whose vertices are of four types: “operator” (computation resource or sequencer of operations), “communicator” (communication resource or sequencer of communications, e.g. DMA), memory resource of type RAM (random access) or SAM (sequential access), “bus/mux/demux/(arbiter)” (choice resource or selection of data from or to a memory) possibly with arbiter (arbitration of memory accesses when the memory is shared by several operators), and whose edges are directed connections. For example, a typical processor is a graph composed of an operator, interconnected with memories (program and data) and communicators, through bus/mux/demux/(arbiter). A “communication medium” is a linear graph composed of memories, communicators, bus/mux/demux/arbiters corresponding to a “route”, i.e. a path in the architecture graph. Like for the algorithm model, the architecture model is hierarchical but specific rules must be satisfied, e.g. a hierarchical memory vertex may be specified with bus/mux/demux and memories (e.g. several banks), but not with operator. Although this model seems very basic, it is the result of several studies in order to find the appropriate granularity allowing, on the one hand to provide accurate optimization results, and on the other hand to quickly obtain these results during the rapid prototyping phase. Data communications can be precisely modeled through shared memory or through message passing possibly using routes. Furthermore, complex interactions between operators and communicators can be taken into account through bus/mux/demux/arbiter, e.g. when communications with DMA require the sequencer of a processor.

Our model of integrated circuit architecture is the typical RTL model. It is a directed graph whose vertices are of two types: combinatorial circuit executing an instruction, and register storing data used by instructions, and whose edges are data transfers between a combinatorial circuit and a register, and reciprocally.

In order to unify both multicomponent and integrated circuit models we extend the RTL model in a new one called “macro-RTL”. Thus, an operator executes “macro-instructions”, i.e. operations, which consume and produce data in “macro-registers”. This model allows to encapsulate specific details related to the instructions set such as cache, pipe-line and other non deterministic features of processors that are difficult to take into account.

3.3.1.3. Implementation mapping

An implementation of a given algorithm onto a given multicomponent architecture corresponds to a distribution and a scheduling of, not only the algorithm operations onto the architecture operators, but also a distribution and a scheduling of the data transfers between operations [27] onto communication media.

The distribution consists in distributing each operation of the algorithm graph onto an operator of the architecture graph. This leads to a partition of the operations set, in as many sub-graphs as there are operators. Then, for each operation two vertices called “alloc” for allocating program (resp. data) memory are added, and each of them is allocated to a program (resp. data) RAM connected to the corresponding operator. Moreover, each “inter-operator” data transfer between two operations distributed onto two different operators, is distributed onto a route connecting these two operators. In order to actually perform this data transfer distribution, according to the element composing the route as many "communication operations" as there are communicators, as many "identity" vertices as there are bus/mux/demux, and as many "alloc" vertices for allocating data to communicate as there are RAM and SAM, are created and inserted. Finally, communication operations, identity and alloc vertices are distributed onto the corresponding vertices of the architecture graph. All the alloc vertices, those for allocating data and program memories as well as those for allocating data to communicate, allow to determine the amount of memory necessary for each processor of the architecture.

The scheduling consists in transforming the partial order of the corresponding subgraph of operations distributed onto an operator, in a total order. This "linearization of the partial order" is necessary because an operator is a sequential machine which executes sequentially the operations. Similarly, it also consists in transforming the partial order of the corresponding subgraph of communications operations distributed onto a communicator, in a total order. Actually, both schedulings amount to add edges, called "precedence dependences" rather than data dependences, to the initial algorithm graph. To summarize, an implementation corresponds to the transformation of the algorithm graph (addition of new vertices and edges to the initial ones) according to the architecture graph.

Finally, the set of all the possible implementations of a given algorithm onto a given architecture may be modeled, in intention, as the composition of three binary relations: namely the "routing", the "distribution", and the "scheduling" [41]. Each relation is a mapping between two pairs of graphs (algorithm graph, architecture graph). It also may be seen as a external compositional law, where an architecture graph operates on an algorithm graph in order to give, as a result, a new algorithm graph, which is the initial algorithm graph distributed and scheduled according to the architecture graph. Therefore, the "implementation graph" is of type algorithm that may in turn be composed with another architecture graph, allowing complex combinations.

The set of all the possible implementations of a given algorithm onto a specific integrated circuit is different because we need a transformation of the algorithm graph into an architecture graph which is directly the implementation graph. This graph is composed of two parts: the data-path obtained by translating each operation in a corresponding logic function, and the control path obtained by translating each control structure in a "control unit", which is a finite state machine made of counters, multiplexers, demultiplexers and memories, managing repetitions and conditionings [21].

3.3.2. Optimization

We must choose among all the possible implementations a particular one for which the constraints are satisfied and possibly some criteria are optimized.

In the case of a multiprocessor architecture the problem of finding the best distribution and scheduling of the algorithm onto the architecture, is reputed to be of NP-hard complexity [68]. This amounts to consider, in addition to precedences constraints specified through the algorithm graph model, one latency constraint between the first operation(s) (without predecessor) and the last operation(s) (without successor), equal to a unique periodicity constraint (cadence) for all the operations. We propose several heuristics based on the characterization of the operations (resp. communication operations) relatively to the operators (resp. communicators). For example the total execution time of the algorithm (makespan) onto the distributed architecture, may be minimized with a cost function taking into account the schedule flexibility of operations, and also the increase of the critical path when two operations are distributed onto two different operators inducing a communication, possibly through concurent routes [27]. The characterization amounts to relate the logical time described by the interleaving of events with the physical time. We mainly develop "greedy heuristics" because they are very fast [69], and thus, well suited to rapid prototyping of realistic industrial applications. In this type of applications the algorithm graph may have up to ten thousand vertices and the architecture graph may have several tens of vertices. However, we extend these greedy heuristics to iterative versions [40] which are much slower, due to back-tracking, but give better results when it is time to produce the final commercial product. For the same reason we also develop local neighborhood heuristics such as simulated anealing, and also genetic algorithm, all based on the same type of cost function.

New applications in the automobile, avionic, or telecommunication domains, lead us to consider more complex constraints. In such applications it is not sufficient to consider the execution duration of the algorithm graph. We need also to consider periodicity constraints for the operations, possibly different, and several latency constraints imposed possibly on whatever pair of operations. Presently there are only partial and simple results for such situations in the multiprocessor case, and only few results in the monoprocessor case. Then, we began few years ago to investigate this research area, by interpreting, in our algorithm graph model, the typical scheduling model given by Liu and Leyland [70] for the monoprocessor case. This leads us to redefine the notion of periodicity through infinite and finite repetitions of an operations graph (i.e. the algorithm), thus generalizing the SDF (Synchronous Data-Flow) model [67] proposed in the software environment Ptolemy. For simplicity reason and because this is consistent with the application domains we are interested in, we presently only consider that our real-time systems are non-preemptive, and that “strict periodicity” constraints are imposed on operations, meaning that an operation starts as soon as it period occurs. In this case we give a schedulability condition for graph of operations with precedence and periodicity constraints in the non-preemptive case [20].

We also formally defined the notion of latency which is more powerful [19], for the applications we are interested in, than the usual notion of “deadline” that does not allow to impose directly a timing constraint on a pair of operations, connected by at least one path, like it is necessary for “end-to-end constraints”. In order to study schedulability conditions for multiple latency constraints we defined three relations between pair of paths, such that for each pair a latency constraint is imposed on its extremities. Using these relations called *II*, *Z* and *X*, we give a schedulability condition for graph of operations with precedence and latency constraints in the non-preemptive case. Then by combining both previous results we give a schedulability condition for graph of operations with precedence, periodicity and latency constraints in the non-preemptive case, using an important result which gives a relation between periodicity and latency. We also give an optimal scheduling algorithm in the sense that, if there is a schedule the algorithm will find it.

Thanks to these results obtained in the monoprocessor (one operator) case we study our problem of distribution and scheduling in the multiprocessor case (several operators) with more complex constraints that we did previously, i.e. with precedence, and one latency constraint equal to a unique periodicity constraint. We proved this problem is NP-hard for systems with precedence and periodicity constraints, we proposed a heuristic which takes into account the communication times. We proved that operations with periods which are not co-prime can not be scheduled on the same operator. We proved this problem is NP-hard for systems with precedence and latency constraints, we proposed a heuristic which takes into account the communication times. This heuristic uses the schedulability results obtained in the case of one operator concerning the three relations *II*, *Z* and *X* between pairs of operations, on which latency constraints are defined. These latter results prove that the best way of scheduling operations is to avoid scheduling, between the first and the last operation of a latency constraint, operations which do not belong to this latency constraint. Finally we proved this problem is NP-hard for systems with precedence, periodicity and latency constraints, we proposed a heuristic which takes into account the communication times. We proved that operations belonging to the same latency constraint must have the same period. A direct consequence is that the operations belonging to the same pair or to pairs which are in relation *II*, *Z* or *X* must have the same period. So, the heuristic may use the main ideas of the heuristic for the case of precedence and latency constraints and of the heuristic for the case of precedence and periodicity constraints. The performances of these three heuristics were compared to those of exact algorithms. The numerical results show that the heuristics are definitely faster then the exact algorithms for all cases when the heuristics find a solution.

The aforementioned scheduling problems only takes into account periodic operations. Aperiodic operations issued from aperiodic events, usually related to control, must be handled on-line. Presently we take them into account off-line by integrating the control-flow in our data-flow model, well suited to distribution, and by maximizing the control effects. We study relations between control-flow and data-flow in order to better exploit their respective advantages. Finally, we study the possibility to mix off-line for periodic operations and on-line approaches for aperiodic operations.

In the case of integrated circuit the potential parallelism of the algorithm corresponds exactly to the actual parallelism of the circuit. However, this may lead to exceed the required surface of an ASIC or the number of CLB (Combinatorial Logic Block) of a FPGA, and then some operations must be sequentially repeated several times in order to reuse them, reducing in this way the potential parallelism to an actual parallelism with less logic functions. But reducing the surface has a price in terms of time, and also in terms of surface but of a lesser importance, due to the sequentialization itself (instead of parallelism) performed by the finite state machines (control units) necessary to implement the repetitions and the conditionings. Then, we are seeking a compromise taking into account surface and performances. Because these problems are again of NP-hard complexity, we propose greedy and iterative heuristics in order to solve them [21].

Finally, we plan to work on the unification of multiprocessor heuristics and integrated circuit heuristics in order to propose “automatic hardware/software partitioning” for co-design, instead of the usual manual one. The most difficult issue concerns the integration in the cost functions of the notion of “flexibility” which is crucial for the choice of software versus hardware. However, this optimization criterion is difficult to quantify because it mainly relies on user’s expertise.

3.3.3. Automatic code generation

As soon as an implementation is chosen among all the possible ones, it is straightforward to automatically generate executable code through an ultimate graphs transformation leading to a distributed real-time executive for the processors, and to a structural hardware description, e.g. synthetizable VHDL, for the specific integrated circuits.

For a multicomponent each operator (resp. each communicator) has to execute the sequence of operations (resp. communication operations) described in the implementation graph. Thus, this graph is translated in an “executive graph” [28] where new vertices and edges are added in order to manage the infinite and finite loops, the conditionings, the inter-operator data dependences corresponding to “read” and “write” when the communication medium is a RAM, or to “send” and “receive” when the communication medium is a SAM. Specific vertices, called “pre” and “suc”, which manage semaphores, are added to each read, write, send and receive vertices in order to synchronize the execution of operations and of communication operations when they must share, in mutual exclusion, the same sequencer as well as the same data. These synchronizations insure that the real-time execution will satisfy the partial order specified in the initial algorithm. Executives generation is proved to be dead-lock free [25] maintaining the properties, in terms of events ordering, shown thanks to the synchronous language semantics. This executive graph is directly transformed in a macro-code [26] which is independent of the processor. This macro-code is macro-processed with “executive kernels” libraries which are dependent of the processors and of the communication media, in order to produce as many source codes as there are processors. Each library is written in the best adapted language regarding the processors and the media, e.g. assembler or high level language like C. Finally, each produced source code is compiled in order to obtain distributed executable code satisfying the real-time constraints.

For an integrated circuit, because we associate to each operation and to each control unit an element of a synthetizable VHDL library, the executable code generation relies on the typical synthesis tools of integrated circuit CAD vendors like Synopsis or Cadence.

3.3.4. Fault tolerance

For the applications we are dealing with, if real-time constraints are not satisfied, this may have catastrophic consequences in terms of human beings lost or pollution, for example. When a fault occurs despite formal verifications which allow safe design by construction, we propose to specify the level of fault the user accepts by adding redundant processors and communication media. Then, we extended our optimization heuristics in order to generate automatically the redundant operations and data dependences necessary to make transparent these faults. Presently, we only take into account “fail silent” faults. They are detected using “watchdogs”, the duration of which depends on the operations and data transfers durations. We first obtained results in the case of processor faults only, i.e. when the communication media are assumed error free. Then, we studied, in addition to processors faults, media faults.

We propose three kinds of heuristics to tolerate both faults. The first one tolerates a fixed number of arbitrary processors and links (point-to-point communication medium) faults. It is based on the software redundancy of operations. The second one tolerates a fixed number of arbitrary processors and buses (multipoint communication medium) faults. It is based on the active software redundancy of the operations and the passive redundancy of the communications with the fragmentation in several packets of the transfered data on the buses. The third one tolerates a fixed number of arbitrary processors and communication media (point-to-point or multipoint) faults. It is based on a quite different approach. This heuristic generates as much distributions and schedulings as there are of different architecture configuration corresponding to the possible faults. Then, all the distributions and schedulings are merged together to finally obtain a resulting distribution and a scheduling which tolerates all the faults.

Finally, we propose a heuristic for generating reliable distributions and schedulings. The software redundancy is used to maximize the reliability of the distribution and scheduling taking into account two criteria: the minimization of the latency (execution duration of the distributed and scheduled algorithm onto the architecture) and the maximization of the reliability of the processors and the communication media.

As soon as the redundant hardware is fully exploited, “degraded modes” are necessary. They are specified at the level of the algorithm graph by combining delays and conditionings.

4. Application Domains

4.1. Embedded systems

Our generic field of applications is termed as “Embedded Systems”, meaning all kind of equipments including software and electronical parts, apart from regular mainstream computers. This includes transportation vehicles (cars, aircrafts,...), mobile robotics, communicating appliances such as mobile telephones, or System-on-Chip design. These fields are further described in their particular aspects below.

Common to all embedded systems are: the *reactivity* aspect (supervising or simply interacting with an outside environment); the demand for *safety* (often critical); the *heterogeneity of models*, incumbing a multiplicity of engineering techniques from different scientific disciplines. Large development projects usually involve many providers for components, and the design flow is far less linear as in traditional software: it includes customarily various stages of modeling, prototyping, simulation/evaluation/dimensioning, manual reencoding, and iterative (re)design space exploration. Component reuse is also usually a great concern (as components might be physical preexisting parts). Here formal models and notations can greatly help keep the traceability of the design process, and justify some of its steps for soundness and accurary across the flow and its various actors.

4.2. Mobile robotics, automotive and transportation

Keywords: *embedded automotive electronics*.

With increasing functionality demands in powertrain, body comfort or telematics applications, modern cars are becoming complex systems including Real-Time OS (OSEK), complex data buses (CAN or TTP/FlexRay), with distributed intelligent sensors and powerful computing power. Software and electronic are now becoming a prominent part of both the price and added value. Still, no “ideal” hardware/software architecture is yet standardized, and the development methodologies are still at infancy. Proposals for high-level modeling and infrastructure platform organization have been proposed, as in the EAST-EEA and AutoSar consortium. We are taking part in the first initiative, mostly through the AAA methodology which proposes computer-aided mapping of synchronous applications onto heterogeneous platforms. This methodology was amply demonstrated in the framework of CyCab mobile robotic applications.

4.3. Mobile phones and other communicating objects

Such systems usually combine intensive (multimedia) data processing with mode control switching, and wireless or on-chip communications. The issue is often here to integrate design techniques for the three domains (data, control, communications), while preserving modular independence of concern as much as possible. At high-level modeling this translate into combining models of computations that are state-oriented (imperative) or datapath-oriented (declarative), with appropriate communication models, while preserving the sound semantics of the systems built from all such kinds of components. Dedicated architecture platforms here usually associate general-purpose processor (ARM) with specific DSP coprocessors. In the future the level of integration should become even higher, with the corresponding challenges for design methodologies.

4.4. System-on-Chip design

While design of digital circuits is already a fairly complex development process, involving many modeling and programming stages, together with intensive testing and involved low-leval synthesis and place-and-route techniques, SoC desig adds yet new complexity dimensions to this process. Fully synchronous designs are not feasible anymore, and custon IP reuse becomes mandatory to integrate full processor cores into a new designs. New aproaches are being proposed, which try to depart only as little as possible form the synchronous/cycle-accurate prevailing design techniques, while allowing more timing flexibility at interfaces between blocks. These aaproaches are generally flagged as GALS (Globally-Asynchronous/Locally-Synchronous). They usually put a stress on proper mathematical modeling at every stage, thereby revisiting and associating known models with new intent. Synthesis seen as model-transformation seems here a nice way to bring some of the OMG MDA schemes into true existence.

5. Software

5.1. SyncCharts/Esterel

Keywords: *Esterel*, *SyncCharts*, *compilation*, *static analysis*, *verification*.

Participants: Charles André, Robert de Simone, Dumitru Potop.

The main software development activities concerning synchronous formalisms went to the ESTEREL TECHNOLOGIES company as it was spun-off from the former Meije team. We still carrry some experimental development on the former academic versions of Esterel and SynCharts, mostly to validate new algorithmic model transformations or analysis.

5.2. SynDEx

Participants: Christophe Gensoul, Arnaud Rouanet, Yves Sorel.

SynDEx is a system level CAD software implementing the AAA methodology for rapid prototyping and for optimizing distributed real-time embedded applications. It can be downloaded free of charge, under INRIA copyright, at the url: http://www.syndex.org. It provides the following functionalities:

- specification and verification of an application algorithm as a directed acyclic graph (DAG) of operations, or interface with specification languages such as the synchronous languages providing formal verifications, AIL a language for automobile architectures, Scicos a Simulink-like language, AVS for image processing, CamlFlow a functional data-flow language, etc,
- specification and verification of a “multicomponent” architecture as a graph composed of programmable components (processors) and/or specific non programmable components (ASIC, FPGA), all interconnected through communication media (shared memory, message passing),
- specification of the algorithm characteristics, relative to the hardware components (execution and transfer time, period, memory, etc), and specification of the real-time constraints to satisfy (latencies, periodicities),

- exploration of possible implementations (distribution and scheduling) of the algorithm onto the multicomponent, performed manually or automatically with optimization heuristics, and visualization of a timing diagram simulating the distributed real-time implementation,
- generation of dedicated distributed real-time executives, or configuration of general purpose real-time operating systems: RTlinux, Osek, etc. These executives are deadlock free and based on off-line policies. Dedicated executives which induce minimal over-head are built from processor-dependent executive kernels. Presently executives kernels are provided for: ADSP21060, TMS320C40, TMS320C60, i80386, MC68332, MPC555, i80C196 and Unix/Linux workstations. Executive kernels for other processors can be easily ported from the existing ones.

The distribution and scheduling heuristics, as well as the timing diagram, help the user to parallelize his algorithm and to explore architectural solutions while satisfying real-time constraints. Since SynDEx provides a seamless framework from the specification to the distributed real-time execution, formal verifications obtained during the early stage of the specification, are maintained along the whole development cycle. Moreover, since the executives are automatically generated, part of tests and low level hand coding are eliminated, decreasing the development cycle duration.

SynDEx was evaluated by the main companies involved in the domain of distributed real-time embedded systems, and is presently used to carry out new products at Robosoft, MBDA, and Mitsubishi Electric ITE.

5.3. SynDEx-IC

Participants: Mohamed Akil [Professor ESIEE Noisy-Le-Grand], Thierry Grandpierre, Pierre Niang [PhD. student ESIEE Noisy-Le-Grand], Yves Sorel.

SynDEx-IC is a CAD software for the design of non-programmable components such as ASIC or FPGA for which, the application algorithm to implement is specified with the graph model of the AAA methodology. It is developed in collaboration with the team A2SI of ESIEE. It allows to specify the application algorithm like in SynDEx, and automatically synthesizes the data path and the control path of the specific integrated circuit as a synthetizable VHDL program while real-time and surface constraints are satisfied. Because these problems are again of NP-hard complexity, we propose greedy and iterative heuristics based on "loop-unrolling" of the algorithm graph, in order to solve them.

This integrated circuit synthesis was tested on image processing applications. Using SynDEx-IC we specified and implemented, for example, several digital image filters onto the XC2S100 SPARTAN FPGA based on executive kernel for synthetizable VHDL. We extended the architecture model in order to support the specificities of FPGA: internal memories, configuration of computational units, communication unit with other components. Using this extended architecture model, we modelled the architecture of different commercial Boards (Virtex II pro card from Xilinx, Stratix board from Altera) and applyied the graph transformations needed to obtain an optimized hardware implementation on this kind of architecture.

Such non-programmable components designed with SynDEx-IC may be in turn used in SynDEx in order to specify complex multicomponent architectures composed of non-programmable and programmable components all together interconnected. Presently, both softwares SynDEx and SynDEx-IC are separated, the hardware/software partitioning phase of co-design being done manually. We plan in the future to integrate SynDEx-IC in an unique software environment, and also to provide heuristics to automaticallyperform hardware/software partitioning.

6. New Results

6.1. A UML profile for Real-Time Embedded System modeling

Participants: Charles André, Gérard Cristau, Arnaud Cuccuru, Robert de Simone.

Embedded system modeling has always relied upon graphical diagrams with strong intuitive meaning: component netlists, data-flow block diagrams, or hierarchical state diagrams to name a few. Our concern was to define a superset of the UML2.0 OMG standard (technically called a profile), amenable to the precise semantics of synchronous and GALS systems. The motivation is to use UML-based editors and modeling tools as front-ends for the powerful compilation, synthesis, implementation and analysis techniquees developed around our mathematical model foundations. This includes formal modeling of execution platforms, with software and hardware aspects, in the platform-based design approach advocated in our team.

This year we investigated several modeling domains. Our technical propositions were partially phrased as proposal contributions to the MARTE profile [51], [55], currently being built as a response to the RFP call at OMG. We worked more specifically on the definition of a general time model, that can be used to match "physical" metric time, as well as more logical single-clock or multiclock time bases, where the time units can be measured by the successive occurrences of any given event, not necessarily linked with physical periodic regular timing. We also worked on the introduction of regular iterator constructs, both for architetcural structure and for application behaviors. Finally we also investigated hardware component and architecture representation, in a way that comply with platform-based design and allocation.

We conducted these researches in the context of the CARROLL PROTES project and of the proMARTE international consortium (see "Contracts" section).

6.2. Formal modeling of latency-insensitive systems

Participants: Julien Boucaron, Robert de Simone, Jean-Vivien Millo.

The Latency-Insentive approach to GALS design proceeds in several steps: a synchronous "golden specification", used as reference, is first de-synchronized into amarked graph (a specific subclass of conflict-free Petri Nets). Then mandatory latencies are introduced at several locations, and the system is re-synchronized in a way compliant with them. Each step requires the introduction of protocol elements needed to control the flow of signal events. In the second stage the virtually infinite (unbounded) Fifo queues introduced in the first stage are replaced by fixed-size so-called *Relay-Stations*, with both restricted buffering capacities, and also back-pressure wired-in mechanisms. The computation units themselves are surrounded by sequential wrappers, allowing their (gated) clocks to trigger computations only when all signal values have arrived.

We provided the first formal modeling of all these eleements included in the Latency-Insensitive version of de-synchronized systems, and proved their correctness by establishing a number of temporal properties in the ssytem instantiations with a given number of such elements (typically three computation units and several relay-stations). The results were presented at FMGALS, a specialized workshop on Formal Methods for GALS systems [54].

6.3. UML Patterns for hardware/software architectures

Participants: Charles André, Frédéric Mallet, Marie-Agnès Péraldi.

A first contribution has been presented at the OMER3 (Object-oriented Modeling of Embedded Real-Time Systems) Workshop, at Paderborn (Germany), October 2005 [52]. The approach is twofold:

- a UML2.0 and SysML-based architecture description including functional and structural specification, resource allocations and QoS modelling;
- a quantitative analysis with formal description of behaviour using Time Petri Net.

The quantitative analysis explores the solution space and proves the existence of valid schedulings. A refinement process allows different levels of description of the application.

Works in progress come within the scope of the Model Driven approach to the specification and design of embedded systems. We will propose metamodels for behavior, structure, and analysis, and the associated model transformations.

6.4. AAA models

Participants: Liliana Cucu, Thierry Grandpierre, Patrick Meumeu, Nicolas Pernet, Mickaël Raulet, Yves Sorel.

We clearly stated the differences between our algorithm model and the typical model used by the real-time community in the monoprocessor case. We justified that “release time” and “deadline” are not necessary and that “strict periodicity” is sufficient, for applications using signal, image and control processing algorithms. Also, we justified that the “latency” constraints is more powerful than “deadline” in the sense that on the one hand two deadlines are necessary to impose a latency constraint, and on the other hand these two deadlines over-constraint the system of operations. We proposed a systematic method to transform a problem specified with latency model in a problem specified with deadlines more usual to the real-time community [57]. We proposed a new model to specify conditionings inside data-flow graph such that its implementation onto distributed architecture is effective, whereas this problem is often neglected leading to dramatic errors observed at run-time. This model was successfully used to develop automatic translator from other languages based on control graphs, such as Scicos and SyncCharts[61]. Finally, we introduced “preemption” in our algorithm model such that it is possible to take into account accurately its cost during off-line scheduling.

We slightly extended our architecture model to take into account program and data memories inside multicomponents. This model is simpler than the hierarchical one described in [25] but is sufficient to be exploited by the code generator in order to minimize the number of buffers by re-using them with coloured graphs techniques.

We extended the architecture model in order to support the specificities of FPGA: internal memories, configuration of computational units, communication unit with other components (cf. SynDEx-IC section).

6.5. Scheduling and Optimization

Participants: Liliana Cucu, Omar Kermia, Patrick Meumeu, Alix Munier, Nicolas Pernet, Yves Sorel.

6.5.1. Preemtive off-line scheduling in the monoprocessor case

The two last years we had preliminary results about the introduction of the preemption into the scheduling problem of real-time system with precedence and strict periodicity constraints in the monoprocessor case. Classical off-line approaches based on preemption, such as RM (Rate Monotonic), DM (Deadline Monotonic), EDF (Earliest Deadline First), LLF (Least Laxity First), etc, give schedulability conditions but assuming the cost of the preemption is negligible compared to the duration of the operations. Clearly this is difficult to calculate accurately because, if the cost of one preemption is easy to determine, it is not the same for the number of preemptions. Thus, the main drawback of these approaches is that the designer must take as many margins as there are operations in the system to guarantee that the real-time behavior will conform to the schedulability condition.

We adopted a constructive approach in order to find a schedulability condition which counts the exact number of preemptions to take into account accurately the cost of the preemption. To the best of our knowledge there is only few results about this problem. Constructive means we construct all the possible preemptive schedules relatively to the strict periodicity of the operations and the possible durations taken by each operation, assuming the duration of an operation cannot be greater than its period. That was done after determining the cases where the operations are not schedulable because of their periods, the preemption beeing allowed or not. We proved the two following impossibilities. Firstly, we proved it is not possible to find a schedule as soon as two operations have co-prime periods. Secondly, we proved it is not possible to find a schedule as soon as the time elapsed between the first start times of two operations is a multiple of the greatest common divisor of their periods. Then, we called Ω_λ the set of operations potentially schedulable. We used the scheduling policy which schedules operations according to the increasing order of their periods thanks to the result given in [20] taking into account the precedence constraints. We defined a partition of Ω_λ into the three following subsets. We called V_c the subset of operations such that the start time of a repetition of an operation occurs while the processor is executing another repetition of an already scheduled operation. Such operations cannot be scheduled because the precedences impose that an operation may be preempted, only by another one already scheduled. Again this is due to the result previously cited. We called V_d the subset of operations such that their periods grow harmonically, i.e. the period of any operation is a multiple of the period of the previous one. We called V_i the subset of operations such that it exists, at least, two operations whose periods are not multiple. For V_d we proposed a scheduling algorithm which constructs, as previously discribed, all the schedules, and in addition gives the number of preemptions. That corresponds to a schedulability analysis taking into account accurately the cost of the preemption when the cost of one preemption is known for each operation.

6.5.2. Non-preemtive off-line scheduling in the multiprocessor case

We studied the distribution and scheduling problem for system with precedence and strict periodicity constraints. The latter constraint means each operation of the graph is labeled by its period beside its duration. First, each operation is repeated within the LCM (Least Common Multiple) of all periods of operations (hyper-period), allowing to unroll the graph over the hyper-period. We propose an algorithm adding the missing edges. We repeat each operation n times, n is equal to $n = (P/T)$ where the period of this operation is T and P is the value of the hyper-period. In order to maintain the data transfers between operations during the unrolling we must add new edges between the repetitions of the same operations, and between the repetitions of different operations. The operations are classified (clustered) by their periods in such way that each class corresponds to each different period. The resulting class are assigned to the processors as following: if the number of processors is greater than the number of classes we must create as many classes as there are of processors for which no class was assigned. We propose a load-balancing algorithm such that the number of class is equal to the number of processors. Otherwise if the number of class is lower or equal to the number of class we assign the resulting class to the processors (by ascendant period order), and some class may not be assigned.

We proposed an extension of our distribution and scheduling greedy heuristic in the case of precedence, and one latency constraint equal to a unique periodicity constraint. We apply this extended heuristic on the unrolled and assigned graph in order to distribute and schedule the periodic operations onto the multiprocessor. We compute the complexity of the proposed heuristic, and perform a performance comparison which shows its effectiveness.

6.5.3. Mixing off-line and on-line scheduling for aperiodic operations

The AAA methodology deals with distributed hard real-time systems. Thus it is presently based on off-line heuristics to find a schedule satisfying in addition to precedence and periodicities, complex real-time constraints as latency constraints introduced in [19]. A latency constraint concerns a couple of operations and defines the maximum amount of time which can separate the beginning of the first operation and the end the second one. Off-line scheduling reduces execution overhead because it does not require a scheduler as on-line scheduling do. The drawback of these approaches is that there are not suited for aperiodic operations, that is to say operations whose occurrence time is unknown contrary to periodic operations. Nevertheless, we aim at handling on-line aperiodic operations while satisfying all the real-time constraints of the periodic operations scheduled off-line.

Previous works on aperiodic operations rely on specific scheduling algorithms for periodic operations (Rate Monotonic, Earliest Deadline First) and a simplified model (periods equal to deadlines, no latency constraints, no precedence). Consequently, they are unusable for solving our problem. The only method which does not impose such restrictions is the Slot Shifting [65]. It consists in splitting the scheduling in execution intervals. These intervals are delimited by the deadlines of each operation, and by inter-processor communication operations. Then, the spare capacity of each interval, that is to say the amount of time which is available in this interval to execute aperiodic operations, is computed. This approach is independent on the scheduling algorithm or heuristic employed to perform the off-line scheduling, and takes into account communications costs.

We extended the Slot Shifting algorithm to deal with several latency constraints. First, assuming that we have an off-line schedule of the periodic operations which satisfy their constraints (deadlines and latencies), we have to handle, on-line, aperiodic operations still satisfying the previous constraints. We construct a subset of these constraints such as satisfying them implies to satisfy the whole set of constraints. Second, we give a method to use slot shifting with several latency constraints, and we show that latency constraints offer more in-line spare capacity than deadlines do. This method consists in transforming each latency constraint in a deadline for the second operation of the couple of operations concerned by a latency constraint. Nevertheless, this transformation only takes place on-line when the first operation of the couple of operations begins its execution. Indeed, before this is known it is not possible to translate the latency constraint in a deadline. Our method dealing with dealdine and latency constraints constraints for distributed systems, and is totally independent of the heuristic (because distributed implies NP-hard problem) used to perform the off-line scheduling.

For soft-real time applications we continued to investigate the adaptive scheduling technique. We proposed a new heuristics based on the one used in the case of precedence, and one latency constraint equal to a unique periodicity constrain, extended by a PID controller which manages aperiodic operations [53].

6.5.4. *Cyclic scheduling problem with latencies*

The execution of a real-time application may be considered as a cyclic scheduling problem for which a set of generic tasks constrained by precedence and resource relations are executed a large number of time. The model AAA [20] also considered non classical constraints between the generic tasks (called latency constraints) to model additional temporal relations between their executions. We proved that this new class of constraints may be modelled using an extension of the basic cyclic scheduling problem [17], [18].

Firstly, we studied the problem without resource constraint. We expressed an original necessary and sufficient condition for the existence of a schedule. Since there is no resource constraint, the earliest schedule has a maximal throughput : we characterized the structure of this schedule, and we showed that it is always possible to build a periodic schedule (for which the schedule of one iteration is repeated) with the same throughput. We also developed a polynomial time algorithm to compute the necessary and sufficient condition of schedulabilty, the maximal throughput and a periodic schedule [46]. The conclusions for the AAA model are double :

- If the latencies are fixed, a periodic schedule with a minimum latency may be polynomially computed.
- If the throughput is fixed, we can polynomially compute a periodic schedule which optimizes linear functions of the latencies using linear programming.

In both cases, the periodic schedule obtained is optimal (since it has the same characteristic as the earliest schedule).

Then we considered the corresponding sequencing problem with some restrictions on the structure of the latencies corresponding to the practical problem. The existence of a feasible schedule is an NP-complete problem. We proved that, if a periodic schedule exists, it is always possible to increase its throughput such that no idle slots remains on the processor in the steady state. This property allows us to develop an original branch and bound method to solve the problem of the existence of a schedule. For each node, several original necessary conditions allows us to reduce significantly the number of children.

We noticed than, in more than 95 percent of randomly generated case, the non existence of a schedule is checked using the necessary and sufficient condition obtained for the problem without resource constraints. Moreover, up to 50 tasks, 99 percent of randomly generated instances are solved with less than 10.000 nodes. The perspectives of this work are the development of efficient algorithms to solve quickly the problem of the existence of a periodic schedule for one or a fixed number of processors for the AAA model.

6.5.5. Memory optimization

Because the amount of memory is usually limited in embedded processors we aim at minimizing memory utilization. Presently, due to the data-flow model, the RAMs necessary for storing local data memory (inside the processor) and for storing data to transfer (between processors) are specified in the implementation model by as many `alloc` vertices as there are of data dependences. A buffer is associated to each `alloc` vertex during the code generation. Obviously, this method consumes too much memory while some buffers should be reused when there are not used latter on. We proposed several heuristics based on coloured graph technics in order to minimize the number of buffers. This problem is particularly complex when conditionings are considered.

6.6. Fault tolerance

Participants: Hamoudi Kalla, Yves Sorel.

The last two years we studied several heuristics which are extensions of the one used in AAA/SynDEx tolerating a fixed number of arbitrary processors, links (point-to-point communication medium), and buses (multipoint communication medium) faults. Moreover, we proposed a heuristic generating reliable distributions and schedulings and taking into account two criteria: the minimization of the latency (execution duration of the distributed and scheduled algorithm onto the architecture) and the maximization of the reliability of the processors and the communication media. These heuristics are all based on automatic software redundancy. They have been detailed in [42] and implemented in the version 6 of SynDEx.

6.7. CyCab experimentations

Participants: Céline Roussel, Nicolas Pernet, Yves Sorel.

Since three years we have been experimenting our SynDEx software onto different applications running onto the CyCab. We experimented firts a stop when obstacle application using ultra-sound sensors. Then, we began to experiment a CyCab recognition application in order to carry out virtual train of CyCabs, i.e. several CyCabs follow the preceding CyCab, the first one being driven by a human person. In order to carry out the CyCab recognition application we designed and implement different image processing algorithms that were integrated on a low cost vision system based on a webcam and an embedded PC. This low cost smart that is connected to the embedded PC of the CyCab was developed in collaboration with the team COSI of ESIEE. We made first tests to use this vision system in a control loop algorithm aiming at maintaining a security distance between two CyCabs whatever the speed of the preceding one is.

6.8. Improvements in SynDEx

Participants: Christophe Gensoul, Benoît Miramond, Quentin Quadrat, Arnaud Rouanet, Yves Sorel.

Version 6 of SynDEx was the latest major release. It was completely redesigned, in OCaml instead of C++ which was used previously. It provides new features such as hierarchical modularity (essential for top-dow design of huge application), repetition (equivalent to `For...Do...`) and conditioning (equivalent to `If...Then...Else...`) constructs. SynDEx-6.0 is available since April 2002. In 2005 we had a lively of bug-fixing activity, mainly due to the success met by this software environment, and the many reports by industrial users developing practical applications for innovative products. We provide further details on the kind of applications conducted at MBDA, Mitsubishi Electric ITE, and PSA in the contractual section below.

We had to improve the code generator of SynDEx, mainly because the time to generate automatically the real-time distributed executives, and in a smaller proportion the amount of code itself, increase dramatically with the size of the application. The code generator was entirely redesigned and reprogrammed in Ocaml leading to an acceleration of a factor ten. Some simple optimizations were performed slightly decreasing the amount of generated code. Moreover, to minimize the memory we tested a first version of a heuristic based on coloured graph technics which reuses the data buffers when the code is generated.

We developped and tested a new type of heuristic which was not of type “greedy but” of type “simulated anealing”, however based on the same cost function.

We introduced in the GUI of SynDEx a code editor in order to help the designer to write the applicative kernels used by the Gnu m4 macro-processor to produce the executable code dependent of the architecure, from the macro-code that is independent of the architecture, automatically generated by SynDEx.

Finally, we continued to improve the software architecture of the OCaml SynDEx program, to help with its maintenance and its further evolutions.

7. Contracts and Grants with Industry

7.1. ST MIcroelectronics

Participants: Julien Boucaron, Robert de Simone.

This collaboration takes place with the team of Marc Benveniste, from ST Smart Card division located at the Rousset site. The goal is to study abstraction/refinement techniques for SoC specification, in a way inspired from the B Method, but using synchronous languages concepts instead. Julien Boucaron PhD thesis is largely funded on this contract. Clearsy is another partner in the project,and they are using the B method for similar aims. We are currently investigating a common case study around interrupt handling in embedded processors, with USB protocol support.

7.2. Texas Instruments

Participants: Julien Boucaron, Jean-Vivien Millo, Robert de Simone.

This contract was initiated by Texas Instruments because of a need they felt to better understand the theory of *Latency-insensitive* systems, its connexions with synchronous semantics, and more generally GALS description models. We are conducted an extensive bibliographic survey, checking all along the potential (good or bad) implications for efficient SoC modeling, according to our previous experience gathered on the field. In the second year we have focused on the survey of off-line scheduling techniques for latency equalization. In the future we intent to include our partners from Columbia University into this activity, through the INRIA joint-team programme.

7.3. CARROLL PROTES

Participants: Charles André, Gérard Cristau, Anaud Cuccuru, Robert de Simone, Yves Sorel.

CARROLL is a joint initiative between Thales, CEA, and INRIA to launch collaborative projects, mostly on UML and/or MDA based topics. The PROTES project is dedicated to the construction of a specific UML profile for Real-Time Embedded systems that would cover our needs, and a standardization effort towards the OMG.

Early this year our MARTE (Modeling and Analysis of Real-Time Embedded systems) “RFP” (Request-For-Proposal) document was successfully voted at the OMG. This lead the way to the gathering of a global consortium in order to submit a response to this proposal. It currently comprises the main software editing houses in the domain (IBM-Rational, Artisan, I-Logix*, TeleLogic, SofTeam), several large end-users (Thales, Alcatel, Lockheed-Martin, IBM, Mentor Graphics), and several academic partners.

We made a number of proposals for inclusion in the draft that were directly taken from our work on application and architecture modeling, taking up the synchronous or GALS approach. The initial submission was successfully completed at the last OMG Technical Meeting of 2005 in Burlingame, and the revised version (version 1.0) is to be approved in June 2006 (or, more likely, at the next Technical OMG meeting).

This work involved participation on the part of Aoste team members to several OMG Technical meetings in the US (Burlingame, December 2004 and 2005; Boston, June 2005), as well as progress meetings in France, internal to Protes, about every two months.

7.4. MBDA

Participants: Christophe Gensoul, Quentin Quadrat, Arnaud Rouanet, Yves Sorel.

MBDA develops with AAA/SynDEx a new automatic guidance application involving an algorithm with more than 6000 operations executed at different periods, whereas the architecture is made of several PowerPC and ASICs all interconnected through a crossbar.

7.5. Mitsubishi Electric ITE

Participants: Arnaud Rouanet, Yves Sorel.

Mitsubishi Electric ITE develops with AAA/SynDEx software radio applications which involve a lot of modes supported by the conditioning feature and in each mode a lot of signal processing algorithms, each algorithm uses processing repetition feature, whereas the architecture is composed of several TMS320C60 DSP (Digital Signal Processor). Also, we collaborate onto new heuristics using “genetic algorithm” methods allowing multi-criteria optimization (latency and cadence) in order to enhence our “greedy” heuristic.

8. Other Grants and Activities

8.1. Regional collaborations

8.1.1. CIM PACA

Participant: Robert de Simone.

This ambitious regional initiative is intended to foster collaborations between local PACA industry and academia partners on the topics of microelectronic design, though mutualization of equipments, resources and R&D concerns. We are actively participating in the Design Platform (one of the three platforms launched in this context). Other participants are UNSA, CNRS (I3S and LEAT laboratories), L2MP Marseille, CMP-ENSE Gardanne on the academic side, and Texas Instruments, Philips, ST Microelectronics, ATMEL, and Esterel Technologies on the industrial side. , on a project named Spec2RTL. This group of students should be funded on a par basis between industry and local authorities.

8.1.2. Competitivity Poles

Participants: Robert de Simone, Yves Sorel.

We are planning to take part in a number of large projects currently under definition,in the cases of both PACA-based SCS (Secure communicating Solutions), as well as Systemtic located in the Paris area. These labeling were already recognized as supportive in the making of recent national projects (see below).

8.2. Nation-wide collaborations

8.2.1. Relations with other INRIA teams

We have strong ties with INRIA teams ESPRESSO and DaRT through the PROTES initiative on synchronous and more generally RTE (Real-Time Embedded) modeling in UML. We conduct joint work with POP-ART on fault tolerance and adaptive scheduling for robotic applications. Together with the S4 team we regularly attend the same events gathering the “Synchronous languages” community.

We also collaborate with IMARA team which develops with SynDEx new applications onto automatic vehicles such as the CyCab, and with METALAU on coupling of Scilab/Scicos with AAA/SynDEx. Historical links are preserved with the team SOSSO, on adaptive scheduling for applications mixing soft and hard real-time.

8.2.2. RNTL platform Open-eMbeDD

Participants: Charles André, Robert de Simone, Yves Sorel.

This collaboration has just been accepted in the RNTL Call for Proposals on Embedded System design. It is due to start beginning of year 2006. The partners are: INRIA, CEA-List, Thales, Airbus, France Telecom, CS, LAAS, and VERIMAG. The focus is on the use of model-driven approaches to combine various specification formalisms, analysis and modeling techniques, into an interoperable framework. We contribute in this direction through the MARTE profile, our work on compilation-by-transformation of synchronous program, and the AAA-SynDEx methodology.

8.2.3. RNTL project MemVatex

Participants: Marie-Agnès Peraldi, Yves Sorel.

This collaboration has just been accepted in the RNTL Call for Proposals on Embedded System design. It is due to start beginning of year 2006. The partners are: Siemens-VDO, INRIA, CEA-List, CNRS-UTC, and Embelec. The focus is on the use of a model-based approach to the modeling of automotive components, and specially on the validation and tracability aspects of such a design flow.

8.2.4. RNTL project ECLIPSE

Participants: Cyril Faure, Yves Sorel.

The goal of Eclipse is to providea seamless environment from specification/modeling/simulation with Scilab/Scicos to optimized implementation with AAA/SynDEx. It was started in 2003 for a duration of two years and a half. The partners are: PSA, CS-SI, CRIL, ESIEE, INRIA. A translator from Scicos to SynDEx algorithm is now available. It has been used by PSA to model, simulate and implement onto a multi-processor architecture, a complex application coupling an assisted steering with an ESP.

8.2.5. CNRS AS CAT

Participants: Marie-Agnès Peraldi-Frati, Charles André, Frédéric Mallet.

The Specific Action (AS) CAT (Composants Architectures Temps réel) has been created at CNRS with the objective of finding new research directions in the domain of component based engineering for real-time and embedded architecture design.

The partners of the AS are working in different domains such as software engineering, realtime systems and model checking. The main idea is to promote the component at the early stage of the developpement phase, and to integrate in a component informations about its execution support and some physical characteristics such (performances, QoS ...).

The delivery report of the AS has been published in March 2005 [62].

8.3. European collaborations

8.3.1. IST Network of Excellence ARTIST2

Participants: Julien Boucaron, Robert de Simone, Yves Sorel.

Our participation here consists essentially (as for many other partners) in attending working group presentation meetings (without real collaborative work so far). We follow particularly the work of Working Group 1 on Hard Real-Time, with focus on Synchronous languages, Time-Triggered architectures and fixed priority scheduling.

8.4. Research exchange visits

Participants: Julien Boucaron, Luca Carloni, Robert de Simone, Stephen Edwards, Dumitru Potop, Olivier Tardieu.

We started a collaboration with the research group of Stephen Edwards at Columbia University (New York), under the INRIA Associated Team exchange program. Stephen Edwards and Olivier Tardieu came to visit our team in June/July, while Julien Boucaron and Robert de Simone went to New York in september. Dumitru Potop went to Columbia in december.

The work in this collaboration is mainly focused on teh mathematical semantics of synchronous languages [50], and its extension to GALS systems. Efficient compilation schemes are also tackled. A book on Esterel semantics is under preparation.

9. Dissemination

9.1. Leadership within scientific community

Robert de Simone was program committee member for Memocode'05. He is also member of the *Commission de Spécialiste UNSA 27^{e} section*, and INRIA representative to the CIM PACA regional initiative on Microelectrnics design; this includes being appointed to the Strategic Committee of the ARCCIS mother association, and member of the Board of Administrators of the Design Platform association. As INRIA leader of the CARROLL PROTES project he attended several OMG Technical Meetings at various locations in the US. Finally, he is member of the International Advisory Board for the CRC Press on Embedded Systems.

Robert de Simone and Dumitru Potop, together with Jean-Pierre Talpin (Espresso, team, IRISA) wrote a Handbook chapter on Synchronous Languages and formalisms [49]. Charles André also wrote a chapter on Real-Time systems in a collective Encyclopedia work on Computer Science and Information Systems, published in french by Vuibert [45]. Dumitru Potop and Olivier Tardieu, together with Stephen Edwards (U. Columbia), are currently writing a book on Esterel's semantics.

Yves Sorel leads the Theme C Working Group (Adequation Algorithme Architecture) of the PRC-GDR ISIS (Information Signal Images et viSion). He is Program Member for the following conferences and workshops: JFAAA, ERTS, EUSIPCO, GRESTSI, JEAI, SYMPA, RTS. He is permanent member of the LCPC Scientific Committee, of the CARLIT-ONERA Scientific Committee, and of the DETIM-ONERA Evaluation and Orientation Committee. He participated to the following juries PhD.: I. Alzeer, A. Gamatie, F. Peix, HDR: M. Shawky, O. Deforges.

Robert de Simone was reviewer for the HDR thesis of Jean-Pierre Talpin, and the PhD thesis of Jan Mikac and David Merchat (VERIMAG).

9.2. Teaching

Robert de Simone taught courses on Formal Methods and Models for Embedded Systems in the STIC Research Master program of the university of Nice/Sopia-Antipolis (UNSA), and at ISIA-EMP (an engineering school located in Sophia-Antipolis), each time for approximately 24h.

Yves Sorel teaches at ESIEE (an engineering school located in Noisy-le-Grand), in the Research Master cursus at the University of Orsay Paris 11, and at ENSTA (an engineering school located in Paris), on topics comprising the AAA methodology, formal modeling and optimization of distributed embedded systems.

Charles André is a Professor at the University of Nice-Sophia Antipolis, department of Electrical Engineering. He teaches sequential circuits, discrete event systems, computer architecture and real-time programming. He also teaches "synchronous programming" and "UML for engineering systems" at the university polytechnic: EPU (options electrical engineering (Elec) and sofware engineering (SI)) and in the STIC research master.

Marie-Agnès Peraldi-Frati gives courses at different cursus levels of UNSA :

- A course and labs on Real-Time Distributed Systems (30h) in the STIC research master (Embedded Systems) and the STIC professional Master (STREAM01/EPU).
- A course and labs on Synchronous programming and real-time systems (60H) for LP SIL students (L3).
- A course and labs on Computer architectures for first year level of the IUT Informatique (L1).

She is responsible for the option "Informatique embarquée et réseau sans fil" of the LP SIL cursus.

Frédéric Mallet is Associate Professor at the University of Nice-Sophia Antipolis, departement of Infomatics. He teaches Object-oriented Programming at all levels from very beginners to Master level courses, and on all platforms from javacard, PDA, to standard operating systems. He also teaches Computer Architecture to undergraduate students.

10. Bibliography

Major publications by the team in recent years

[1] S. ALAM, R. IBBETT, F. MALLET. *Performance Evaluation of Local Communications: A Case-study*, in "Proceedings of the 15th International Conference on Parallel Distributed Computings and Systems, CA,USA", november 2003.

[2] C. ANDRÉ. *Semantics of SSM (Safe State Machine)*, Esterel Technologies, April 2003, http://www.esterel-technologies.com.

[3] C. ANDRÉ, F. BOULANGER, A. GIRAULT. *Software Implementation of Synchronous Programs*, in "Proceedings of the Second International Conference on Application of Concurrency to System Design", IEEE Computer Society, 2001, p. 133-142.

[4] C. ANDRÉ, M.-A. PERALDI-FRATI, J.-P. RIGAULT. *Integrating the Synchronous Paradigm into UML: Application to Control-Dominated Systems*, in "UML ≪2002≫, Dresden (D)", Springer-Verlag, October 2002, p. 163–178.

[5] C. ANDRÉ, J.-P. RIGAULT. *Variations on the Semantics of Graphical Models for Reactive Systems*, in "SMC'02", IEEE Press, 2002.

[6] C. ANDRÉ, R. DE SIMONE. *Synchronous Programming : Properties within a Reaction*, in "JESA", vol. 36, n^o 7, 2002, p. 891–903.

[7] C. ANDRÉ, R. DE SIMONE. *Towards a Synchronous UML Profile ?*, in "Proceedings of the first Workshop on Specification and Validation of UML models for Real-Time and Embedded Systems (SVERTS'03)", 2003, http://www-verimag.imag.fr/EVENTS/2003/SVERTS/PAPERS-WEB/14-Andre-UMLsync.pdf.

[8] C. ANDRÉ. *Representation and Analysis of Reactive Behavior: a Synchronous Approach*, in "Computational Engineering in Systems Applications (CESA)", IEEE-SMC, 1996, p. 19–29.

[9] C. ANDRÉ, F. BOULANGER, M.-A. PERALDI-FRATI, J.-P. RIGAULT, G. VIDAL-NAQUET. *Objects and Synchronous Programming*, in "RAIRO-APII-JESA", vol. 31, n^o 3, 1997.

[10] C. ANDRÉ, M.-A. PERALDI-FRATI, J.-P. RIGAULT. *Scenario and Property Checking of Real-Time Systems Using a Synchronous Approach*, in "4th International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC 2001), Magdeburg, Germany", IEEE, May 2001.

[11] A. BENVENISTE, G. BERRY. *The Synchronous Approach to Reactive and Real-Time Systems*, in "Proceedings of the IEEE", vol. 79, n^o 9, September 1991, p. 1270-1282.

[12] A. BENVENISTE, P. CASPI, S. EDWARDS, N. HALBWACHS, P. L. GUERNIC, R. DE SIMONE. *Synchronous Languages Twelve Years Later*, in "Proceedings of the IEEE", January 2003.

[13] G. Berry. *The Foundations of Esterel*, Foundations of Computing Series, MIT Press, 2000, http://www-sop.inria.fr/esterel.org/Html/Downloads/Doc/StartDocument.htm.

[14] G. Berry. *The Constructive Semantics of Pure Esterel*, 1999, http://www-sop.inria.fr/esterel.org/Html/Downloads/Doc/StartDocu

[15] G. Berry. *The Esterel Language Primer*, 1999, http://www-sop.inria.fr/esterel.org/Html/Downloads/Doc/StartDocument.htm.

[16] F. Boussinot, R. de Simone. *The Esterel Language*, in "Proceedings of the IEEE", September 1991.

[17] J. Carlier, P. Chrétienne. *Problèmes d'ordonnancement*, Masson, 1988.

[18] G. Cohen, P. Moller, J. Quadrat, M. Viot. *Algebraic tools for the performance evaluation of discrete event systems*, in "IEEE Proceeding: special issue on Discrete Event Systems", vol. 77, n^o 1, 1989, p. 39–58.

[19] L. Cucu, R. Kocik, Y. Sorel. *Real-time scheduling for systems with precedence, periodicity and latency constraints*, in "Proceedings of 10th Real-Time Systems Conference, RTS'02, Paris, France", March 2002.

[20] L. Cucu, Y. Sorel. *Schedulability condition for systems with precedence and periodicity constrainsts without preemption*, in "Proceedings of 11th Real-Time Systems Conference, RTS'03, Paris", March 2003.

[21] A. Dias, C. Lavarenne, M. Akil, Y. Sorel. *Optimized Implementation of Real-Time Image Processing Algorithms on Field Programmable Gate Arrays*, in "Proceedings of Fourth International Conference on Signal Processing, ICSP'98, Beijing, China", October 1998, http://citeseer.ifi.unizh.ch/dias98optimized.html.

[22] V. V. Dongen, G. R. Gao, Q. Ning. *A Polynomial Time Method for Optimal Software Pipelining*, in "Conference on Algorithms and Hardware for Parallel Processing", http://citeseer.ist.psu.edu/vandongen92polynomial.html.

[23] A. Girault, H. Kalla, M. Sighireanu, Y. Sorel. *An Algorithm for Automatically Obtaining Distributed and Fault-Tolerant Static Schedules*, in "Proceedings of International Conference on Dependable Systems and Networks, DSN'03, San Francisco, California, USA", June 2003.

[24] A. Girault, C. Lavarenne, M. Sighireanu, Y. Sorel. *Fault-Tolerant Static Scheduling for Real-Time Distributed Embedded Systems*, in "Proceedings of 21st International Conference on Distributed Computing Systems, ICDCS'01, Phoenix, USA", April 2001.

[25] T. Grandpierre. *Modélisation d'architectures parallèles hétérogènes pour la génération automatique d'exécutifs distribués temps réel optimisés*, Ph. D. Thesis, Université de Paris Sud, Spécialité électronique, 2000.

[26] T. Grandpierre, C. Lavarenne, Y. Sorel. *Modèle d'exécutif distribué temps réel pour SynDEx*, Rapport de Recherche, n^o 3476, INRIA, August 1998, http://www.inria.fr/rrrt/rr-3476.html.

[27] T. Grandpierre, C. Lavarenne, Y. Sorel. *Optimized Rapid Prototyping For Real-Time Embedded Heterogeneous multiprocessors*, in "Proceedings of 7th International Workshop on Hardware/Software Co-Design, CODES'99, Rome, Italy", May 1999.

[28] T. GRANDPIERRE, Y. SOREL. *From Algorithm and Architecture Specification to Automatic Generation of Distributed Real-Time Executives: a Seamless Flow of Graphs Transformations*, in "Proceedings of First ACM and IEEE International Conference on Formal Methods and Models for Codesign, MEMOCODE'03, Mont Saint-Michel, France", June 2003.

[29] N. HALBWACHS. *Synchronous Programming of Reactive Systems*, in "Computer Aided Verification", 1998, p. 1-16, http://citeseer.ist.psu.edu/10686.html.

[30] L. KAOUANE, M. AKIL, Y. SOREL, T. GRANDPIERRE. *From algorithm graph specification to automatic synthesis of FPGA circuit: a seamless flow of graph transformations*, in "Proceedings of 13th international conference on Field-Programmable Logic and Applications, FPL'03, Lisbon, Portugal", September 2003.

[31] C. LAVARENNE, O. SEGHROUCHNI, Y. SOREL, M. SORINE. *The SynDEx Software Environment for Real-Time Distributed Systems, Design and Implementation*, in "Proceedings of European Control Conference, ECC'91, Grenoble, France", July 1991.

[32] C. LAVARENNE, Y. SOREL. *Performance Optimization of Multiprocessor Real-Time Applications by Graph Transformations*, in "Proceedings of Parallel Computing Conference, PARCO'93, Grenoble, France", September 1993.

[33] C. LAVARENNE, Y. SOREL. *Modèle unifié pour la conception conjointe logiciel-matériel*, in "Traitement du Signal", vol. 14, n^o 6, 1997.

[34] E. A. LEE, D. G. MESSERSCHMITT. *Static Scheduling of Synchronous Data Flow Programs for Digital Signal Processing*, 1987.

[35] F. MALLET, R. IBBETT, S. ALAM. *An Extensible Clock mechanism for Computer Architecture Simulations*, in "Proceedings of the 13th International Conference on Modeling and Simulation, Los Angeles", may 2002.

[36] F. MALLET, R. IBBETT. *JavaHase: Automatic Generation of Applets from HASE Simulation Models*, in "Proceedings of the 2003 Summer Computer Simulation Conference, montreal", august 2003.

[37] N. PERNET, Y. SOREL. *From Specification to Optimized Implementation of Distributed Real-Time Embedded Systems Mixing Control and Data Processing*, in "Proceedings of ISCA 16th International Conference: Computer Applications in Industry and Engineering, CAINE'03, Las Vegas Nv", November 2003.

[38] Y. SOREL. *Massively Parallel Systems with Real Time Constraints, the Algorithm Architecture Adequation Methodology*, in "Proceedings of Conference on Massively Parallel Computing Systems, MPCS'94, Ischia, Italy", May 1994.

[39] Y. SOREL. *Real-Time Embedded Image Processing Applications using the AAA Methodology*, in "Proceedings of IEEE International Conference on Image Processing, ICIP'96, Lausanne, Switzerland", September 1996.

[40] A. VICARD, Y. SOREL. *Formalization and Static Optimization for parallel implementations*, in "Proceedings of Workshop on Distributed and Parallel Systems, DAPSYS'98, Budapest, Hungary", September 1998.

[41] A. VICARD. *Formalisation et optimisation des systèmes informatiques distribués temps réel embarqués*, Ph. D. Thesis, Université de Paris Nord, Spécialité informatique, 1999.

Doctoral dissertations and Habilitation theses

[42] H. KALLA. *Génération automatique de distributions/ordonnancements temps réel fiables et tolérant les fautes*, Ph. D. Thesis, Institut National Polytechnique de Grenoble, Spécialité Systèmes et Logiciel, 2005.

[43] L. KAOUANE. *Formalisation et optimisation d'applications s'exécutant sur architecture reconfigurable*, Ph. D. Thesis, Université de Marne-La-Vallée, Spécialité Informatique, 2005.

Articles in refereed journals and book chapters

[44] C. ANDRÉ. *Langages et formalismes synchrones*, in "RS - JESA", 2005.

[45] C. ANDRÉ. *L'approche synchrone pour le développement des systèmes temps réel*, in "Encyclopédie de l'informatique et des systèmes d'information, tome 1: La dimension technologique des systèmes d'information", 2005.

[46] A. M. KORDON. *A generalization of the Basic Cyclic Scheduling Problem*, in "Submitted to Journal of Scheduling", june 2005.

[47] M. RAULET, F. URBAN, J.-F. NEZAN, C. MOY, O. DÉFORGES, Y. SOREL. *Rapid Prototyping For Heterogeneous Multicomponent Systems: An MPEG-4 Stream Over An UMTS Communication Link*, in "Journal of Applied Signal Processing (JASP)", 2005.

[48] E. VECCHIÉ, R. DE SIMONE. *Syntax-driven optimizations for Reachable State Space construction of Esterel programs*, in "International Journal of Embedded Systems", April 2005.

[49] R. DE SIMONE, D. POTOP, J.-P. TALPIN. *The Synchronous Hypothesis and Synchronous Languages*, in "Embedded Systems Handbook", chapter of the Embedded Systems Handbook, CRC Press, 2005.

[50] R. DE SIMONE, O. TARDIEU. *Loops in Esterel*, in "ACM Transactions on Embedded Computing Systems", 2005.

Publications in Conferences and Workshops

[51] C. ANDRÉ, A. CUCCURU, J.-L. DEKEYSER, R. DE SIMONE, C. DUMOULIN, J. FORGE, T. GAUTIER, S. GÉRARD, F. MALLET, A. RADERMACHER, L. RIOUX, T. SAUNIER, Y. SOREL. *MARTE: a new OMG profile RFP for the Modeling and Analysis for Real-Time Embedded Systems*, in "Workshop on UML for SoC, DAC 2005", June 2005.

[52] C. ANDRÉ, F. MALLET, M.-A. PERALDI-FRATI. *Real-Time Architecture Description and Quantitative Analysis using UML*, in "OMER3 Workshop", 2005.

[53] T. AYAV, Y. SOREL. *Feedback Control Static Scheduling for Real-Time Distributed Embedded Systems*, in "Proceedings of the 11th IEEE International Conference on Embedded and Real-Time Computing Systems

and Applications, RTCSA'05, Hong Kong, China", August 2005.

[54] J. BOUCARON, J.-V. MILLO, R. DE SIMONE. *Another glance at relay stations in latency-insensitive design*, in "FMGALS'05", july 2005.

[55] A. CUCCURU, R. DE SIMONE, T. SAUNIER, G. SIEGEL, Y. SOREL. *P2I: An Innovative MDA Methodology for Embedded Real-Time System*, in "EuroMicro'05", 2005.

[56] L. CUCU, Y. SOREL. *Condition d'ordonnançabilité pour systèmes temps réel non-préemptif à contraintes de précédences, de périodicités et de latences*, in "Actes du 6ème congrès de la Société Française de Recherche Opérationnelle et d'Aide à la Décision, ROADEF'05, Tours, France", February 2005.

[57] L. CUCU, Y. SOREL. *Periodic real-time scheduling: from latency-based model to deadline-based model*, in "Proceedings of the Multidisciplinary International Conference on Scheduling: Theory and Applications, MISTA'05, New-York, USA", July 2005.

[58] E. DEKNEUVEL, C. ANDRÉ. *Intelligent Sensor Modelling Using a Synchronous Approach*, in "3rd International Conference on Computing, Communication and Control Technologies", 2005.

[59] N. PERNET, Y. SOREL. *A design method for implementing specifications including control in distributed embedded systems*, in "Proceedings of the 10th IEEE International Conference on Emerging Technologies and Factory Automation, ETFA'05, Catania, Italy", September 2005.

[60] N. PERNET, Y. SOREL. *Spécification et implantation des systèmes distribués temps réel de contrôle et traitement de données*, in "Actes des Journées Francophones sur l'Adéquation Algorithme Architecture, Dijon, France", January 2005.

[61] N. PERNET, Y. SOREL. *Transformations de spécifications incluant du contrôle en spécification flot de données pour implantation distribuée*, in "Actes de la Conférence Modélisation des Systèmes Réactifs, MSR'05, Grenoble, France", October 2005.

Internal Reports

[62] A.-M. DEPLANCHE, M.-A. PERALDI-FRATI, F. MALLET, C. ANDRÉ, ET AL.. *AS 195 Composants et Architectures Temps réel — Rapport de synthèse et Annexes*, Technical report, n° RI2005, IRCCyN, 2005.

Bibliography in notes

[63] R. BALAKRISNAN, K. RANGANATHAN. *A Textbook of Graph Theory*, Springer, 2000.

[64] J. DENNIS. *First Version of a Dataflow Procedure Language*, in "Lecture Notes in Computer Sci.", vol. 19, Springer-Verlag, 1975, p. 362-376.

[65] G. FOHLER. *Joint Scheduling of Distributed Complex Periodic and Hard Aperiodic Tasks in Statically Scheduled Systems*, in "Procedings of IEEE Real-Time Systems Symposium", 1995, p. 152-161.

[66] N. HALBWACHS. *Synchronous programming of reactive systems*, Kluwer Academic Publishers, Dordrecht Boston, 1993.

[67] E. LEE, D. MESSERSCHMITT. *Synchronous Data Flow*, in "Proceedings of the IEEE", 1987, vol. 75, no. 9.

[68] J. LEUNG, W. J.. *On the complexity of fixed-priority scheduling of periodic real-time tasks*, in "Performance Evaluation(4)", 1982.

[69] Z. LIU, C. CORROYER. *Effectiveness of heuristics and simulated annealing for the scheduling of concurrent task. An empirical comparison*, in "PARLE'93, 5th international PARLE conference, June 14-17, Munich, Germany", November 1993, p. 452-463.

[70] C. LIU, J. LAYLAND. *Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment*, in "Journal of the ACM", 1973.

[71] C. MEAD, L. CONWAY. *Introduction to VLSI systems*, Addison-Wesley, 1980.

[72] V. PRATT. *Modeling concurrency with partial orders*, in "International Journal of Parallel Programming", vol. 15, n^{o} 1, 1986.

[73] A. ZOMAYA. *Parallel and distributed computing handbook*, McGraw-Hill, 1996.