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 - Méditerranée - Paris - Rocquencourt*

THEME COM

# Table of contents

# 1. Team

*Aoste is a joint team with UNSA (university of Nice/Sophia-Antipolis) and CNRS, through their UMR I3S. It is also co-located between Sophia-Antipolis and Rocquencourt. Project members originate from the former INRIA Tick and Ostre teams, together with the I3S Sports team.*

**Head of project-team**
Robert de Simone [ INRIA Research Director, HdR ]

**Vice-head of project-team**
Yves Sorel [ INRIA Research Director ]

**Administrative assistant**
Sophie Honnorat [ INRIA, part-time, until October ]
Nelly Maloisel [ INRIA, part-time ]
Viviane Rosello [ UMR I3S, part-time ]
Isabelle Strobant [ INRIA, part-time, since October ]

**Research scientists**
Charles André [ Professor at UNSA, HdR ]
Frédéric Mallet [ Researcher, on leave from UNSA since October ]
Marie-Agnès Peraldi-Frati [ Associate Professor at UNSA ]
Dumitru Potop-Butucaru [ Researcher, Inria ]

**PhD. students**
Julien Boucaron [ ST Microlectronics contractual funding ]
Anthony Coadou [ MESR scholarship ]
Omar Kermia [ INRIA scholarship ]
François Lagarde [ codirected with CEA-List ]
Jean-François Le Tallec [ BDE regional scholarship with ScaleoChip ]
Aamir Mehmood [ Pakistanese government scholarship, from December ]
Patrick Meumeu Yomsi [ INRIA scholarship ]
Jean-Vivien Millo [ BDE regional scholarship with ST Microelectronics ]

**Technical Staff**
Benoît Ferrero [ Project engineer, OpenEmbedd RNTL platform ]
Fadoi Lakhal [ Project engineer, OpenEmbedd RNTL platform ]
Su Young Lee [ Project engineer, MemVatex RNTL project ]
Djamel Louar [ Project engineer, MemVatex RNTL project ]
Christophe Gensoul [ Software development staff ]

**Student interns**
Anthony Coadou [ Research Master, UNSA ]
Ouafae Aidi [ Research Master University Paris 11 ]
Aguibou Barry [ ECE Engineering School Paris ]
Quentin Quadrat [ EPITA Engineering School Paris ]

**External collaborators**
Liliana Cucu [ Post-doctoral fellow ]
Thierry Grandpierre [ Associate Professor at ESIEE Noisy-Le-Grand ]

# 2. Overall Objectives

## 2.1. Introduction

**Keywords:** *AAA methodology*, *Esterel*, *SynDEx*, *SyncCharts*, *SysML*, *UML*, *adequation*, *architectural models*, *automatic verification*, *code distribution*, *compilation*, *embedded systems*, *formal semantics*, *hardware synthesis*, *hardware/software codesign*, *model-driven engineering*, *multiprocessors*, *optimization*, *program analysis*, *real-time*, *scheduling*, *synchronous reactive formalisms*, *system-level design*, *systems-on-Chip*.

The main objective of the Aoste team is to promote the formal design of embedded systems, with their intrinsic concurrent, distributed and real-time aspects. We build upon previous experience by team members on ESTEREL, SYNCCHARTS and synchronous reactive formalismes, SYNDEX and the *Algorithm-Architecture Adequation* methodology (AAA) to reach this goal. Domains of application range from transport (automotive, aircrafts), electronic appliances (mobile phones, HDTV,...), to System-on-Chip design.

For the sake of presentation we split our activities in two: the definition of adequate description formalisms and models, with their formal semantics for precise representation of real-time embedded systems on the one hand; the design of relevant model transformation and analysis techniques to cover an efficient design flow methodology based on these formalisms on the other hand. By transformation here we mean various compilation and implementation techniques (distribution and scheduling), considered as transforming a higher-level model into a lower-level one. Analysis instead work at the same modeling level.

Modern embedded systems combine complexity and heterogeneity both at the level of applications (with a mix of control-flow modes and multimedia data-flow streaming), and at the level of execution platforms (with increasing parallelism and multicore architectures). We use synchronous reactive models, their globally-asynchronous/locally-synchronous, and multiclock extensions and formal semantics to describe application's behaviors as well as architectural and timing constraints. The transformations under consideration include combinations of spatial distribution and temporal scheduling to map the application onto the platform, as well as various optimization techniques to improve compilation. Static and dynamic (model-checking) analysis are also used to provide insights on correctness and efficiency of models according to the prescribed formal semantics.

## 2.2. Highlights of the year

The first book on ESTEREL and its compilation has been published this year [16], cowritten by Dumitru Potop together with Gérard Berry (the father of the language), and Stephen Edwards from Columbia University(New York). It is already reprinted.

The MARTE UML profile for *Modeling and Analysis of Real-Time Embedded* systems was adopted at OMG for its first revised version (in fact this is version 1.0 in UML terminology). This completes a three-year long standardization effort from our group, conducted jointly with CEA-List and Thales. The profile is fully available on the official OMG site http://www.omgmarte.org.

A joint modeling framework associating k-periodic schedules of Marked-Event Graphs with k-periodic routing in the Kahn Process Network philosophy was designed, under the name of *Kahn Event Graphs* [36].

A necessary and sufficient schedulability condition has been obtained, extending rate monotonic analysis (RMA) for hard real-time systems while using the *exact* cost of preemption rather than an approximation [33].

# 3. Scientific Foundations

## 3.1. Languages and Models

**Keywords:** *AAA methodology*, *Esterel*, *SynDEx*, *SyncCharts*, *UML*, *synchronous formalisms*.

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

### *3.1.1. Synchronous reactive formalisms*

Synchronous reactive models are those where concurrent processes all run at the speed of a common global clock, in a discrete logical time framework [48], [50], [39]. Simultaneous behaviors in a single global instant are thus allowed and even often required. The main issue is then to ensure correct causal behavior propagation inside a given instant, as well as across instants. Programs may indeed exist for which there is no such possible execution, and they have to be recognized and banned as ill-behaved. This issue is unfortunately too often overlooked by many existing specification formalisms in the field, as in Matlab/Simulink, VHDL/Verilog/SystemC, or StateCharts, to name a few.

*Synchronous reactive formalisms*, such as ESTEREL/SYNCCHARTS, LUSTRE/SCADE, or SIGNAL/POLYCHRONY were designed explicitly to offer dedicated programming models and proper specification primitives, to study advanced semantic issues and useful transformations schemes, in the realm of synchronous models [39], [50], [14], [1], [2]. ESTEREL and SYNCCHARTS are control-oriented state-based formalisms [41], [37], while *Lustre* and *Signal* are declarative data-flow based formalisms.

These languages were mostly born out of INRIA or of collaborating French teams. The INRIA spin-off Esterel-technologies now markets ESTEREL STUDIO and SCADE. SynDEx and the AAA methodology also born out of INRIA rely greatly on synchronous assumptions.

A book on Esterel compilation was recently co-authored by one of our team member [16].

### *3.1.2. Globally-Asynchronous/Locally-Synchronous (GALS) and multiclock extensions*

Over time the basic synchronous model was shown to suffer expressivity lacks when considering complex heterogenous systems. For such large designs the precise temporal allocation of instants to behaviors on a single global clock would impediment the general development flow, as it requires too many details from the designer. Instead, reasoning with several distinct clock domains, possibly loosely-coupled, provides necessary specification freedom. But this should not impair the underlying semantics, with all its correctness requirements; such is the price to pay to preserve the powerful implementation schemes as considered formerly.

*GALS* models were introduced to relax some of the synchrony hypothesis. They are used in the theory of *Latency-Insensitive Design (LID)*, which allows de-synchronization then re-synchronization of former synchronous specifications into implementations that are synchronous again, but this time will accommodate mandatory extra latencies provided by users. The final design is proven to be equivalent to the former, and requires no modification of the local (computation) IP (Intellectual Property) component blocks; it only adds an extra buffering and synchronization layer in between them. Similarly *multiclock* designs allow to consider hierarchical clock structures where not all components are active simultaneously. In both cases active research is conducted to study practical conditions under which extended models can be transformed into equivalent synchronous ones (in a way that respects behavior, and not necessarily timing). On the other hand the multiclock or LID extensions may provide extra properties that can be taken into account to improve even more the efficiency of embedded implementations. Endochrony is an example of such property. Expansion of multiclock or GALS systems into plain synchronous ones takes the form of operational scheduling of concurrent applications.

LID theory bears strong ties with the former theories of Weighted Marked Graphs and its timed schedulings, either dynamic or static, such as described in [42], [52]. Results of this nature were already used in the context of software pipelining [47], and could certainly be better exploited in LID context.

### *3.1.3. UML modeling diagrams for Real-Time Embedded applications*

Graphical models and diagrams were popular in Embedded System design long before the advent of the UML. In fact the ancestors of UML state, activity and sequence diagrams, namely statecharts, Petri nets and Message Sequence Charts, originated from this field. The long-standing object orientation philosophy of UML

then disclosed many differences in spirit with the former semantics of such models (in the intuitive sense, since UML provides no formal semantics to its behavioral diagrams). This trend has been reversed with the introduction of components and ports in ROOM, then of block diagrams in SysML. But still, no semantics.

The benefits of expressing our models inside the UML framework would be to allow usage of UML editors and tools (textual or graphical), as well as to expose our result to a larger community. The new SysML profile for System Engineering is already a proof of interest towards such a modeling style. Of course this would require from us the definition of a timed semantics both formal and compatible with whatever there is of semantic concerns in current UML 2.1. We are currently taking part in a consortium named PROMARTE which, in part, aims at providing such an official UML profile at OMG level. The profile itself is named MARTE, which stands for *Modeling and Analysis of Real-Time Embedded systems*. The consortium was initiated by INRIA, Thales, and CEA-List, as part of a CARROLL initiative. The profile RFP (Request For Proposals) was voted early 2005, and the initial submission was voted in June 2007. The profile is currently undergoing first revision in the Ad-Hoc Finalization Task Force.

We are exploiting our contributions to MARTE and our experience gained in meta-modeling in a number of downstream activities. We use model transformation techniques to define such a transformation of real-time distributed applications modeled in Marte into Time Petri Nets representations for analysis purpose. In collaboration with other INRIA teams (Espresso and DaRT mostly) we provide meta-models for SYNDEX, and later for other synchronous formalisms to be coupled to MARTE.

Depending on the application domain a number of formalisms have been introduced for the representation of system structure and behavior, that were sometimes later embedded into some sort of UML syntax to benefit from tool vendor support. These include AADL in the avionics domain, EAST-ADL and AUTOSAR in the automotive domain, UML4SOC, and to a lesser extend, SPIRIT in the SoC design field. In all these cases we are seriously considering the connection with our MARTE contributions, to ensure our model enjoys enough semantic expressive power so as to represent formerly the underlying concepts to these profiles.

### *3.1.4. AAA methodology*

The purpose of the AAA methodology is to provide independent modeling of *applications* and supporting *architectures* in a first step. The mapping of applications onto architectures is realized only in a subsequent step, and is subject to algorithmic optimizations relative to the various timing constraints and costs involved. This approach is called *Algorithm-Architecture Adequation* (AAA) [55].

AAA allows the designer to specify "application algorithms" (functionalities) and "Embedded platform" (composed of hardware resources such as processors and specific integrated circuits all together interconnected) with graph models. Consequently, all the possible implementations of a given algorithm onto a given architecture can be described in terms of graphs transformations. An implementation consists in distributing and scheduling a given algorithm onto a given embedded platform. The adequation amounts to explore, manually and/or automatically, the possible implementations, such that the real-time, and the hardware resources are best used. Furthermore, the adequation results are used, as an ultimate graph transformation, to automatically generate two types of code: dedicated distributed real-time executives 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 a fault occurs. Thus AAA generates automatically the redundant operations as well as data dependences (software redundancy) necessary to make transparent these faults when some hardware components fail.

Real-time embedded systems are, first of all, "reactive systems" that obligatorily must react infinitely to each input event of the infinite sequence of events it consumes, such that "input rate" and "latency" constraints are satisfied. The input rate corresponds to the delay between two successive input events, these events may be periodic, sporadic or aperiodic. The latency corresponds to the delay between an input event consumed by the system and an output event produced by the system in reaction, through the computation of several operations, to this input event. The term event is used here in a broad sense. It represents an information which is present relatively to a discrete time reference. When hard (critical) real-time is considered, off-line approaches are preferred due to their predictability and low overhead, and when on-line approaches are unavoidable, mainly

to take into account aperiodic events, we intend to minimize the cost of 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, whereas usually only periodicity is considered. 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. In addition to these timing constraints, the systems must satisfy embedding constraints, such as power consumption, weight, volume, memory, etc, it turns out that hardware resources must be minimized. In the most general case architectures are distributed, and composed of several programmable components (processors) and several specific integrated circuits (ASIC: Application Specific Integrated Circuit or FPGA: Field Programmable Gate Array) all together interconnected with possibly different types of communication media. We call such heterogeneous architectures "multicomponent" [56].

The AAA methodology is implemented in 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, Scilab/Scicos, or UML2.0, 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.

*3.1.4.1. Algorithm model*

Our algorithm model is an extension of the well known data-flow model from Dennis [45]. It is a directed acyclic hyper-graph (DAG) that we call "conditioned factorized data dependence graph", 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 dependence defines a partial order on the operations execution, 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. First, infinite (*resp.* finite) repetition of a sub-graph pattern is considered in order to specify the reactive aspect of real-time systems (*resp.* in order to specify the finite repetition of a sub-graph consuming different data, in a way similar to a loop in imperative languages "data potential parallelism"). Second, "states" are allowed to model the case when data dependences are necessary between infinite repetitions of the sub-graph pattern, which would otherwise introduce cycles that must be avoided by specific vertices called "delays" (similar to $z^{-n}$ in automatic control). Third, conditions guarding an operation by a control dependence play a role similar to conditional control structure in imperative languages, allowing the execution of alternative subgraphs. Delays combined with conditioning allow the programmer to specify FSM (Finite State Machine) necessary for describing "mode changes", e.g. some control law graph is performed when the motor is the state "idle" whereas another one is performed when it is in the state "permanent". Finite repetition and conditioning both exploit the hierarchy principle.

*3.1.4.2. Architecture model*

The typical coarse-grain architecture models such as the PRAM (Parallel Random Access Machines) and the DRAM (Distributed Random Access Machines) [58] are not enough detailed for the optimizations we intend to perform. On the other hand the very fine grain RTL-like (Register Transfer Level) [54] models are too detailed. Thus, our model of multicomponent architecture is also a directed graph [8], 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)" (selection of a memory or arbitration to a shared memory for an operator), 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 in order to abstract architectural details. Although this model seems very basic, it is the result of several studies [8] in order to find out the appropriate granularity allowing, on the one hand to provide accurate optimization results, and on the other hand to obtain as quickly as possible these results during the rapid prototyping phase.

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.

*3.1.4.3. Implementation model*

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 [49] 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, are used 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 dependence" (compared to data dependence), 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" [57]. 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 obtained differently 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 [46].

## 3.2. Transformations and analysis

**Keywords:** *Compilation*, *allocation*, *architecture exploration*, *formal verification*, *optimization*, *synthesis*.

**Participants:** Charles André, Julien Boucaron, Liliana Cucu, Robert de Simone, Omar Kermia, Frédéric Mallet, Patrick Meumeu, Jean-Vivien Millo, Marie-Agnès Péraldi, Dumitru Potop-Butucaru, Yves Sorel.

### *3.2.1. Compilation of synchronous reactive formalisms*

Syntactic constructs in synchronous languages are always provided meaning in terms of formal operational semantics on well-defined interpretation models. As a result, all kinds of compilation/synthesis, analysis and verification, or optimization methods can readily be caracterized as formal transformations on such mathematical models [14], [11]. While implementations may seek various optimality criteria, they should always in principle be established as "correct-by-construction" following semantic equivalence with the basic version.

Concerning Esterel/SyncCharts, compilation was first realized in the 1980's as an expansion 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 execution 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 introduced, as in particular in the PhD thesis of Dumitru Potop-Butucaru; such compilation patterns rely 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). Correctness is in this context ensured by a stronger, more restrictive *acyclicity* criterion, which provides a static evaluation order for signal propagation.

The advanced compilation techniques defined in the PhD thesis of Dumitru Potop-Butucaru [10], where the current hierarchical state and input signals are considered to determine the actual parts of the (concurrent) code which have to be active in the current reaction, are finding their way in the industrial compiler distributed by ESTEREL TECHNOLOGIES, where they are marketed as "fast C" compilation. Meanwhile work by Olivier Tardieu (formerly PhD student in the Aoste team and currently holding a postdoctoral position at Columbia University, NY) is establishing a number of internal semantic-preserving transformations from Esterel to (slightly augmented) Esterel [15]. These transformation steps combine the introduction of various `goto`-like primitives so that programs are rewritten into forms that enjoy "good" natural properties (such as *non-schizophrenia* for instance). These extensions are promising as a way to introduce synchronous formalisms to support several stages in the design flow of complex, heterogeneous embedded systems.

### *3.2.2. Dynamic analysis, automatic verification and model-checking*

Because of the static structure of the Embedded Systems we consider, their models are frequently finite-state when considering only the control aspects. Thus, under some appropriate data abstraction, they are amenable to automatic verification (often known as "model-checking"). This is in fact put in use in many of the previous compilation schemes for Esterel/SyncCharts, which rely on computations performed at compile time, while these computations converge only in this finite-state case.

In the past years we developed the XEVE model-checker for Esterel, based on BDD symbolic representation techniques. We developed a number of approaches to partition the computation of the reachable state space according to the program structure [13]. In turn these methods have been shown to hold close relations with efficient compilation techniques as well. While these activities are currently in a somehow "stand-by" mode in the team, it is important to maintain some competency in this domain because of its interactions with other fields.

### *3.2.3. Time Refinement*

In our design approach applications and architectural platforms are first described independently, both with their behavioral and structural aspects, and possibly with their logical time constraints. Asynchonous processes lead to loosely coupled or independent (virtual) clocks, synchronous systems rely rather on common global clocks. Clocks are of a more logical nature in applications, of more physical nature in the platform. The

application mapping of application functions onto platform resources realizes an association between the whole set of clocks by solving constraints. In many simple cases the solutions, which represent the scheduling of the application of the target platform, can be represented syntactically as "first-order objects" of the modelization framework.

We used this general philosophy in our work on static scheduling of latency-insensitive synchronous systems. Starting from a fully synchronous early specification, mandatory latencies on long communication wires are imposed from outside. The specification is de-desynchronized, then resynchronized to form a latency-independent version [43]. Then hardware implementation constraints impose the form and placement of specific buffering elements, named *relay-stations*, to realize the mandatory flow control. But as shown by classical results [42], [38], the resulting global behavior is ultimately $k$-periodic (for strongly connected systems). Then an explicit schedule can be effectively constructed, and used to optimized the allocation of *relay-stations*.

### *3.2.4. Distributed Real-Time Scheduling and Optimization*

From algorithm and architecture models it is possible to express the space of all possible implementations. We must then choose a particular one for which the constraints are satisfied, while at the same time optimizing some criteria. In the case of a multiprocessor architecture the problem of finding the best distribution and scheduling of the algorithm onto the architecture, is known to be of NP-hard complexity.

The first problem we address consists in considering, 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 (input rate) for all the operations. We propose several heuristics based on the characterization of the operations (resp. communication operations) relatively to the operators (resp. communicators) which associate to each pair (operation, operator) (resp. (communication operation, communicator)) a set of values representing an execution time, a power consumption, a surface, etc. For example we minimize the total execution time of the algorithm (makespan) onto the distributed architecture, 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 [49]. We mainly develop "greedy heuristics" because they are very fast, and thus, well suited to rapid prototyping of realistic industrial applications. In this type of applications the algorithm graphs we deal with may count in the order of ten thousand vertices, and the architecture graph may have several dozens of vertices. However, we extend these greedy heuristics to iterative versions 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, led us to consider new problems with 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 [53] 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 [52] 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 its period occurs. In this case we give a schedulability condition for graph of operations with precedence and periodicity constraints in the non-preemptive case [3].

We also formally defined the notion of latency which is more powerful, for the applications we are interested in, than the usual notion of "deadline" that is not able to impose directly a timing constraint on a pair of

operations, indeed when two deadlines are used the problem becomes overconstrained. Both operations of the pair are connected by at least one path as usually found in "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 the problem of distribution and scheduling in the multiprocessor case (several operators) with more complex constraints than in the case previously studied, 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. The 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. These results show that the heuristics are definitely faster than the exact algorithms for all cases when the heuristics find a solution [4].

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. 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 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 conditions. 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 [46].

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.

### *3.2.5. Generation of Distributed Code*

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" [8] 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 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 [8] 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.

### *3.2.6. Fault tolerance*

For the applications we are dealing with, if the real-time constraints are not satisfied, this may have catastrophic consequences in terms of human beings lost or pollution. 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 [7]. 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 [6].

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. Mobile robotics, automotive and transportation

**Keywords:** *embedded automotive electronics*, *transportation*.

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 provided, 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.

We are also involved in the definition of a methodology based on the new version of the ADL EAST-ADL2.0. EAST-ADL 2.0 is a UML2.0 profile dedicated to automotive systems. Our aim is first to formally introduce architectural concepts such as temporal characteristics, complex scheduling and resource allocation constraints in the EAST process, and to find out the transformations rules from EAST model elements to formalisms such as Petri Net and synchronous models. The second objective is to apply validation technics for temporal verification, architectural constraints or ressource allocation of an EAST description at the different level of the EAST Process.

New standards are emerging in this field, such as AADL in avionics and AUTOSAR in automotive. They are strongly linked to UML representation models, as specific profiles. In both cases we are considering the connections with our MARTE profile effort, and the precise temporal semantics that we try to endow these formalisms with.

## 4.2. 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 translates 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 a 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.

## 4.3. 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-level synthesis and place-and-route techniques, SoC design adds yet new complexity dimensions to this process. Fully synchronous designs are not feasible anymore, and custom IP reuse becomes mandatory to integrate full processor cores into a new design. New approaches 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 approaches 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.

Here again new standards are emerging such as SPIRIT for the representation of SoC structure and behavior, both at low (RTL) and high (TLM) description level. Again we are considering these in connection with our modeling approach.

# 5. Software

## 5.1. SyncCharts/Esterel

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

**Participants:** Charles André, Robert de Simone, Dumitru Potop-Butucaru.

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 SyncCharts, mostly to validate new algorithmic model transformations or analysis.

We are reviewing experts for the IEEE standardization of ESTEREL. In the next future we should be able to contribute code processors to the commercial environnement by connecting into exchange format files.

## 5.2. K-passa

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

We developed this analysis software to characterize the feasible K-periodic as-soon-as-possible static schedules of a (strongly connected) latency-insensitive system. The software is written in Java, and uses the mascOpt library developed in the MASCOTTE team. This library in turn is based on the commercial solver CPlex, by Ilog, for linear constraint solving.

## 5.3. SynDEx

**Participants:** Christophe Gensoul, 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, PIC18F2680, 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, as for example in companies such as Robosoft, MBDA, and IFP.

## 5.4. SynDEx-IC

**Participants:** Mohamed Akil [Professor ESIEE Noisy-Le-Grand], Thierry Grandpierre, 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 automatically perform hardware/software partitioning.

# 6. New Results

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

**Participants:** Charles André, Benoît Ferrero, Frédéric Mallet, Marie-Agnès Péraldi-Frati, Robert de Simone.

We completed the design of the Time Model subprofile in the MARTE official OMG UML profile, which was released and adopted in its Initial Version in May 2007, and is currently undergoing First Revision in the ad-hoc OMG Finalization Task Force.

The Time Model allows annotations of modeling elements, both behavioral and structural, with timing information. Time can be discrete or continuous, physical or logical (using then user-defined “clocks” as abstract generators of instants). Logical time is our major tool for assigning independent time threads to application parts. Different clocks can be independent (asynchrony) or partially correlated (multi-clock design). The purpose of the methodology we promote becomes then to view spatial distribution as well as temporal scheduling as ways to better associate and adjust various time threads to a common world, more strictly synchronous. This is done under the constraints provided by the target architecture platform, or the user requirements. This involves the specification annotations of such constraints, as well as the explicit representation of allocation functions into the modeling framework. We thus also devised the Allocation subprofile of MARTE (here allocation means “*spatial mapping + temporal scheduling*”).

Concretely, we defined a fairly comprehensive set of clocks and timed events constraint relations, such as *is subclock of*, *is faster than*, *is k-periodic* (of period k), to name a few. K-periodic subclock patterns play a specific role in static scheduling, and we paid special care to their syntactic definition.

We presented the Time Model subprofile of UML MARTE at the Models'07 conference [24] and at the FDL'07 Forum [25].

As part of the SOFTWAREFACTORY and RNTL OPENEMBEDD projects we provided a full grammar for time expressions and constraints, named `Clock Constraint Specification Language`, written in ANTLR v2.7. It is downloadable from the OPENEMBEDD source forge, as an XMI specification which runs under the MARTE profile.

## 6.2. Formal modeling of latency-insensitive systems

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

This line of work was prompted by a number of questions asked by industrial partners, mostly Texas Istruments and ST Microelectronics. It is conducted mostly in the context of the CIM PACA project SYS2RTL. The purpose is to propose a *Latency-Insensitive* design approach to the issues of *Timing Closure* and GALS modeling.

In previous year we worked towards formal characterization of such systems as "Marked/Event Graphs with asap semantics, capacity-2 bounded places and expanded integer latencies", whose basic transitions represent the individual IP component blocks. This allows us to validate hardware implementations schemes, to prove structural and dynamical properties (such as liveness and safety), and to inherit powerful known results on static (k-periodic) scheduling of such systems.

### *6.2.1. Smooth static schedules*

In a continuing attempt at optimization of such systems, and nost notably of the number of possible *relay-Stations* and congestion-control signalization wires involved in the design, we studied the desired shapes of stationary phase, so as to define some where the instants of activity and non-activity were most uniformly spread. We call such schedulings *smooth*, and their definition relies on a specific class of infinite binary words, which we want to study further for its algebraic and analytic properties.

We already know that, given $p$ and $k$ fixed period and periodicity respectively, the set of smooth words consist of a single rotation orbit generated by a single primary word. We also know that given such a representative, there exists a single other word in the same class obtained by the substitution $10 \rightarrow 01$ in a single location. By iteration we can find another cyclic ordering of the $p$ smooth words in $k, p)$.

Then, we use smooth words to assign schedulings to IP nodes/MEG transitions across the system. The important thing to notice here is that the activity firing rule is thus not obtained by symbolic execution, so that lenghty simulation is not required. The question of whether this matches an effective such execution is still left open, and we are actively working on the topic in the context of the ongoing PhD thesis of Jean-Vivien Millo.

### *6.2.2. Of k-periodic schedulings and routings*

Marked-Event Graphs are free-choice, conflict-free formalisms for the simple reason that computed values always flow along the same routes. We want here to generalize this scheme by allowing alternative routes, but without loosing the determinism and confluence that stems from conflict-freeness. Generalizing behaviors to unrestricted Petri Nets would incur a loss of such properties. So we took instead inspiration from the general philosophy of Kahn network, where only internal non-determinism is allowed: a data value is expected on a single channel at a time, implying that choices between input guards, in which the first available data on a selection of channels is selected, are forbidden.

We proposed the introduction of two simple primitives `Merge` and `Select` in addition to the existing Marked/Event Graph transitions. The main asumption is that the two kinds of nodes perform switching (production or consumption on alternative channels) according to *k-periodic* patterns, using the same kind of infinite binary words for routing as we used previously for scheduling. One can show how augmented Marked/Event Graphs, which we call *Kahn-Event Graphs (KEGs)*, can indeed encode an abstract form of Kahn process networks, where abstraction here refers to data. One can then prove that another kind of abstraction, which this time abstracts the *token game* firing rule up to an hyperperiod, can be used to encode our KEGs into the SDF model of Edward Lee, thereby solving the problem of balanced production/consumption of tokens with SDF so-called *balance equations*. This provides an effective criterion for checking safety and liveness of our models under a given initial marking. We also relate our KEG model to the previous attempts at *Cyclostatic DataFlow (CDF)* modeling, which we discovered only recently.

With the help of two auxiliary operators `On/When` on infinite binary words we are able to state a number of useful identities about the permutation, expansion and factorization of networks of (pure) `Merge/Select` nodes. All these operations have a natural interpretation as transformations for sharing or unsharing of channels and buses, under static scheduling, in Networks-on-Chip. This seems very promising for a future design flow approach associating k-periodic schedules with k-periodic routings.

The results were presented in part at the FMGALS workshop held in Nice in June, and in several informal seminars [28], [36]. They form the topic of Julien Boucaron's PhD thesis, defended on December 14th [17].

## 6.3. Formal modeling of the AAA methodology

**Participants:** Dumitru Potop-Butucaru, Yves Sorel.

Recent evolutions in the classes of specifications and desired implementations of SynDEx resulted in a need for revisiting and improving the formal support of the AAA/SynDEx methodology defined by Sorel *et al* [8] with graphs, partial orders, and sequential machine composition.

In particular, we want to:

- Be able to define global correctness criteria relating the functional specification with its real-time implementation, and
- Better understand the hypotheses made by the SynDEx architecture models, and their relations with the actual implementation.

Our first objective here was to define a formal framework allowing us to represent complex implementation transformations involving changes in both the temporal structure of the model, and in its sets of events and components. Typical transformations we want to cover include temporal and structural refinement, real-time scheduling, distribution, and various optimizations.

Our first result in this direction is the modelling of the successive temporal transformations of the AAA/SynDEx methodology. The results presented during the Synchron'07 workshop, is done in a formal framework where:

- Temporal aspects are modeled using the tagged systems of Benveniste, Caillaud *et al.* [40].
- The changes in the sets of events and components are modelled using a very general notion of transformation, which relates variables and events of the source to those of the target models of the transformation.

This modelling experiment showed that the chosen modelling framework allows the representation of the complex AAA/SynDEx transformations, but at the same time showed that the manipulation of tagged systems is difficult because the tagged system formalism is trace-based.

We are currently investigating the use of dedicated hardware description languages to realize a complete operational definition of AAA/SynDEx architectures, for semantic and simulation purposes.

## 6.4. Globally asynchronous implementations of synchronous specifications

**Participants:** Dumitru Potop-Butucaru, Robert de Simone, Yves Sorel.

We have revisited some aspects of the Theory of Endochronous systems. This theory focuses on multiclock formalisms based on partially coupled, partially independent clocks. The goal is to recognize how input events can be grouped into sets that are mutually independently concurrent. We have continued our work on the deterministic globally asynchronous implementation of synchronous specifications. Our approach is that of the Theory of Endochronous systems. We consider multi-clock formalisms based on partially coupled, partially independent clocks. We encode signal absence and explicit idling execution of the synchronous model with actual absence of communication and absence of execution (de-activation) in the globally asynchronous implementation.

We seek to determine under which conditions the asynchronous implementation preserves the function of the synchronous program/core (as an I/O stream mapping) while allowing for elastic timing.

To explore the limits of the approach, we focused on the semantics-preserving execution of a single synchronous program/core in an asynchronous environment. Our contribution has been to characterize the class of synchronous programs/cores that produce deterministic implementations using a very general execution machine based on: (1) the chosen signal absence encoding and (2) an ASAP (as soon as possible) reaction triggering policy. The characterization is a form of *confluence* and *determinism*.

As a practical application we plan to extend the class of implementations supported by AAA/SynDEx, through the definition of new synchronization schemes and associated algorithms. The objective is to allow operations and communication lines to be inactive in certain logical instants (repetitions of the graph pattern of the algorithm) depending on the state and input data.

We presented our results at the EMSOFT 2007 conference in October in Salzburg [35]. This year EMSOFT 2007 was part of the ESWEEK federation of Events.

## 6.5. Metamodeling of SynDEx

**Participants:** Fadoi Lakhal, Djamel Louar, Yves Sorel.

We developed an eCore metamodel of SynDEx (under Eclipse EMF). This metamodel is based on the grammar of SynDEx but also is compliant with both metamodels of Polychrony and Scicos. In order to create models from the SynDEx metamodel a graphic editor was developed. This editor is a plugin TopCased under Eclipse which allows the user to specify a diagram for the application algorithm, a diagram for the architecture, and to define associations between elements of the application algorithm and elements of the architecture. We developed also a translator from the XMI file representing these models into a file (.sdx) using the SynDEx format. Then, SynDEx can be used as usual for optimized implementation, and automatic code generation.

This metamodel mainly developed in the OpenEmbedd project was used in the MemVatex and in the OpenDevFactory projects.

## 6.6. UML and AADL

**Participants:** Charles André, Su-Young Lee, Frédéric Mallet, Robert de Simone.

AADL (Architecture Analysis & Design Language) is a standard of the Society of Automotive Engineers (SAE). It is used to design and analyze the software and hardware architectures of embedded and real-time systems for performance-critical characteristics (e.g., end-to-end latency, schedulability, and reliability). AADL and MARTE have many similar features. MARTE was designed to be very general and is expected to be the basis for UML representation of AADL models. The adopted MARTE OMG specification provides guidelines in this direction. However, AADL provides specific communication schemes between tasks, that need to be represented in MARTE: AADL tasks may be periodic or aperiodic, and in the former case of harmonic or independent periods; communication between tasks may use event-data or pure-data ports (with events triggering the recipient task behavior, while pure data are only sampled and used as such

whenever the consuming tasks is otherwise activated). Representing all these kinds of communications (periodic vs. aperiodic, event-triggered vs. sampled data) in MARTE is not only a challenge, but also an opportunity to provide timed semantics inside the modeling framework (and not aside, as separately provided semantic interpretation to time attributes). We build this semantic construction using MARTE Time Model, which is intended exactly for this: specifying in a formal fashion new timed domains of computation and communication. To demonstrate this ability, we have defined with MARTE a model library for AADL that should be used as a black-box by end-users. Using MARTE to describe other model libraries for other environments related to automotive systems (like AutoSar/East-ADL2) could be an interesting follow-up of this action.

We presented our results at the FDL'07 Forum in Barcelona [23], and in the technical workshop ICECCS in New-Zealand [29].

## 6.7. MARTE and automotive standards

**Participants:** Fadoi Lakhal, Djamel Louar, Marie-Agnès Peraldi-Frati, Dumitru Potop-Butucaru, Yves Sorel.

EAST-ADL [44] and AUTOSAR [51] are two standards for electronic embedded system design in automotive. EAST-ADL (Embedded electronic Architecture STudy - Architecture Description Language) is both a design process and a language. The process adopts a decomposition of the design by abstraction levels. Each levels is domain oriented (control/command, software design, implementation). The EAST-ADL language provides model elements to describe the structure and the behaviour of application. AUTOSAR (AUTomotive Open System ARchitecture) is an open and standardized automotive software architecture. Autosar is clearly situated at the implementation level in a design process. The upper levels are those of EAST-ADL2.0[3]. Autosar and EAST-ADL2.0 have the same approach for system design, i.e. they consider an independent development for the hardware part and the software part of a system. These models cannot cover the expression of the temporal aspects of a system (deadline, duration, time stamping of events for time triggered function, timing characteristics of hardware ....). The model elements of EAST-ADL only consider time at the requirement level. Autosar inherits from the EAST-ADL process the design of its software components. The same issues of expressing the temporal aspects of components appear in Autosar. We use conjointly EAST-ADL2, AUTOSAR and MARTE for modelling temporal aspects of hardware and software. components. Behaviours of ADLFunction and runable entities are described with UML behaviours stereotyped by TimedProcessing. TimedEvent are associated with the beginning and the end of behaviour. As ADLFunctions can be hierarchical, multiple timing chains can be deduced from the description with an objective of performing a temporal analysis of the system. This work will be extended by applying techniques of partitioning, allocation of function onto the hardware using the AAA approach.

## 6.8. Modeling SPIRIT IP-XACT in UML MARTE

**Participants:** Charles André, Frédéric Mallet, Aamir Mehmood Khan, Robert de Simone.

Reuse and integration of heterogeneous Intellectual Property (IP) from multiple vendors is a major issue of System-on-Chip (SoC) design. There is a clear demand for a multi-level description of SoC, with verification, analysis and optimization possibly conducted at the various modeling levels. This requires interoperability of IP components described at the corresponding stages, and the use of traceability to switch between different abstraction layers. To reach this goal there is a increased use of standards such as OCSI SystemC, SPIRIT Consortium IP-XACT, Silicon Integration Initiative OpenAccess API, and also recent Unified Modeling Language (UML)-based standards like UML for SoC, UML for SystemC and the UML Profile for Modeling and Analysis of Real-Time and Embedded systems (MARTE) that specifically targets real-time and embedded systems.

System Modeling requires representation of both structural/architectural/platform aspects at different levels of abstraction and behavioral/functional aspects possibly considering timing viewpoints such as untimed/asynchronous/causal, logical synchronous/cycle-accurate or physical/timing models. For system structure representation, UML uses component and composite structure diagrams while SysML uses block diagrams. Tools like Esterel Studio, and virtual platforms like CoWare, Synopsys CoreAssembler and ARM RealView, introduce their own architecture diagrams. IP-XACT is a language-independent front-end that allows for the specification of IP meta-data and tool interfaces. It uses its own XML syntax to describe structure. For behavioral representation, SystemC provides programming libraries to represent IP component behavior at different abstraction levels, from Transaction Level Modeling (TLM) to RTL but it requires additional support for architecture modeling. IP-XACT relies on SystemC, VHDL, and Verilog HDL to specify the actual behavior of components. UML behavioral diagrams provide a support for describing "untimed" algorithms, while MARTE also provides support for logically or physically timed behaviors. UML can be tailored to a specific modeling domain thanks to the generic profiling extension mechanism.

We are currently using Model Driven Engineering (MDE) to transform UML models into IP-XACT models. We started by creating an ad-hoc IP-XACT UML profile. And we now study how a proper subset of MARTE can be used to add modeling capacity for timed behavior. The gain of such an approach would be the reuse of existing UML graphical tools (e.g., Eclipse, Magic Draw, Rational Software Architect, Papyrus), which have already been tried and tested by the software community, thus reducing the effort of creating new ones and providing interoperability based on UML interchange models.

A satellite workshop of the DATE'08 conference in Munich should be held on that topic.

## 6.9. Preemptive real-time scheduling in the monoprocessor case

**Participants:** Aguibou Barry, Liliana Cucu, Patrick Meumeu, Dumitru Potop-Butucaru, Yves Sorel.

Since last years, we have been studying the scheduling problem of dependent, strictly periodic, preemptive tasks (called later on operations) considering the exact cost of preemptions. Indeed, in the case of hard real-time embedded systems the designer must guarantee that all the deadlines of all the tasks are met, otherwise dramatic consequences may occur. Moreover, in the embedded system context resources must carefully be minimized. Therefore, it is necessary to take into account this exact cost rather than relying on its approximation in the Worst Case Execution Time (WCET) of each task, which possibly leads to a wrong real-time execution whereas the schedulability analysis concluded the system was schedulable, and certainly leads to a waste of resources due to the margins the designer must take.

In order to achieve this goal, as a first step we divided the set of all systems of operations into five sub-sets, and we showed that three of them were definitely non-schedulable due to relationships between their periods and their WCETs [9]. The two remaining sub-sets were potentially schedulable and corresponded respectively to the sub-set of operations with harmonic periods [1], and the sub-set of operations with non-harmonic periods.

For the first case, we gave a scheduling algorithm which led to a necessary and sufficient schedulability condition [34]. To achieve this, we used for each individual operation the time units still available in each of its instances, and the transitions between time units still available and those already executed. Each transition corresponds to a preemption. It is worth noticing that from each operation point of view, the time units already executed were due to the execution of the operations preceding it, relatively to the precedence constraints.

For the second case, which happened to be more complicated to handle because the previous approach could not work anymore, we decided to tackle first the simpler scheduling problem of hard real-time systems composed of independent periodic preemptive tasks where we assume that tasks are scheduled by using Liu & Layland's pioneering model, and according to the Rate Monotonic Analysis (RMA) to benefit from hints that we could extend to get a solution to our problem. Scheduling tasks according to RMA means we are in the fixed priority context, and the highest fixed priority is assigned to the task with the shortest period. When two tasks have the same period they are scheduled arbitrarily. The principle that we used is as follows.

[1] A sequence $(a_i)_{1\leq i\leq n}$ is harmonic if and only if there exists $q_i \in \mathbb{N}$ such that $a_{i+1} = q_i a_i$. Notice that we may have $q_{i+1} \neq q_i \quad \forall i \in \{\overline{1, \cdots, n}\}$.

We consider a set of $n$ independent periodic preemptive tasks $\tau_i, 1 \leq i \leq n$. Each task $\tau_i$ is an infinite sequence of instances $\tau_i^k$, $k \in \mathbb{N}^+$, and is characterized by a WCET $C_i$, not including the approximation of the cost of the preemption, a period $T_i$, and a release time relative to 0, $r_i$. We assume that all tasks are released simultaneously, and as this is repeated every hyperperiod $H$ it is sufficient to perform the schedulability analysis in the interval $[0, H]$ where $H$ is the least common multiple of the periods of the tasks. Now, since the worst case response time of a task may not occur in the first instance, we consider all instances of a task within a hyperperiod, and perform the schedulability analysis only within the first hyperperiod. All timing characteristics in our model are assumed to be non-negative integers, i.e. they are multiples of some elementary time interval (for example the "CPU tick") and $\alpha$ denotes the cost of one preemption for a given processor. Since all tasks except the one with the highest priority may be preempted, the execution time of a task may vary from one instance to another. We call *preempted execution time* (PET) the WCET augmented with the exact cost due to preemptions for each instance of a task within a hyperperiod. It depends on the instance, and on the number of preemptions occuring in that instance. From the point of view of task $\tau_i$, since it may only be preempted by higher priority tasks, we define the *hyperperiod at level i*, $H_i$, as the least common multiple of the set of tasks with a priority higher than or equal to task $\tau_i$. Hence, task $\tau_i$ is released $\sigma_i$ times in each hyperperiod at level $i$ starting from 0, where $\sigma_i$ is the ratio between $H_i$ and $T_i$. According to the number of preemptions $N_p(\tau_i^k)$ of task $\tau_i = (C_i, T_i)$ in each instance $\tau_i^k$, its PET $C_i^k$ may be different from one instance to another. Now, as task $\tau_i$ may only be preempted by the set of tasks with a priority higher than $\tau_i$, then there are exactly $\sigma_i$ different PETs for task $\tau_i$. In other words, from the point of view of any task $\tau_i$, there exists a function $\pi$, which maps its WCET $C_i$ into its respective PET $C_i^k$ in each instance $\tau_i^k$. These PETs are computed by using the time units still available in each instance $\tau_i^k$ and the transitions from time units still available to time units already executed because of the execution of a higher priority task. In addition, we compute the response time in each instance by adding the corresponding PET to the cardinal of time units already executed occuring before the completion of the PET in that instance. This process is repeated for each individual task in the system relatively to those with a higher priority. Therefore, at the end, we can define for a given task set the *exact total utilization factor* of the processor and therefore, deduce the exact cost due to preemptions incurred by the system. A necessary and sufficient schedulability condition for our system of $n$ tasks, all released at the same time and scheduled according to RMA **which takes the exact cost due to preemption** into account, is given by the *exact total utilization load factor* of the processor less than or equal to 1. These results are presented in [33]. We extended this result to the case of our scheduling problem which consists of operations with precedence and strict periodicity constraints and with periods forming a non-harmonic sequence.

Last year we have started introducing jitter constraints to our model, and the preliminary results we got were about non-schedulability conditions. As a first result, we showed that among the set of all systems of operations, those systems which comprise a couple of operations with co-prime periods are strongly not schedulable, that is to say, such a system can never be schedulable wathever the jitter is allowed or not. As a second result, we showed that systems of operations which comprise a couple of operations such that the least common divisor of their periods divises the elapsed-time between their respective start times are weakly not schedulable, that is to say, some systems could be schedulable whereas some others could not be.

Currently, we are introducing latencies contraints, and we are extending jitter constraints to other sub-cases of our model – systems with harmonic and non-harmonic periods – in order to make the latter more realistic.

## 6.10. Non-preemptive real-time scheduling in the multiprocessor case

**Participants:** Ouafae Aidi, Omar Kermia, Yves Sorel.

Last year we proposed a greedy heuristic for non-preemptive multiprocessor real-time scheduling of systems with precedence and strict periodicity constraints. It was presented this year in [30]. The comparison tests performed between this heuristic and an exact algorithm showed that, in term of execution speed, the proposed heuristic satisfied the rapid prototyping goal we have. However, in term of schedulability (either to find a schedule or decide whether the systems is not schedulable), the heuristic presented some weaknesses due to the schedulability condition it was based on. Indeed, this condition took under consideration only two

tasks at each time, and allowed us to schedule only tasks with the same or multiple periods onto the same processor, in order to improve the schedulability condition we modified the assignment algorithm inside the multiprocessor real-time scheduling heuristic we had proposed. We recall that the proposed heuristic is composed of three algorithms: assignment, unrolling, and finally, distribution and scheduling. Therefore, we performed a schedulability study for an arbitrary number of tasks, taking into account both periods and WCET of tasks. Consequently, the schedulability decision can be made immediately after the assignment algorithm rather than waiting for the complete execution of the heuristic. Moreover, we introduced back-tracking in the assigment algorithm to increase the possibilities to find solutions. Since back-tracking is performed only when a task can not be assigned, it does not increase significantly the complexity of the algorithm. Finally, we obtained an effective assignment algorithm which allowed the heuristic to produce better results than the previous version, and whose execution speed is acceptable.

Since the heuristic is based on the hyper-period, i.e. the least common multiple of all tasks periods, it may have a large value compared to the periods of the tasks. Consequently, some tasks are repeated much more times than other tasks, and in turn some processors need much more memory than others. This is why we proposed a load balancing, and efficient memory usage algorithm. This load-balancing is executed after the execution of the distribution and scheduling algorithm. It consists in redistributing tasks in order to minimize the memory use in the processors while reducing the total execution time.

Concerning the implementation of data dependences between periodic tasks onto a communication medium (bus, point to point link) we considered last year that the producer task must have a period equal to, or which divides the period of the consumer task, in order not to loose any data. Bibliographic studies brought out that the producer task can have a period greater (but necessarily multiple) than that of the consumer task. Therefore, we modified our distribution and scheduling heuristic, as well as the code generator in consequence. These improvements were exploited on a “visual control of autonomous CyCabs for platooning” application.

In order to increase the quality of the solutions that our fast greedy heuristic provides we performed a bibliographic study about metaheuristics. We chose the Genetic Algorithm metaheuristic type because these metaheuristics allow on the one hand double optimization that we need in our problem (satisfy several constraints, and minimize the total execution time), and on the other hand it does not need, to operate, an initial solution which is very difficult to determine in our case. We gave a preliminary version of this metaheuristic. We plan to program it in Ocaml, and to compare its performances relatively to the greedy one.

## 6.11. Automotive case study in UML MARTE

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

In the context of the ANR RNTL MemVatex project, we considered a specific automotive case study to support our model-driven design approach. It consists of a Knock control part in an ignition system, prevention backfiring in the cylinders. The modeling uses two distinct logical clocks, one being the computing processor cycle time, the other the engine revolution angle. The knock detection must be performed in real-time, which implies that computerized processing must complete in less than a 90° crankshaft angle. The modeling uses independent clocks; solving the constraints allows to bound the engine speed to establish correctness.

We modeled the problem in UML MARTE, examplifying our methodology. It was presented in the conference SIES [26] and ERTCS’07 [27].

## 6.12. CyCab experimentations

**Participants:** Omar Kermia, Quentin Quadrat, Yves Sorel.

Last years we developed with SynDEx the first version of a “visual control of autonomous CyCabs for platooning” application. We separately implemented the vision and the control algorithms onto the distributed architecture of the CyCab while satisfying real-time constraints. This year was devoted to execute both algorithms together. Due to the heterogeneous nature of the CyCab hardware architecture we had to solve numerous problems to execute properly in real-time the vision algorithm onto the embedded PC using the

Linux/RTAI executive kernel, and execute the control algorithm onto several MPC555 microcontrollers using the MPC555 executive kernel, all the processors communicating through a CAN bus. Presently, the application is working well. In addition, we improved the control algorithm with a Kalmann filter for the longitudinal control of the CyCab trajectory. Lateral control needs to be improved yet.

## 6.13. Improvement of SynDEx

**Participants:** Djamel Louar, Christophe Gensoul, Omar Kermia, 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-down 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 2007 we did a great deal of testing and debugging of the new flattening algorithm we had designed in 2006. We also implemented a long waited feature which is preventing from the creation of cyclic specification directly at design time in the graphical user interface (GUI). As we did this work using the new data structures we designed in 2006, we implemented a GUI for algorithm design which works upon these new structures, taking into account the remarks we had from our users. in order to provide requirements, as well as their corresponding solution models for the case study proposed by Siemens-VDO. This new GUI is intended to be more user friendly, providing clear error messages, using less windows through the use of a browser-like window (using too many windows was often noted as a weakness of the previous GUI) and implementing an "Undo" mecanism.

The multiperiod version of SynDEx has been pursued and is now the main branch of developement of the tool. It still requires some more testing and debugging.

All this work represent a lot of new code for the SynDEx tool which will be available soon for download as a new major version, SynDEx 7.0.

The SynDEx executive kernel for the RTOS RTAI (based on linux) developed last year for monoprocessor was extended to multiprocessor. Inter-processor communications are based on the TCP/IP protocol. Tests were performed for a hardware architecture composed of several Linux workstations.

# 7. Contracts and Grants with Industry

## 7.1. ST MIcroelectronics

**Participants:** Julien Boucaron, Robert de Simone.

This collaboration ended at the end of August. We provided a last year report on Latency-Insensitive Design from the point of view of synchronous reactive systems. But altogether, with changes in direction at our industrial partner, these results were rather overlooked at their place. Still, this collaboration accounted altogether for the financial and technical support of the PhD thesis of Julien Boucaron, in December 2007. Marc Benveniste, from ST Microelectronics, was member of the jury.

## 7.2. Texas Instruments

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

This collaboration takes the form of a series of one-year grants (4 so far). We explore Latency-Insensitive Design, based on the original work of Luca Carloni (now at Columbia University), with whom we share an associated-team programme supported by INRIA.

This year we worked on the extension of our modeling framework to encompass alternative routing and signal redirection. In the case of ultimately k-periodic schedules of LID systems, we also consider k-periodic switching schemes. The results are utterly interesting in that we prove that algebraic transformation hold, by which we can either share less channels with more interleaving and demanding scheduling, or progress communications faster with more channeling resources, where easier schedules may be feasible. The combination of scheduling time periodicity and routing space periodicity allows also to compute predictable buffering needs to accomodate the throughput. One can then hope in the future to devise techniques for re-allocating and re-routing, similar to nowadays retiming/recycling approach.

While theoretical results are rather satisfactory, the practical efficiency of the methods have to be further assessed. Particular care should be taken in the future as to the data representation formats for schedules and routes.

## 7.3. CARROLL CORTESS

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

CARROLL is a joint partnership between Thales, CEA, and INRIA. Its aim is to launch collaborative projects in the model-driven engineering technical area. The PROTES action, and its current CORTESS follow-up, were initiated by CARROLL to foster a specific OMG UML profile dedicated to Real-Time Embedded Modeling and Analysis, therefore named MARTE. We participate jointly with Espresso and DaRT INRIA teams.

As a result of these projects we defined MARTe *Request for Proposal (RFP)*, then put up a consortium PROMARTE involving many industrial, academic and vendor partners to submit an *Initial Proposal*, then a *Revised Proposal* balloted and voted at OMG. We are currently in the *Finalization Task Force* phase, where the different parts of the proposal are aligned and made fully compatible together; it will last for another year (from June 2007) before the *Final Version* is issued at OMG.

This work involved participation from Aoste members to several OMG Technical meetings in the US (Burlingame, St-Louis, Brussels,...), as well as internal progress meetings held in France, about every two months. Our results on time modeling presented in section 3.1.3 were included in the standard. We also produced, together with Thales and CEA-List, a full-length tutorial presentation.

## 7.4. MBDA

**Participants:** Christophe Gensoul, Yves Sorel.

MBDA completed this year the development with AAA/SynDEx of 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.

# 8. Other Grants and Activities

## 8.1. Regional collaborations

### *8.1.1. CIM PACA*

**Participants:** Jean-Vivien Millo, 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.

Inside this platform we are coordinating a dedicated project, named Spec2RTL, on methodological flows for high-level SoC synthesis. Participants are Texas Instruments, NXP, ST Microelectronics, Synopsys, Esterel Technologies as industrial partners, INRIA, I3S (CNRS/UNSA) and ENST on the academic side. A pool of PhD students are funded on a par basis between industrial partners and local PACA PhD grants under the BDI programme. There are currently 4 such students, one of them hosted by the Aoste team in conjunction with ST Microelectronics. Main research topic is LID design for GALS systems.

### *8.1.2. System@tic OpenDevFactory*

**Participants:** Charles André, Benoît Ferrero, Fadoi Lakhal, Djamel Louar, Frédéric Mallet, Robert de Simone, Yves Sorel.

We are taking part in the OPENDEVFACTORY subproject of the regional Ile-de-France SYSTEM@TIC Competitivity Cluster.

We have implemented the subprofiles "Time" and "Allocation" of the MARTE profile. This is part of the CT-RTE component developed in collaboration with CEA-List. This component is reused by most other components of the OPENDEVFACTORY platform. We also provided parsers for time expressions and clock constraints.

Another contribution in this project is to extend the applicability of the MARTE UML profile by implementation into a commercial or public-domain modeling tool, by connecting it with analysis and transformation tools (such as SYNDEX), and by applying it to industrial case studies (in relation with IFP mostly).

The problem proposed by IFP as case study in OpenDevFactory is currently being modeled in Scilab/Scicos. We use the new features of Scilab/Scicos allowing SynDEx code generation to obtain the algorithm graph. Then we specify in SynDEx the architecture, composed of several Linux workstations. This enables us to run the case study on this architecture. Since Linux is not actually a real-time operating system we use the Linux/RTAI executive kernel to obtain a distributed real-time implementation of the case study.

We developed a translator to transform the models using the XMI format, and are compliant with the SynDEx metamodel, into the "sdx" format of the SynDEx software. We also participated to the redaction of the document called "CasIFP-EvalComposant" describing the case study proposed by IFP. A part of the IFP case study was specified with Scicos and translated in SynDEx. Then, we generated a distributed real-time code for a multi-workstation using the new SynDEx Linux/RTAI executive kernel.

## 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 wish to draw closer ties with ALCHEMY and PROVAL on the topic of synchronous and $N$-synchronous modeling, in relation to code distribution and parallel execution.

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 OpenEmbeDD*

**Participants:** Charles André, Benoît Ferrero, Fadoi Lakhal, Robert de Simone, Yves Sorel.

This is a large platform project aimed at connecting several formalisms with model-driven engineering tools, in the embedded domain. The project partners are: INRIA, CEA-List, Thales, Airbus, France Telecom, CS, LAAS, and VERIMAG. Four INRIA teams are involved (ATLAS, Triskell, Aoste and DaRT).

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 to this in several directions: first, we provide the definition and implementation of the MARTE profile, as described in 3.1.3; second, we contribute our work on compilation-by-transformation of synchronous program; last, we develop the meta-model and profile for the AAA-SynDEx methodology, and the transformation needed to couple the tools.

We improved the SynDEx metamodel to be compliant with the Polychrony and Scicos metamodels, and thus to allow model transformations in order to translate a Polychrony program, or a Scicos diagram into an application algorithm of SynDEx. Based on this metamodel we developed an editor in the TopCased environment proposed by Airbus.

The various partner contributions in this project are assembled together by a dedicated engineer team of two people located at IRISA, as part of an INRIA forge.

### *8.2.3. RNTL project MemVatex*

**Participants:** SuYoung Lee, Djamel Louar, Fadoi Lakhal, Marie-Agnès Peraldi-Frati, Dumitru Potop-Butucaru, Yves Sorel.

The partners in this project are: Siemens-VDO, INRIA, CEA-List, CNRS-UTC, and Embelec.

The focus is on the tracability and the validation of requirements in a methodology for automotive applications. This methodology has to be defined in the project and is based on the new standards EAST-ADL2 and Autosar. Both of them put UML and SysML as central formalisms in the design flow. The project is currently in the "homogeneous" phase centered arround UML formalisms. Inria contributes in the definition of the methodology by introducing a formal integration of temporal and architectural characteristics. This project provides an interesting and complex industrial case study for demonstrating the MARTE results on multiclock representation, as well as our UML patterns for hardware software architecture. In a second phase we will target synchronous formalisms such as ESTEREL and SYNDEX, in order to apply the associated validation techniques and tools.

We extracted requirements as well as interesting design features from the specific case study proposed by Siemens-VDO. We expressed them using the allocation and architectural aspects of EAST-ADL2 and UML MARTE profiles. We reported on this in the technical delivrables JSP1T3-1b, JSP1T3-1c, and JSP1T3-1d of the project.

## 8.3. European collaborations

### *8.3.1. IST Network of Excellence ARTIST2*

**Participants:** Julien Boucaron, Robert de Simone, Frédéric Mallet, Dumitru Potop-Butucaru, 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.

Frédéric Mallet attended an international Symposium held in Vienna on automotive modeling (“Beyond AUTOSAR”) in this context.

## 8.4. Research exchange visits

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

This year was the second of our team-associationship with Columbia University, named HIDES. In the framework of this collaboration we had several exchange visits. Julien Boucaron and Jean-Vivien Millo visited Columbia for a week in June. Stephen Edwards and Olivier Tardieu visited us at Sophia-Antipolis in April for a month, and in Olivier’s case for another two weeks in July. Robert de Simone visited Columbia again in late October for a week.

As an important outcome of this collaboration, a book on Esterel compilation is in press [11].

# 9. Dissemination

## 9.1. Leadership within scientific community

Robert de Simone defended his Habilitation Thesis in October [18]. Robert de Simone was Program Chair for the IEEE/ACM conference MemoCode'07, held in Nice in May; he was also Program Chair for the FMGALS'07 satellite workshop. He is still member of the *Commission de Spécialiste UNSA 27$^e$ section*, and INRIA representative to the CIM PACA regional initiative on Microelectronics design; this includes being appointed to the Strategic Committee of the ARCSIS 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. He served as Program Committee member for MSR'2007. He is member of the International Advisory Board for the CRC Press on Embedded Systems. He was reviewer for the PhD thesis of J. Ouy (IRISA).

Charles André is member of the *Commission de Spécialiste UNSA 61$^e$ section*. He was Program Chair of IES'2006, the First international Symposium of Industrial Embedded Systems, held in Antibes.

Yves Sorel is Program Member for the following conferences and workshops: DASIP, ERTS, EUSIPCO, GRESTSI, JEAI, SYMPA. 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 PhD jurys: F. Bimbard, Hui Xue Shao.

## 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/Sophia-Antipolis (UNSA), for approximately 15h.

Yves Sorel teaches at ESIEE (an Engineering School located in Noisy-le-Grand), in the SETI Research Master 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. He provided didactic software as teaching material for basic architecture simulation.

Marie-Agnès Peraldi-Frati gives courses at different cursus levels of UNSA: A course and labs on real-time distributed systems in the STIC reaserch master (Embedded systems) and the STIC professionnal master (STREAM01/EPU), differents courses (Programming, Web developpment, Computer architecture) at the L1 level of the IUT Informatique. She is responsible for the option "Informatique embarquée et réseau sans fil" of the LPSIL cursus. She is member of the CERTEC (conseil d'études et de la recherche technologique) of the IUT of NICE.

Frédéric Mallet is Associate Professor at the University of Nice-Sophia Antipolis, department of Informatics. 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 and graduate students. He is on leave at INRIA starting from October.

# 10. Bibliography

## Major publications by the team in recent years

[1] C. ANDRÉ. *L'approche synchrone pour le développement des systèmes temps réel - Chapitre 5*, in "Encyclopédie de l'informatique et des systèmes d'information. Tome 1: La dimension technologique des systèmes d'information. Section 7: Systèmes temps réel", Vuibert, France, 2006, p. 151–166.

[2] 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.

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

[4] L. CUCU, Y. SOREL. *Non-preemptive multiprocessor scheduling for strict periodic systems with precedence constraints*, in "Proceedings of 23rd Annual Workshop of the UK Planning and Scheduling Special Interest Group, PLANSIG'04, Cork, Ireland", December 2004.

[5] 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.

[6] C. DIMA, A. GIRAULT, Y. SOREL. *Static fault-tolerant real-time scheduling with "pseudo-topological" orders*, in "Proceedings of Joint Conference on Formal Modelling and Analysis of Timed Systems and Formal Techniques in Real-Time and Fault Tolerant System, FORMATS-FTRTFT'04", LNCS, vol. 3253, Springer-Verlag, 2004.

[7] A. GIRAULT, H. KALLA, Y. SOREL. *A Scheduling Heuristics for Distributed Real-Time Embedded Systems Tolerant to Processor and Communication Media Failures*, in "International Journal of Production Research", vol. 42, n$^{o}$ 14, July 2004, p. 2877–2898.

[8] 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.

[9] P. MEUMEU YOMSI, Y. SOREL. *Non-Schedulability Conditions for Off-line Scheduling of Real-Time Systems Subject to Precedence and Strict Periodicity Constraints*, in "Proceedings of 11th IEEE International Conference on Emerging technologies and Factory Automation, ETFA'06, WIP, Prague, Czech Republic", September 2006, http://www-rocq.inria.fr/syndex/pub/etfa06/etfa06.pdf.

[10] D. POTOP-BUTUCARU, R. DE SIMONE. *Optimizations for Faster Execution of Esterel Programs*, in "MEMOCODE'03", 2003.

[11] D. POTOP-BUTUCARU, S. EDWARDS, G. BERRY. *Compiling Esterel*, to appear, Springer, 2006.

[12] Y. SOREL. *From modeling/simulation with Scilab/Scicos to optimized distributed embedded real-time implementation with SynDEx*, in "Proceedings of the International Workshop On Scilab and Open Source Software Engineering, SOSSE'05, Wuhan, China", October 2005.

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

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

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

## Year Publications

### Books and Monographs

[16] D. POTOP-BUTUCARU, S. EDWARDS, G. BERRY. *Compiling Esterel*, Springer, 2007.

### Doctoral dissertations and Habilitation theses

[17] J. BOUCARON. *Modélisation formelle de systèmes Insensibles à la Latence et ordonnancement*, Ph. D. Thesis, Université de Nice/Sophia-Antipolis, 2007.

[18] R. DE SIMONE. *Modélisation Synchrone dans la Conception de Systèmes Embarqués*, Habilitation à Diriger des Recherches, Université de Nice/Sophia-Antipolis, 2007.

### Articles in refereed journals and book chapters

[19] J. BOUCARON, J.-V. MILLO, R. DE SIMONE. *Formal Methods for Scheduling of Latency-Insensitive Designs*, in "Eurasip Journal on Embedded Systems", 2007, http://www.hindawi.com/GetArticle.aspx?doi=10.1155/2007/39161.

[20] L. CUCU, N. PERNET, Y. SOREL. *Periodic real-time scheduling: from deadline-based model to latency-based model*, in "Annals of Operations Research", 2007, http://www-rocq.inria.fr/syndex/pub/aor07/aor07.pdf.

[21] D. POTOP-BUTUCARU, B. CAILLAUD. *Correct-by-construction asynchronous implementation of modular synchronous specifications*, in "Fundamenta Informaticae", 2007.

### Publications in Conferences and Workshops

[22] A. ALBINET, J.-L. BOULANGER, H. DUBOIS, M.-A. PERALDI-FRATI, Y. SOREL, Q.-D. VAN. *Model-Based Methodology for Requirements Traceability in Embedded Systems*, in "Proceedings of 3rd European Conference on Model Driven Architecture® Foundations and Applications, ECMDA'07, Haifa, Israel", June 2007, http://www-rocq.inria.fr/syndex/pub/ecmda07/ecmda07.pdf.

[23] C. ANDRÉ, F. MALLET, R. DE SIMONE. *Modeling of immediate vs. delayed data communications: from AADL to UML MARTE*, in "FDL'07 Forum on specification and Design Languages, Barcelona, Spain", 6 pages, Sept. 2007.

[24] C. ANDRÉ, F. MALLET, R. DE SIMONE. *Modeling Time(s)*, in "MODELS'2007 10th Intern. Conf on Model Driven Engineering Languages and Systems, Nashville-TN, USA", ACM IEEE (editor), 15 pages, Sep-Oct 2007.

[25] C. ANDRÉ, F. MALLET, R. DE SIMONE. *Time Modeling in MARTE*, in "FDL'07 Forum on specification and Design Languages, Barcelona, Spain", 6 pages, Sept. 2007.

[26] C. ANDRÉ, F. MALLET, M.-A. PERALDI-FRATI. *A multiform time approach to real-time system modeling: Application to an automotive system*, in "Int. Symp. on Industrial Embedded Systems, Lisboa, Portugal", UNIVERSIDADE NOVA DE LISBOA (editor), IEEE, July 2007, p. 234–241.

[27] C. ANDRÉ, F. MALLET, M.-A. PERALDI-FRATI. *Multiform time in UML for real-time embedded applications*, in "IEEE International Conference on Embedded and Real-Time Computing Systems and Applications, Daegu, Korea", Aug. 2007, 6 p..

[28] J. BOUCARON, J.-V. MILLO. *Compositionality of Statically Scheduled IP*, in "Proceedings Third International Workshop on Formal Methods for Globally Asynchronous Locally Synchronous Design (FMGALS'2007)", EATCS electronic publishing, 2007.

[29] M. FAUGÈRE, T. BOURBEAU, R. DE SIMONE, S. GÉRARD. *MARTE: Also an UML Profile for Modeling AADL Applications*, in "ICECCS", 2007, p. 359–364.

[30] O. KERMIA, Y. SOREL. *A Rapid Heuristic for Scheduling Non-Preemptive Dependent Periodic Tasks onto Multiprocessor*, in "Proceedings of ISCA 20th international conference on Parallel and Distributed Computing Systems, PDCS'07, Las Vegas, Nevada, USA", Sept. 2007, http://www-rocq.inria.fr/syndex/pub/pdcs07/pdcs07.pdf.

[31] F. LAGARDE, F. TERRIER, C. ANDRÉ, S. GÉRARD. *Constraints modeling for (profiled) UML models*, in "European Conference on Model-Driven Architecture: Foundations and Applications 2007 (ECMDA 07), Haïfa, Israel", Lecture Notes in Computer Science, vol. 4530, June 2007, p. 130–143.

[32] F. LAGARDE, F. TERRIER, C. ANDRÉ, S. GÉRARD. *Extending OCL to ensure model transformations*, in "Foundations and Practices of UML, Auckland, New Zealand", Lecture Notes in Computer Science, workshop of ER 2007, vol. 4802, November 2007.

[33] P. MEUMEU YOMSI, Y. SOREL. *Extending Rate Monotonic Analysis with Exact Cost of Preemptions for Hard Real-Time Systems*, in "Proceedings of 19th Euromicro Conference on Real-Time Systems, ECRTS'07, Pisa, Italy", Jul. 2007, http://www-rocq.inria.fr/syndex/pub/ecrts07/ecrts07.pdf.

[34] P. MEUMEU YOMSI, Y. SOREL. *Schedulability Analysis with Exact Number of Preemptions and No Idle Time for Real-Time Systems with Precedence and Strict Periodicity Constraints*, in "Proceedings of 15th International Conference on Real-Time and Network Systems, RTNS'07, Nancy, France", March 2007, http://www-rocq.inria.fr/syndex/pub/rtns07/rtns07.pdf.

[35] D. POTOP-BUTUCARU, Y. SOREL, R. DE SIMONE. *Necessary and Sufficient Conditions for Deterministic Desynchronization*, in "Proceedings of Conference on Embedded Systems Software, EMSOFT'07, Salzburg, Austria", Oct. 2007, http://www-rocq.inria.fr/syndex/pub/emsoft07/emsoft07.pdf.

### Internal Reports

[36] J. BOUCARON, B. FERRERO, J.-V. MILLO, R. DE SIMONE. *Statically Scheduled Process networks*, Research Report, n$^{O}$ RR-6152, INRIA, 2007, http://hal.inria.fr/inria-00168757/fr/.

## References in notes

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

[38] F. BACCELLI, G. COHEN, GEERT JAN. OLSDER, J.-P. QUADRAT. *Synchronization and Linearity: an algebra for discrete event systems*, John Wiley & Sons, 1992, http://www-rocq.inria.fr/metalau/cohen/SED/book-online.html.

[39] 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.

[40] A. BENVENISTE, L. CARLONI, P. CASPI, A. SANGIOVANNI-VINCENTELLI. *Heterogeneous Reactive Systems Modeling and Correct-by-Construction Deployment*, in "Proceedings EMSOFT'03", 2003.

[41] F. BOUSSINOT, R. DE SIMONE. *The Esterel Language*, in "Proceedings of the IEEE", September 1991.

[42] J. CARLIER, P. CHRÉTIENNE. *Problèmes d'ordonnancement*, Masson, 1988.

[43] L. CARLONI, K. MCMILLAN, A. SANGIOVANNI-VINCENTELLI. *Theory of Latency-Insensitive Design*, in "IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems", 2001.

[44] V. DEBRUYNE, F. SIMONOT-LION, Y. TRINQUET. *EAST-ADL an Architecture Description Language, Validation and Verification Aspects*, in "IPIP World Computer Congress, Workshop on Architecture Description Language, Toulouse", 2004.

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

[46] 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", 1998.

[47] 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.

[48] S. EDWARDS. *Languages for Digital Embedded Systems*, Kluwer, 2000.

[49] 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", 1999.

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

[51] H. HEINECKE. *AUTOSAR, an industrywide initiative to manage the complexity of emerging Automotive E/E-Architecture*, in "Electronic Systems for Vehicles 2003, VDI Congress, Baden-Baden", 2003.

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

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

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

[55] 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.

[56] 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.

[57] 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.

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