Activity Report 2016

Project-Team TEA

Time, Events and Architectures

RESEARCH CENTER
Rennes - Bretagne-Atlantique

THEME
Embedded and Real-time Systems

Table of contents

1. Members **2**
2. Overall Objectives **2**
2.1. Introduction 2
2.2. Context 3
2.3. Motivations 3
2.4. Challenges 4
3. Research Program **4**
3.1. Previous Works 4
3.2. Modeling Times 5
3.3. Modeling Architectures 6
3.4. Scheduling Theory 6
3.5. Virtual Prototyping 7
4. Application Domains **8**
4.1. Automotive and Avionics 8
4.2. Factory Automation 9
5. Highlights of the Year **9**
6. New Software and Platforms **9**
6.1. ADFG: Affine data-flow graphs scheduler synthesis 9
6.2. The Eclipse project POP 10
6.3. The Polychrony toolset 11
7. New Results **12**
7.1. Toward a distribution of ADFG 12
7.2. Modular verification of cyber-physical systems using contract theory 13
7.3. Runtime verification and trace analysis 14
7.4. Polychronous automata and formal validation of AADL models 14
7.5. Formal Semantics of Behavior Specifications in the Architecture Analysis and Design Language Standard 15
7.6. Integration of Polychrony with QGen 16
7.7. Code generation for poly-endochronous processes 16
8. Bilateral Contracts and Grants with Industry **17**
8.1. Bilateral Contracts with Industry 17
8.2. Bilateral Grants with Industry 17
9. Partnerships and Cooperations **18**
9.1. National Initiatives 18
9.1.1. ANR 18
9.1.2. PAI CORAC 18
9.2. International Initiatives 18
9.2.1. International Project Grants 18
9.2.1.1. US Air Force Office for Scientific Research – Grant FA8655-13-1-3049 18
9.2.1.2. Applied Science & Technology Research Institute (ASTRI, Hong Kong) 19
9.2.2. Inria International Labs 19
9.2.3. Inria International Partners 19
9.2.3.1. POLYCORE 19
9.2.3.2. VESA 20
9.2.3.3. TIX 20
9.3. International Research Visitors 20
9.3.1. Visits of International Scientists 20
9.3.2. Visits to International Teams 21
10. Dissemination **21**

10.1. Promoting Scientific Activities 21
10.1.1. Scientific events organisation 21
10.1.1.1. General chair, scientific chair 21
10.1.1.2. Member of the organizing committees 21
10.1.2. Scientific events selection 21
10.1.3. Journal 22
10.1.3.1. Member of the editorial boards 22
10.1.3.2. reviewer 22
10.2. Teaching - Supervision - Juries 22
10.2.1. Invited talks 22
10.2.2. Supervision 22
10.2.3. Juries 22
11. Bibliography **22**

Project-Team TEA

Creation of the Team: 2014 January 01, updated into Project-Team: 2015 January 01

Keywords:

Computer Science and Digital Science:

1.2. - Networks
1.2.7. - Cyber-physical systems
1.2.8. - Network security
1.5. - Complex systems
1.5.1. - Systems of systems
1.5.2. - Communicating systems
2.1. - Programming Languages
2.1.1. - Semantics of programming languages
2.1.6. - Concurrent programming
2.1.8. - Synchronous languages
2.1.10. - Domain-specific languages
2.2. - Compilation
2.2.1. - Static analysis
2.2.3. - Run-time systems
2.3. - Embedded and cyber-physical systems
2.3.1. - Embedded systems
2.3.2. - Cyber-physical systems
2.3.3. - Real-time systems
2.4. - Verification, reliability, certification
2.4.1. - Analysis
2.4.2. - Model-checking
2.4.3. - Proofs
2.5. - Software engineering
4.4. - Security of equipment and software
4.5. - Formal methods for security
4.7. - Access control
5.7.2. - Music
6.1.1. - Continuous Modeling (PDE, ODE)
6.1.3. - Discrete Modeling (multi-agent, people centered)
6.2.1. - Numerical analysis of PDE and ODE
6.2.5. - Numerical Linear Algebra
6.2.6. - Optimization
7.4. - Logic in Computer Science
7.6. - Computer Algebra

Other Research Topics and Application Domains:

5.1. - Factory of the future
5.2. - Design and manufacturing

6.1.1. - Software engineering
6.4. - Internet of things
6.6. - Embedded systems
9.2.1. - Music, sound
9.4.1. - Computer science

1. Members

Research Scientists

Jean-Pierre Talpin [Inria, Team leader, Senior Researcher, HDR]
Thierry Gautier [Inria, Senior Researcher]
Vania Joloboff [Inria, Senior Researcher]

Engineers

Loïc Besnard [CNRS SED, Senior Research Engineer, seconded at 60%]
Clément Guy [Inria, until Aug 2016]
Alexandre Honorat [Inria]
Hai Nam Tran [Inria, from Dec 2016]

PhD Student

Simon Lunel [Mitsubishi Electrics R&D, granted by CIFRE]

Visiting Scientists

Rajesh Gupta [UC San Diego, Jul 2016]
Brian Larson [FDA/KSU, Jul 2016]

Administrative Assistants

Stéphanie Lemaile [Inria]
Armelle Mozziconacci [CNRS, from May 2016]

Other

Daian Yue [Inria, Master Intern, from Feb 2016 until Jun 2016]

2. Overall Objectives

2.1. Introduction

An embedded architecture is an artifact of heterogeneous constituents and at the crossing of several design viewpoints: software, embedded in hardware, interfaced with the physical world. Time takes different forms when observed from each of these viewpoints: continuous or discrete, event-based or time-triggered. Unfortunately, modeling and programming formalisms that represent software, hardware and physics significantly alter this perception of time. Moreover, time reasoning in system design is usually isolated to a specific design problem: simulation, profiling, performance, scheduling, parallelization, simulation. The aim of project-team TEA is to define conceptually unified frameworks for reasoning on composition and integration in cyber-physical system design, and to put this reasoning to practice by revisiting analysis and synthesis issues in real-time system design with soundness and compositionality gained from formalization.

2.2. Context

In the construction of complex systems, information technology (IT) has become a central force of revolutionary changes, driven by the exponential increase of computational power. In the field of telecommunication, IT provides the necessary basis for systems of networked distributed applications. In the field of control engineering, IT provides the necessary basis for embedded control applications. The combination of telecommunication and embedded systems into networked embedded systems opens up a new range of systems, capable of providing more intelligent functionality thank to information and communication (ICT). Networked embedded systems have revolutionized several application domains: energy networks, industrial automation and transport systems.

20th-century science and technology brought us effective methods and tools for designing both computational and physical systems. But the design of cyber-physical systems (CPS) is much more than the union of those two fields. Traditionally, information scientists only have a hazy notion of requirements imposed by the physical environment of computers. Similarly, mechanical, civil, and chemical engineers view computers strictly as devices executing algorithms. To the extent we have designed CPS, we have done so in an ad hoc, on-off manner that is not repeatable. A new science of CPS design will allow us to create new machines with complex dynamics and high reliability, to apply its principles to new industries and applications in a reliable and economically efficient way. Progress requires nothing less than the construction of a new science and technology foundation for CPS that is simultaneously physical and computational.

2.3. Motivations

Beyond the buzzword, a CPS is an ubiquitous object of our everyday life. CPSs have evolved from individual independent units (e.g an ABS brake) to more and more integrated networks of units, which may be aggregated into larger components or sub-systems. For example, a transportation monitoring network aggregates monitored stations and trains through a large scale distributed system with relatively high latency. Each individual train is being controlled by a train control network, each car in the train has its own real-time bus to control embedded devices. More and more, CPSs are mixing real-time low latency technology with higher latency distributed computing technology.

In the past 15 years, CPS development has moved towards Model Driven Engineering (MDE). With MDE methodology, first all requirements are gathered together with use cases, then a model of the system is built (sometimes several models) that satisfy the requirements. There are several modeling formalisms that have appeared in the past ten years with more or less success. The most successful are the *executable* models [1] [2] [3], i.e., models that can be simulated, exercised, tested and validated. This approach can be used for both software and hardware.

A common feature found in CPSs is the ever presence of concurrency and parallelism in models. Large systems are increasingly mixing both types of concurrency. They are structured hierarchically and comprise multiple synchronous devices connected by buses or networks that communicate asynchronously. This led to the advent of so-called GALS (Globally Asynchronous, Locally Synchronous) models, or PALS (Physically Asynchronous, Logically Synchronous) systems, where reactive synchronous objects are communicating asynchronously. Still, these infrastructures, together with their programming models, share some fundamental concerns: parallelism and concurrency synchronization, determinism and functional correctness, scheduling optimality and calculation time predictability.

Additionally, CPSs monitor and control real-world processes, the dynamics of which are usually governed by physical laws. These laws are expressed by physicists as mathematical equations and formulas. Discrete CPS models cannot ignore these dynamics, but whereas the equations express the continuous behavior usually using real numbers (irrational) variables, the models usually have to work with discrete time and approximate floating point variables.

[1] *Matlab/Simulink*, https://fr.mathworks.com/products/simulink.html
[2] *Ptolemy*, http://ptolemy.eecs.berkeley.edu
[3] *SysML*, http://www.uml-sysml.org

2.4. Challenges

A cyber-physical, or reactive, or embedded system is the integration of heterogeneous components originating from several design viewpoints: reactive software, some of which is embedded in hardware, interfaced with the physical environment through mechanical parts. Time takes different forms when observed from each of these viewpoints: it is discrete and event-based in software, discrete and time-triggered in hardware, continuous in mechanics or physics. Design of CPS often benefits from concepts of multiform and logical time(s) for their natural description. High-level formalisms used to model software, hardware and physics additionally alter this perception of time quite significantly.

In model-based system design, time is usually abstracted to serve the purpose of one of many design tasks: verification, simulation, profiling, performance analysis, scheduling analysis, parallelization, distribution, or virtual prototyping. For example in non-real-time commodity software, timing abstraction such as number of instructions and algorithmic complexity is sufficient: software will run the same on different machines, except slower or faster. Alternatively, in cyber-physical systems, multiple recurring instances of meaningful events may create as many dedicated logical clocks, on which to ground modeling and design practices.

Time abstraction increases efficiency in event-driven simulation or execution (i.e SystemC simulation models try to abstract time, from cycle-accurate to approximate-time, and to loosely-time), while attempting to retain functionality, but without any actual guarantee of valid accuracy (responsibility is left to the model designer). Functional determinism (a.k.a. conflict-freeness in Petri Nets, monotonicity in Kahn PNs, confluence in Milner's CCS, latency-insensitivity and elasticity in circuit design) allows for reducing to some amount the problem to that of many schedules of a single self-timed behavior, and time in many systems studies is partitioned into models of computation and communication (MoCCs). Multiple, multiform time(s) raises the question of combination, abstraction or refinement between distinct time bases. The question of combining continuous time with discrete logical time calls for proper discretization in simulation and implementation. While timed reasoning takes multiple forms, there is no unified foundation to reasoning about multi-form time in system design.

The objective of project-team TEA is henceforth to define formal models for timed quantitative reasoning, composition, and integration in embedded system design. Formal time models and calculi should allow us to revisit common domain problems in real-time system design, such as time predictability and determinism, memory resources predictability, real-time scheduling, mixed-criticality and power management; yet from the perspective gained from inter-domain timed and quantitative abstraction or refinement relations. A regained focus on fundamentals will allow to deliver better tooled methodologies for virtual prototyping and integration of embedded architectures.

3. Research Program

3.1. Previous Works

The challenges of team TEA support the claim that sound Cyber-Physical System design (including embedded, reactive, and concurrent systems altogether) should consider multi-form time models as a central aspect. In this aim, architectural specifications found in software engineering are a natural focal point to start from. Architecture descriptions organize a system model into manageable components, establish clear interfaces between them, collect domain-specific constraints and properties to help correct integration of components during system design. The definition of a formal design methodology to support heterogeneous or multi-form models of time in architecture descriptions demands the elaboration of sound mathematical foundations and the development of formal calculi and methods to instrument them. This constitutes the research program of team TEA.

System design based on the “synchronous paradigm” has focused the attention of many academic and industrial actors on abstracting non-functional implementation details from system design. This elegant design abstraction focuses on the logic of interaction in reactive programs rather than their timed behavior, allowing to secure functional correctness while remaining an intuitive programming model for embedded systems. Yet, it corresponds to embedded technologies of single cores and synchronous buses from the 90s, and may hardly cover the semantic diversity of distribution, parallelism, heterogeneity, of cyber-physical systems found in 21st century Internet-connected, true-time^{TM}-synchronized clouds, of tomorrow’s grids.

By contrast with a synchronous hypothesis yet from the same era, the polychronous MoCC implemented in the data-flow specification language Signal, available in the Eclipse project POP [4] and in the CCSL standard. [5], are inherently capable of describing multi-clock abstractions of GALS systems. The POP and TimeSquare projects provide tooled infrastructures to refine high-level specifications into real-time streaming applications or locally synchronous and globally asynchronous systems, through a series of model analysis, verification, and synthesis services. These tool-supported refinement and transformation techniques can assist the system engineer from the earliest design stages of requirement specification to the latest stages of synthesis, scheduling and deployment. These characteristics make polychrony much closer to the required semantic for compositional, refinement-based, architecture-driven, system design.

While polychrony was a step ahead of the traditional synchronous hypothesis, CCSL is a leap forward from synchrony and polychrony. The essence of CCSL is “multi-form time” toward addressing all of the domain-specific physical, electronic and logical aspects of cyber-physical system design.

3.2. Modeling Times

To make a sense and eventually formalize the semantics of time in system design, we should most certainly rely on algebraic representations of time found in previous works and introduce the paradigm of "time systems" (type systems to represent time) in a way reminiscent to CCSL. Just as a type system abstracts data carried along operations in a program, a time system abstracts the causal interaction of that program module or hardware element with its environment, its pre and post conditions, its assumptions and guarantees, either logical or numerical, discrete or continuous. Some fundamental concepts of the time systems we envision are present in the clock calculi found in data-flow synchronous languages like Signal or Lustre, yet bound to a particular model of concurrency, hence time.

In particular, the principle of refinement type systems [6], is to associate information (data-types) inferred from programs and models with properties pertaining, for instance, to the algebraic domain on their value, or any algebraic property related to its computation: effect, memory usage, pre-post condition, value-range, cost, speed, time, temporal logic [7]. Being grounded on type and domain theories, a time system should naturally be equipped with program analysis techniques based on type inference (for data-type inference) or abstract interpretation (for program properties inference) to help establish formal relations between heterogeneous component “types”. Just as a time calculus may formally abstract timed concurrent behaviors of system components, timed relations (abstraction and refinement) represent interaction among components.

Scalability and compositionality requires the use of assume-guarantee reasoning to represent them, and to facilitate composition by behavioral sub-typing, in the spirit of the (static) contract-based formalism proposed by Passerone et al. [8]. Verification problems encompassing heterogeneously timed specifications are common and of great variety: checking correctness between abstract and concrete time models relates to desynchronisation (from synchrony to asynchrony) and scheduling analysis (from synchrony to hardware). More generally, they can be perceived from heterogeneous timing viewpoints (e.g. mapping a synchronous-time software on a real-time middle-ware or hardware).

[4] *Polychrony on Polarsys*, https://www.polarsys.org/projects/polarsys.pop
[5] *Clock Constraints in UML/MARTE CCSL*. C. André, F. Mallet. RR-6540. Inria, 2008. http://hal.inria.fr/inria-00280941
[6] *Abstract Refinement Types*. N. Vazou, P. Rondon, and R. Jhala. European Symposium on Programming. Springer, 2013.
[7] *LTL types FRP*. A. Jeffrey. Programming Languages meets Program Verification.
[8] *A contract-based formalism for the specification of heterogeneous systems*. L. Benvenistu, et al. FDL, 2008

This perspective demands capabilities not only to inject time models one into the other (by abstract interpretation, using refinement calculi), to compare time abstractions one another (using simulation, refinement, bi-simulation, equivalence relations) but also to prove more specific properties (synchronization, determinism, endochrony). All this formalization effort will allow to effectively perform the tooled validation of common cross-domain properties (e.g. cost v.s. power v.s. performance v.s. software mapping) and tackle equally common yet though case studies such as these linking battery capacity, to on-board CPU performance, to static software schedulability, to logical software correctness and plant controllability: the choice of the right sampling period across the system components.

3.3. Modeling Architectures

To address the formalization of such cross-domain case studies, modeling the architecture formally plays an essential role. An architectural model represents components in a distributed system as boxes with well-defined interfaces, connections between ports on component interfaces, and specifies component properties that can be used in analytical reasoning about the model. Several architectural modeling languages for embedded systems have emerged in recent years, including the SAE AADL [9], SysML [10], UML MARTE [11].

In system design, an architectural specification serves several important purposes. First, it breaks down a system model into manageable components to establish clear interfaces between components. In this way, complexity becomes manageable by hiding details that are not relevant at a given level of abstraction. Clear, formally defined, component interfaces allow us to avoid integration problems at the implementation phase. Connections between components, which specify how components affect each other, help propagate the effects of a change in one component to the linked components.

Most importantly, an architectural model is a repository to share knowledge about the system being designed. This knowledge can be represented as requirements, design artifacts, component implementations, held together by a structural backbone. Such a repository enables automatic generation of analytical models for different aspects of the system, such as timing, reliability, security, performance, energy, etc. Since all the models are generated from the same source, the consistency of assumptions w.r.t. guarantees, of abstractions w.r.t. refinements, used for different analyses becomes easier, and can be properly ensured in a design methodology based on formal verification and synthesis methods.

Related works in this aim, and closer in spirit to our approach (to focus on modeling time) are domain-specific languages such as Prelude [12] to model the real-time characteristics of embedded software architectures. Conversely, standard architecture description languages could be based on algebraic modeling tools, such as interface theories with the ECDAR tool [13].

In project TEA, it takes form by the normalization of the AADL standard's formal semantics and the proposal of a time specification annex in the form of related standards, such as CCSL, to model concurrency time and physical properties, and PSL, to model timed traces.

3.4. Scheduling Theory

Based on sound formalization of time and CPS architectures, real-time scheduling theory provides tools for predicting the timing behavior of a CPS which consists of many interacting software and hardware components. Expressing parallelism among software components is a crucial aspect of the design process of a CPS. It allows for efficient partition and exploitation of available resources.

The literature about real-time scheduling [14] provides very mature schedulability tests regarding many scheduling strategies, preemptive or non-preemptive scheduling, uniprocessor or multiprocessor scheduling, etc. Scheduling of data-flow graphs has also been extensively studied in the past decades.

[9] *Architecture Analysis and Design Language*, AS-5506. SAE, 2004. http://standards.sae.org/as5506b
[10] *System modeling Language*. OMG, 2007. http://www.omg.org/spec/SysML
[11] *UML Profile for MARTE*. OMG, 2009. http://www.omg.org/spec/MARTE
[12] *The Prelude language*. LIFL and ONERA, 2012. http://www.lifl.fr/~forget/prelude.html
[13] *PyECDAR, timed games for timed specifications*. Inria, 2013. https://project.inria.fr/pyecdar
[14] *A survey of hard real-time scheduling for multiprocessor systems*. R. I. Davis and A. Burns. *ACM Computing Survey* 43(4), 2011.

A milestone in this prospect is the development of abstract affine scheduling techniques [15]. It consists, first, of approximating task communication patterns (here Safety-Critical Java threads) using cyclo-static data-flow graphs and affine functions. Then, it uses state of the art ILP techniques to find optimal schedules and concretize them as real-time schedules for Safety Critical Java programs [16] [17].

Abstract scheduling, or the use of abstraction and refinement techniques in scheduling borrowed to the theory of abstract interpretation [18] is a promising development toward tooled methodologies to orchestrate thousands of heterogeneous hardware/software blocks on modern CPS architectures (just consider modern cars or aircrafts). It is an issue that simply defies the state of the art and known bounds of complexity theory in the field, and consequently requires a particular address.

To develop the underlying theory of this promising research topic, we first need to deepen the theoretical foundation to establish links between scheduling analysis and abstract interpretation. A theory of time systems would offer the ideal framework to pursue this development. It amounts to representing scheduling constraints, inferred from programs, as types or contract properties. It allows to formalize the target time model of the scheduler (the architecture, its middle-ware, its real-time system) and defines the basic concepts to verify assumptions made in one with promises offered by the other: contract verification or, in this case, synthesis.

3.5. Virtual Prototyping

Virtual Prototyping is the technology of developing realistic simulators from models of a system under design; that is, an emulated device that captures most, if not all, of the required properties of the real system, based on its specifications. A virtual prototype should be run and tested like the real device. Ideally, the real application software would be run on the virtual prototyping platform and produce the same results as the real device with the same sequence of outputs and reported performance measurements. This may be true to some extent only. Some trade-offs have often to be made between the accuracy of the virtual prototype, and time to develop accurate models.

In order to speed-up simulation time, the virtual prototype must trade-off with something. Depending upon the application designer's goals, one may be interested in trading some loss of accuracy in exchange for simulation speed, which leads to constructing simulation models that focus on some design aspects and provide abstraction of others. A simulation model can provide an abstraction of the simulated hardware in three directions:

- *Computation abstraction*. A hardware component computes a high level function by carrying out a series of small steps executed by composing logical gates. In a virtual prototyping environment, it is often possible to compute the high level function directly by using the available computing resources on the simulation host machine, thus abstracting the hardware function.
- *Communication abstraction*. Hardware components communicate together using some wiring, and some protocol to transmit the data. Simulation of the communication and the particular protocol may be irrelevant for the purpose of virtual prototyping: communication can be abstracted into higher level data transmission functions.
- *Timing Abstraction*. In a cycle accurate simulator, there are multiple simulation tasks, and each task makes some progress on each clock cycle, but this slows down the simulation. In a virtual prototyping experiment, one may not need such precise timing information: coarser time abstractions can be defined allowing for faster simulation.

The cornerstone of a virtual prototyping platform is the component that simulates the processor(s) of the platform, and its associated peripherals. Such simulation can be *static* or *dynamic*.

[15] *Buffer minimization in EDF scheduling of data-flow graphs*. A. Bouakaz and J.-P. Talpin. LCTES, ACM, 2013.
[16] *ADFG for the synthesis of hard real-time applications*. A. Bouakaz, J.-P. Talpin, J. Vitek. ACSD, IEEE, June 2012.
[17] *Design of SCJ Level 1 Applications Using Affine Abstract Clocks*. A. Bouakaz and J.-P. Talpin. SCOPES, ACM, 2013.
[18] *La vérification de programmes par interprétation abstraite*. P. Cousot. Séminaire au Collège de France, 2008.

A solution usually adopted to handle time in virtual prototyping is to manage hierarchical time scales, use component abstractions where possible to gain performance, use refinement to gain accuracy where needed. Localized time abstraction may not only yield faster simulation, but facilitate also verification and synthesis (e.g. synchronous abstractions of physically distributed systems). Such an approach requires computations and communications to be harmoniously discretized and abstracted from originally heterogeneous viewpoints onto a structuring, articulating, pivot model, for concerted reasoning about time and scheduling of events in a way that ensures global system specification correctness.

In the short term these component models could be based on libraries of predefined models of different levels of abstractions. Such abstractions are common in large programming workbench for hardware modeling, such as SystemC, but less so, because of the engineering required, for virtual prototyping platforms.

The approach of team TEA provides an additional ingredient in the form of rich component interfaces. It therefore dictates to further investigate the combined use of conventional virtual prototyping libraries, defined as executable abstractions of real hardware, with executable component simulators synthesised from rich interface specifications (using, e.g., conventional compiling techniques used for synchronous programs).

4. Application Domains

4.1. Automotive and Avionics

From our continuous collaboration with major academic and industrial partners through projects TOPCASED, OPENEMBEDD, SPACIFY, CESAR, OPEES, P and CORAIL, our experience has primarily focused on the aerospace domain. The topics of time and architecture of team TEA extend to both avionics and automotive. Yet, the research focus on time in team TEA is central in any aspect of, cyber-physical, embedded system design in factory automation, automotive, music synthesis, signal processing, software radio, circuit and system on a chip design; many application domains which, should more collaborators join the team, would definitely be worth investigating.

Multi-scale, multi-aspect time modeling, analysis and software synthesis will greatly contribute to architecture modeling in these domains, with applications to optimized (distributed, parallel, multi-core) code generation for avionics (project Corail with Thales avionics, section 8) as well as modeling standards, real-time simulation and virtual integration in automotive (project with Toyota ITC, section 8).

Together with the importance of open-source software, one of these projects, the FUI Project P (section 8), demonstrated that a centralized model for system design could not just be a domain-specific programming language, such as discrete Simulink data-flows or a synchronous language. Synchronous languages implement a fixed model of time using logical clocks that are abstraction of time as sensed by software. They correspond to a fixed viewpoint in system design, and in a fixed hardware location in the system, which is not adequate to our purpose and must be extended.

In project P, we first tried to define a centralized model for importing discrete-continuous models onto a simplified implementation of SIMULINK: P models. Certified code generators would then be developed from that format. Because this does not encompass all aspects being translated to P, the P meta-model is now being extended to architecture description concepts (of the AADL) in order to become better suited for the purpose of system design. Another example is the development of System modeler on top of SCADE, which uses the more model-engineering flavored formalism SysML to try to unambiguously represent architectures around SCADE modules.

An abstract specification formalism, capable of representing time, timing relations, with which heterogeneous models can be abstracted, from which programs can be synthesized, naturally appears better suited for the purpose of virtual prototyping. RT-Builder, based on Signal like Polychrony and developed by TNI, was industrially proven and deployed for that purpose at Peugeot. It served to develop the virtual platform simulating all on-board electronics of PSA cars. This 'hardware in the loop" simulator was used to test equipments supplied by other manufacturers with respect to virtual cars. In the advent of the related automotive standard, RT-Builder then became AUTOSAR-Builder.

4.2. Factory Automation

In collaboration with Mitsubishi R&D, we explore another application domain where time and domain heterogeneity are prime concerns: factory automation. In factory automation alone, a system is conventionally built from generic computing modules: PLCs (Programmable Logic Controllers), connected to the environment with actuators and detectors, and linked to a distributed network. Each individual, physically distributed, PLC module must be timely programmed to perform individually coherent actions and fulfill the global physical, chemical, safety, power efficiency, performance and latency requirements of the whole production chain. Factory chains are subject to global and heterogeneous (physical, electronic, functional) requirements whose enforcement must be orchestrated for all individual components.

Model-based analysis in factory automation emerges from different scientific domains and focus on different CPS abstractions that interact in subtle ways: logic of PLC programs, real-time electromechanical processing, physical and chemical environments. This yields domain communication problems that render individual domain analysis useless. For instance, if one domain analysis (e.g. software) modifies a system model in a way that violates assumptions made by another domain (e.g. chemistry) then the detection of its violation may well be impossible to explain to either of the software and chemistry experts. As a consequence, cross-domain analysis issues are discovered very late during system integration and lead to costly fixes. This is particularly prevalent in multi-tier industries, such as avionic, automotive, factories, where systems are prominently integrated from independently-developed parts.

5. Highlights of the Year

5.1. Highlights of the Year

In 2016, TEA was successfully evaluated, one year after its creation. The team started fruitful collaborations with UC San Diego, with Mitsubishi R&D, with ASTRI, to elaborate our research program on system composition, verification, and simulation toward novel applications perspectives in codesign, operating system design, factory automation, robotics.

6. New Software and Platforms

6.1. ADFG: Affine data-flow graphs scheduler synthesis

Participants: Alexandre Honorat, Jean-Pierre Talpin, Thierry Gautier, Loïc Besnard.

We proposed [2], and implemented [19], a new data-flow design model: ADFG, initially to synthesize schedulers for SCJ/L1 applications. The principle of ADFG is to perform a linear abstraction of complex cyclo-static scheduling problems followed by the exploration of a concrete solution extracted from the abstract solution space, hence the name: abstract affine data-flow scheduling. ADFG guarantees schedules that ordinary (e.g. RTJ, SCJ) task-sets do not cause overflows or underflows. ADFG objectives are to maximize the throughput (the processors utilization) while minimizing buffering storage space needed between actors. ADFG supports EDF and fixed-priority scheduling policies for uni-, multi-processors and distributed systems.

The data-flow design model of ADFG comes with a development tool integrated in the Eclipse IDE for easing the development of SCJ/L1 applications and enforcing the restrictions imposed by the design model. It consists of a GMF editor where applications are designed graphically and timing and buffering parameters can be synthesized. Abstract affine scheduling is first applied on the data-flow subgraph, that consists only of periodic actors, to compute timeless scheduling constraints (e.g. relation between the speeds of two actors) and buffering parameters. Then, symbolic fixed-priority schedulability analysis (i.e., synthesis of timing and scheduling parameters of actors) considers both periodic and aperiodic actors.

[19] *The ADFG tool*, Adnan Bouakaz, http://people.irisa.fr/Adnan.Bouakaz/software.htm

In the case of safety-critical Java, and through a model-to-text transformations using Acceleo, SCJ code for missions, interfaces of handlers, and the mission sequencer is automatically generated in addition to the annotations needed by the memory checker. Channels are implemented as cyclic arrays or cyclical asynchronous buffers; and a fixed amount of memory is hence reused to store the infinite streams of tokens.

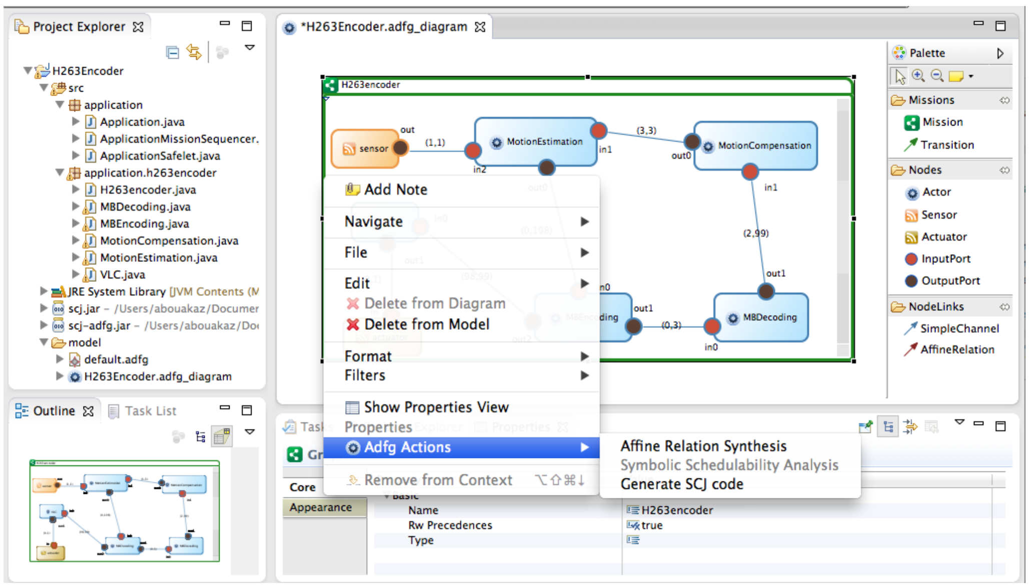


Figure 1. The ADFG tool

6.2. The Eclipse project POP

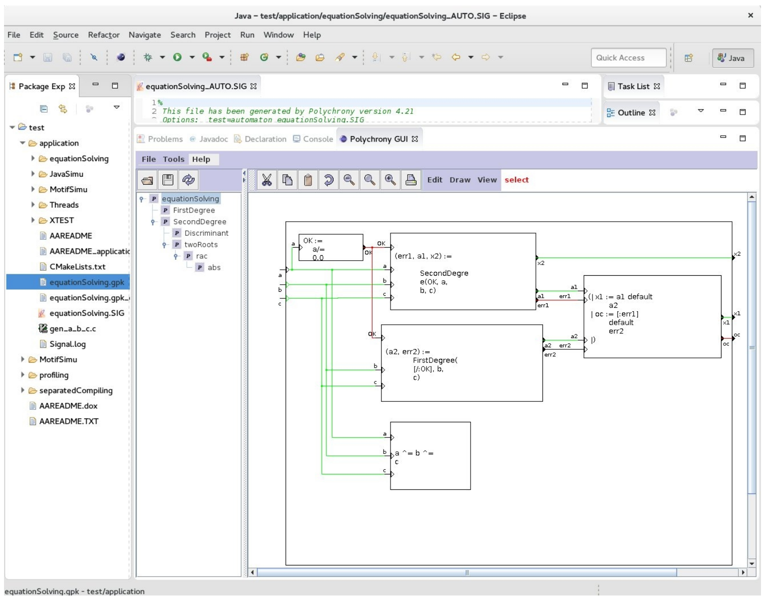
Participants: Loïc Besnard, Thierry Gautier, Jean-Pierre Talpin.

The distribution of project POP is a major achievement of the ESPRESSO (and now TEA) project-team. The Eclipse project POP is a model-driven engineering front-end to our open-source toolset Polychrony. It was finalized in the frame of project OPEES, as a case study: by passing the POLARSYS qualification kit as a computer aided simulation and verification tool. This qualification was implemented by CS Toulouse in conformance with relevant generic (platform independent) qualification documents. Polychrony is now distributed by the Eclipse project POP on the platform of the POLARSYS industrial working group. Project-team TEA aims at continuing its dissemination to academic partners, as to its principles and features, and industrial partners, as to the services it can offer.

Project POP is composed of the Polychrony tool set, under GPL license, and its Eclipse framework, under EPL license. SSME (Syntactic Signal-Meta under Eclipse), is the meta-model of the Signal language implemented with Eclipse/Ecore. It describes all syntactic elements specified in Signal Reference Manual [20]: all Signal operators (e.g. arithmetic, clock synchronization), model (e.g. process frame, module), and construction (e.g. iteration, type declaration). The meta-model primarily aims at making the language and services of the Polychrony environment available to inter-operation and composition with other components (e.g. AADL, Simulink, GeneAuto, P) within an Eclipse-based development tool-chain. Polychrony now comprises the

20

SIGNAL V4-Inria version: Reference Manual. Besnard, L., Gautier, T. and Le Guernic, P. http://www.irisa.fr/espresso/Polychrony, 2010



```
%
This file has been generated by Polychrony version 4.21
Options:  test=automaton equationSolving.SIG
```

Figure 2. The Eclipse POP Environment

capability to directly import and export Ecore models instead of textual Signal programs, in order to facilitate interaction between components within such a tool-chain. The download site for project POP has opened in 2015 at https://www.polarsys.org/projects/polarsys.pop. It should be noted that the Eclipse Foundation does not host code under GPL license. So, the Signal toolbox useful to compile Signal code from Eclipse is hosted on our web server.

6.3. The Polychrony toolset

Participants: Loïc Besnard, Thierry Gautier, Jean-Pierre Talpin.

The Polychrony toolset is an Open Source development environment for critical/embedded systems. It is based on Signal, a real-time polychronous data-flow language. It provides a unified model-driven environment to perform design exploration by using top-down and bottom-up design methodologies formally supported by design model transformations from specification to implementation and from synchrony to asynchrony. It can be included in heterogeneous design systems with various input formalisms and output languages. The Polychrony tool-set provides a formal framework to: validate a design at different levels, by the way of formal verification and/or simulation; refine descriptions in a top-down approach; abstract properties needed for black-box composition; compose heterogeneous components (bottom-up with COTS); generate executable code for various architectures. The Polychrony tool-set contains three main components and an experimental interface to GNU Compiler Collection (GCC):

- The Signal toolbox, a batch compiler for the Signal language, and a structured API that provides a set of program transformations. Itcan be installed without other components and is distributed under GPL V2 license.
- The Signal GUI, a Graphical User Interface to the Signal toolbox (editor + interactive access to compiling functionalities). It can be used either as a specific tool or as a graphical view under Eclipse. In 2015, it has been transformed and restructured, in order to get a more up-to-date interface allowing

multi-window manipulation of programs. It is distributed under GPL V2 license.

- The SSME platform, a front-end to the Signal toolbox in the Eclipse environment. It is distributed under EPL license.

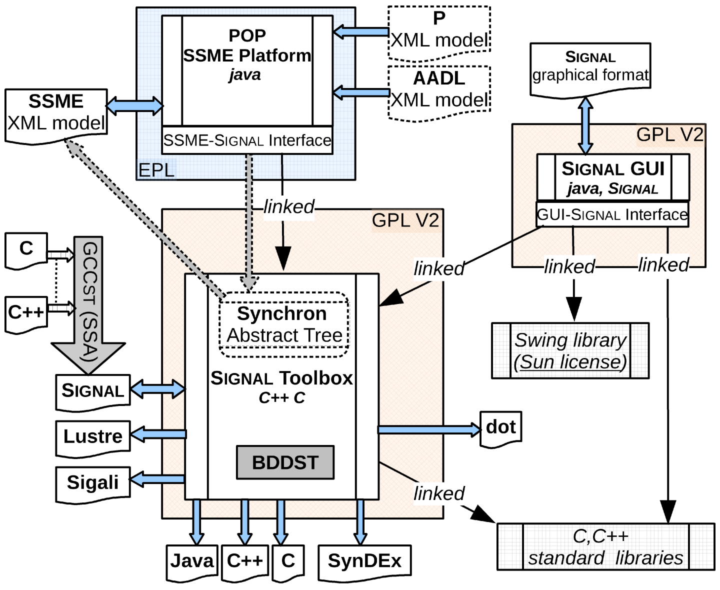


Figure 3. The Polychrony toolset high-level architecture

As part of its open-source release, the Polychrony tool-set not only comprises source code libraries but also an important corpus of structured documentation, whose aim is not only to document each functionality and service, but also to help a potential developer to package a subset of these functionalities and services, and adapt them to developing a new application-specific tool: a new language front-end, a new back-end compiler. This multi-scale, multi-purpose documentation aims to provide different views of the software, from a high-level structural view to low-level descriptions of basic modules. It supports a distribution of the software "by apartment" (a functionality or a set of functionalities) intended for developers who would only be interested by part of the services of the tool-set. The Polychrony tool-set also provides a large library of Signal programs and examples, user documentations and developer-oriented implementation documents, and facilities to generate new versions. The Polychrony tool-set can be freely downloaded from http://polychrony.inria.fr/. This site, intended for users and for developers, contains executable and source versions of the software for different platforms, user documentation, examples, libraries, scientific publications and implementation documentation. In particular, this is the site for the open-source distribution of Polychrony. The Inria GForge https://gforge.inria.fr contains the whole source of the environment and its documentation. It is intended for developers.

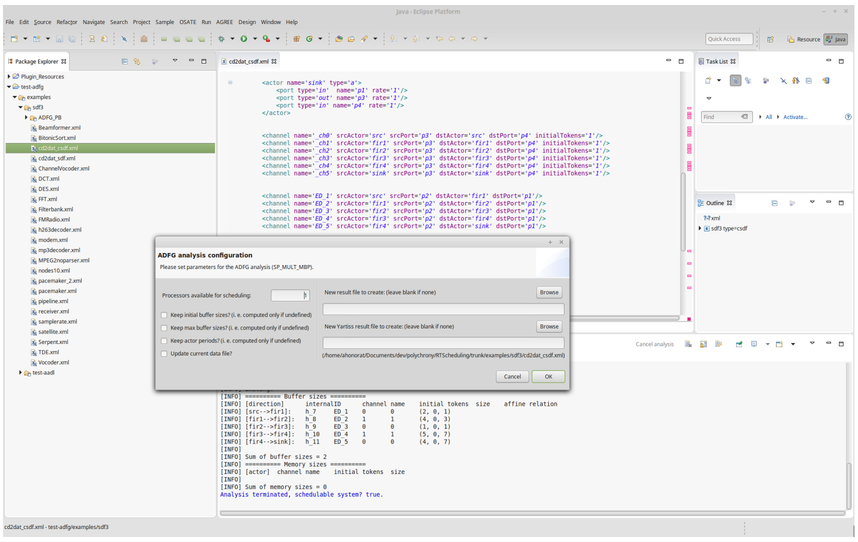
7. New Results

7.1. Toward a distribution of ADFG

Participants: Alexandre Honorat, Jean-Pierre Talpin, Thierry Gautier, Loïc Besnard.

The ADFG tool is being developed in the context of the ADT "Opama" in order to serve both as scheduler synthesis tool from AADL specifications and ordinary tasksets. ADFG has been partly rewritten in order to target more users : it is now freely available online and comes with a complete documentation. These improvements imply that ADFG does not anymore provide Safety Critical Java application generation; its main purpose of scheduler synthesis is now available from an Eclipse plugin, as a command-line interface, and also in Polychrony (as part of the AADL to Signal translation process). Moreover ADFG accepts and exports several file formats with related scheduling tools: SDF3, Yartiss and soon Cheddar.

The Eclipse interface has changed significantly with a dialog window and a console to present the results (as shown in the figure 4). Also the graphical data-flow graph editor is still present but has been simplified. An other big change (not seen by the end-user) is the internal use of the free LpSolve linear programming software instead of CPLEX. Finally, it will soon be possible to use this software not only as a scheduling synthesizer but also as a scheduling checker (with timing properties given by the user).



```
<actor name='sink' type='a'>
    <port type='in'  name='p1' rate='1'/>
    <port type='out' name='p3' rate='1'/>
    <port type='in' name='p4' rate='1'/>
</actor>


<channel name='_ch0' srcActor='src' srcPort='p3' dstActor='src' dstPort='p4' initialTokens='1'/>
<channel name='_ch1' srcActor='fir1' srcPort='p3' dstActor='fir1' dstPort='p4' initialTokens='1'/>
<channel name='_ch2' srcActor='fir2' srcPort='p3' dstActor='fir2' dstPort='p4' initialTokens='1'/>
<channel name='_ch3' srcActor='fir3' srcPort='p3' dstActor='fir3' dstPort='p4' initialTokens='1'/>
<channel name='_ch4' srcActor='fir4' srcPort='p3' dstActor='fir4' dstPort='p4' initialTokens='1'/>
<channel name='_ch5' srcActor='sink' srcPort='p3' dstActor='sink' dstPort='p4' initialTokens='1'/>


<channel name='ED_1' srcActor='src' srcPort='p2' dstActor='fir1' dstPort='p1'/>
<channel name='ED_2' srcActor='fir1' srcPort='p2' dstActor='fir2' dstPort='p1'/>
<channel name='ED_3' srcActor='fir2' srcPort='p2' dstActor='fir3' dstPort='p1'/>
<channel name='ED_4' srcActor='fir3' srcPort='p2' dstActor='fir4' dstPort='p1'/>
<channel name='ED_5' srcActor='fir4' srcPort='p2' dstActor='sink' dstPort='p1'/>
```

ADFG analysis configuration

Please set parameters for the ADFG analysis (SP_MULT_MBP).

Processors available for scheduling: 1

- [ ] Keep initial buffer sizes? (i. e. computed only if undefined)
- [ ] Keep max buffer sizes? (i. e. computed only if undefined)
- [ ] Keep actor periods? (i. e. computed only if undefined)
- [ ] Update current data file?

New result file to create: (leave blank if none)

New Yartiss result file to create: (leave blank if none)

(/home/ahonorat/Documents/dev/polychrony/RTScheduling/trunk/examples/sdf3/cd2dat_csdf.xml)

```
[INFO] ========== Buffer sizes ==========
[INFO] [direction]      internalID      channel name    initial tokens  size    affine relation
[INFO] [src-->fir1]:    h_7     ED_1    0       0       (2, 0, 1)
[INFO] [fir1-->fir2]:   h_8     ED_2    1       1       (4, 0, 3)
[INFO] [fir2-->fir3]:   h_9     ED_3    0       0       (1, 0, 1)
[INFO] [fir3-->fir4]:   h_10    ED_4    1       1       (5, 0, 7)
[INFO] [fir4-->sink]:   h_11    ED_5    0       0       (4, 0, 7)
[INFO]
[INFO] Sum of buffer sizes = 2
[INFO] ========== Memory sizes ==========
[INFO] [actor]  channel name    initial tokens  size
[INFO]
[INFO] Sum of memory sizes = 0
Analysis terminated, schedulable system? true.
```

Figure 4. ADFG under Eclipse

7.2. Modular verification of cyber-physical systems using contract theory

Participants: Jean-Pierre Talpin, Benoit Boyer, David Mentre, Simon Lunel.

The primary goal of our project, in collaboration with Mitsubishi Electronics Research Centre Europe (MERCE), is to ensure correctness-by-design in realistic cyber-physical systems, i.e., systems that mix software and hardware in a physical environment, e.g., Mitsubishi factory automation lines or water-plant factory. To achieve that, we develop a verification methodology based on contract reasoning.

We have first performed a state of the art of the research and the work of A. Platzer with the Differential Dynamic Logic ($d\mathcal{L}$) retained our attention [21]. This a formalism built on the Dynamic Logic of V. Pratt augmented with the possibility of expressing Ordinary Differential Equations (ODEs). ODEs are the usual way to model physical behaviors in physics and $d\mathcal{L}$ permits to accurately model cyber-physical systems. But this logic can also express properties on real arithmetic and there is proof system associated, under the form of a sequent calculus, which let us a mean to prove specifications. To finish, it is very natural to use contract

[21] *Differential Dynamic Logic for Hybrid Systems*, André Platzer, http://symbolaris.com/logic/dL.html

to specify systems since it was the primary goal of the work of V. Pratt. To conclude, $d\mathcal{L}$ is particularly fit to our purpose.

We have some preliminary results about a design-by-composition methodology: we have defined a syntactic composition operator in $d\mathcal{L}$, which enjoys associativity and commutativity. We have then characterized the conditions under which we can derive automatically a proof of the contract of our composition. To exemplified our ideas, we are currently studying a simplified water-tank system, which will serve as a basis for more realistic case studies. We plan to provide refinement and abstraction mechanisms to ultimately allow a mix between vertical and horizontal design.

7.3. Runtime verification and trace analysis

Participants: Vania Joloboff, Daian Yue, Frédéric Mallet.

When engineers design a new cyber physical system, there are well known requirements that can be translated as system properties that must be verified. These properties can be expressed in some formalism and when the model has been designed, the properties can be checked at the model level, using model checking techniques or other model verification techniques.

This requires that the properties are well specified at the time the virtual prototype is assembled. However it is also the case that many intrinsic properties are actually unforeseen when the virtual prototype is assembled, for example that some hardware buffer overflow should not remain unnoticed by the software. In most cases, during system design the simulation fails: the engineers then must investigate the cause of the failure.

A widely used technique for that consists in storing all of the trace data of simulation sessions into trace files, which are analyzed later with specialized trace analyzer tools. Such trace files have become huge, possibly hundred of Gigabytes as all data are stored into the trace files, and have become intractable by human manual handling.

In order to better identify the reason for such failures and capture the missing properties that the system should verify we have started to work on a new run time verification approach based on trace analysis. Approaches like PSL requires that the properties to verify are known before hand. Our approach is attempting for the engineers to experiment various property verification of failing simulations without re-building the virtual prototype. We have established a technique that makes it possible to investigate properties either statically working from a trace file or dynamically by introducing a dynamic verification component into the virtual prototype, or actually the real system.

The Trace Runtime Analysis Platform (TRAP) provides a model-based framework and implements the corresponding tool chain to support runtime analysis and verification of traces generated by virtual prototypes or cyber-physical systems. The main goal is to make it easy for engineers to define system properties that should be satisfied and verify them at system runtime (or from a recorded session). The property verification tools proposed do not require a detailed knowledge of the system implementation, do not require any modification or recompilation of the system to investigate different properties, and do not require the engineers to be familiar with temporal logic. TRAP proposes Domain Specific Languages (DSL's) integrated within the Eclipse Modeling Framework to express the properties. The DSL tool-chain uses the concept of Logical Clock defined by CCSL and takes advantage of CCSL clock algebra as the underlying formal support. The DSL's compilers eventually generate C++ code to verify the properties at run time, making usage of dynamically loaded code.

This year we have investigated and implemented this approach, using Eclipse EMF. The STML and TPSL compilers are implemented in Java and generate C++ code. Results of this work have been published at the FDL'16 conference referenced on IEEE Explore. [17]

7.4. Polychronous automata and formal validation of AADL models

Participants: Loïc Besnard, Thierry Gautier, Alexandre Honorat, Clément Guy, Jean-Pierre Talpin.

We have defined a model of *polychronous automata* based on clock relations [7]. A specificity of this model is that an automaton is submitted to clock constraints: these finite-state automata define transition systems to express explicit reactions together with properties, in the form of Boolean formulas over logical time, to constrain their behavior. This allows one to specify a wide range of control-related configurations, either reactive, or restrictive with respect to their control environment. A semantic model is defined for these polychronous automata, that relies on a Boolean algebra of clocks. Polychronous automata integrate smoothly with data-flow equations in the polychronous model of computation.

This polychronous MoC has been used previously as semantic model for systems described in the core AADL standard. The core AADL is extended with annexes, such as the Behavior Annex, which allows to specify more precisely architectural behaviors. The translation from AADL specifications into the polychronous model should take into account these behavior specifications, which are based on description of automata.

For that purpose, the AADL state transition systems are translated as Signal automata (a slight extension of the Signal language has been defined to support the model of polychronous automata). States are declared as Signal labels. Transitions are expressed using a call to a specific Signal process `Automaton\_Transition` which takes as parameters the labels of the source and destination states, and the condition expression corresponding to the AADL guard of the transition. The transition processes implicitly declare the equations that are required to compute the firing instants of the transitions. These processes, viewed as macros, are replaced during Signal compilation with a set of Signal equations handling current state and transition firing.

Once the AADL model of a system transformed into a Signal program, one can analyze the program using the Polychrony framework in order to check if timing, scheduling and logical requirements over the whole system are met.

We have implemented the translation and experimented it using a concrete case study, which is the AADL modeling of an Adaptive Cruise Control (ACC) system, a highly safety-critical system embedded in recent cars.

7.5. Formal Semantics of Behavior Specifications in the Architecture Analysis and Design Language Standard

Participants: Loïc Besnard, Thierry Gautier, Clément Guy, Jean-Pierre Talpin.

In system design, an architecture specification or model serves, among other purposes, as a repository to share knowledge about the system being designed. Such a repository enables automatic generation of analytical models for different aspects relevant to system design (timing, reliability, security, etc.). The Architecture Analysis and Design Language (AADL) is a standard proposed by SAE to express architecture specifications and share knowledge between the different stakeholders about the system being designed. To support unambiguous reasoning, formal verification, high-fidelity simulation of architecture specifications in a model-based AADL design work-flow, we have defined a formal semantics for the behavior specification of the AADL. Since it began being discussed in the AADL standard committee, our formal semantics evolved from a synchronous model of computation and communication to a semantic framework for time and concurrency in the standard: asynchronous, synchronous or timed, to serve as a reference for model checking, code generation or simulation tools uses with the standard [14]. These semantics are simple, relying on the structure of automata present in the standard already, yet provide tagged, trace semantics framework to establish formal relations between (synchronous, asynchronous, timed) usages or interpretations of behavior.

We define the model of computation and communication of a behavior specification by the synchronous, timed or asynchronous traces of automata with variables. These constrained automata are derived from *polychronous automata* defined within the polychronous model of computation and communication [7].

States of a behavior annex transition system can be either observable from the outside (`initial`, `final` or `complete` states), that is states in which the execution of the component is paused or stopped and its outputs are available; or non observable execution states, that is internal states. We thus define two kinds of steps in the transition system: *small steps*, that is non-observable steps from or to an internal state; and *big steps*, that is observable steps from a *complete* state to another, through a number of small steps). The semantics of the AADL considers the observable states of the automaton. The set of states S_A of automaton A (used to interpret the behavior annex) thus only contains states corresponding to these observable states and the set of transitions T_A big-step transitions from an observable state to another (by opposition with small-step transitions from or to an execution state). The action language of the behavior annex defines actions performed during transitions. Actions associated with transitions are action blocks that are built from basic actions and a minimal set of control structures (sequences, sets, conditionals and loops). Typically, a behavior action sequence is represented by concatenating the transition systems of its elements; a behavior action set is represented by composing the transition systems of its elements.

For our semantics, we considered a significant subset of the behavioral specification annex of the AADL. This annex allows one to attach a behavior specification to any components of a system modeled using the AADL, and can be then analyzed for different purposes which could be, for example, the verification of logical, timing or scheduling requirements.

7.6. Integration of Polychrony with QGen

Participants: Loïc Besnard, Thierry Gautier, Christophe Junke, Jean-Pierre Talpin.

The FUI project P gave birth to the GGen qualifiable model compiler, developed by Adacore. The tool accepts a discrete subset of Simulink expressed in a language called P and produces C or Ada code.

Our contribution was about providing a semantic bridge between Polychrony and QGen [15]. Our objective was to use Polychrony to compute fined-grained static scheduling of computations and communications for P models based on architectural properties. This work was twofold. First, we defined an alternative unambiguous static block scheduler for QGen, which can compute both partial and total orders based on user preferences. The purpose of this sequencer is to allow QGen to inter-operate with external sequencing tools while providing guarantees about the compatibility of external block execution orders with respect to both QGen's compilation scheme and user expectations. On the other hand, we developed a transformation function from the P language, more precisely, from the System Model subset of P, to the Signal meta-model, SSME. This work is based on a high-level API designed on top of SSME and can be used to transform a subset of Simulink to Signal. We validated our approach with the test suite used by QGen which is composed of over two-hundred small-sized Simulink models. We tested both block sequencing and model transformations. We ran the conversion tool and the set of models used by QGen for its regression tests and successfully converted medium to large models. The P language is capable of representing a useful subset of Simulink. That is why it is an interesting tool to help interpreting Simulink models and possibly architectural properties as executable Signal programs. The programs currently produced with our transformation tool can be compiled by Polychrony and reorganized as clusters of smaller processes.

7.7. Code generation for poly-endochronous processes

Participants: Loïc Besnard, Thierry Gautier, Jean-Pierre Talpin.

The synchronous modeling paradigm provides strong correctness guarantees for embedded system design while requiring minimal environmental assumptions. In most related frameworks, global execution correctness is achieved by ensuring the insensitivity of (logical) time in the program from (real) time in the environment. Tis property, called endochrony, can be statically checked, making it fast to ensure design correctness. Unfortunately, it is not preserved by composition, which makes it difficult to exploit with component-based design concepts in mind. It has been shown that compositionality can be achieved by weakening the objective of endochrony: a weakly endochronous system is a deterministic system that can perform independent computations and communications in any order as long as this does not alter its global state. Moreover, the non-blocking composition of weakly endochronous processes is isochronous, which means that the synchronous and asynchronous compositions of weakly endochronous processes accept the same behaviors. Unfortunately, testing weak endochrony needs state-space exploration, which is very costly in the general case. Then, a particular case of weak endochrony, called polyendochrony, was defined, which allows static checking thanks to the existing clock calculus. The clock hierarchy of a polyendochronous system may have several trees, with synchronization relations between clocks placed in different trees, but the clock expressions of the clock system must be such that there is no clock expression (especially, no root clock expression) defined by symmetric difference: root clocks cannot refer to absence. In other words, the clock system must be in disjunctive form [10].

We have now implemented code generation for poly-endochronous systems in Polychrony. This generation reuses techniques of distributed code generation, with rendez-vous management for synchronization constraints on clocks which are not placed in the same tree of clocks. The approach has been validated on several use cases running in parallel with time to time synchronization.

8. Bilateral Contracts and Grants with Industry

8.1. Bilateral Contracts with Industry

8.1.1. Toyota Info-Technology Centre (2014-2016)

Title: Co-Modeling of Safety-Critical Multi-threaded Embedded Software for Multi-Core Embedded Platforms

Inria principal investigator: Jean-Pierre Talpin

International Partner (Institution - Laboratory - Researcher):

Toyota Info-Technology Centre, Mountain View, California

Virginia Tech Research Laboratories, Arlington

Duration: renewed yearly since 2014

Abstract: We started a new project in April 2014 funded by Toyota ITC, California, to work with Huafeng Yu (a former post-doctorate of team ESPRESSO) and with VTRL as US partner. The main topic of our project is the semantic-based model integration of automotive architectures, virtual integration, toward formal verification and automated code synthesis. This year, Toyota ITC is sponsoring our submission for the standardization of a time annex in the SAE standard AADL.

In a second work-package, we aim at elaborating a standardized solution to virtually integrate and simulate a car based on heterogeneous models of its components. This year, it will be exemplified by the elaboration of a case study in collaboration with Virginia Tech. The second phase of the project will consist of delivering an open-source, reference implementation, of the proposed AADL standard and validate it with a real-scale model of the initial case-study.

8.2. Bilateral Grants with Industry

8.2.1. Mitsubishi Electric R&D Europe (2015-2018)

Title: Analysis and verification for correct by construction orchestration in automated factories

Inria principal investigator: Jean-Pierre Talpin, Simon Lunel

International Partner: Mitsubishi Electric R&D Europe

Duration: 2015 - 2018

Abstract: The primary goal of our project is to ensure correctness-by-design in cyber-physical systems, i.e., systems that mix software and hardware in a physical environment, e.g., Mitsubishi factory automation lines. We plan to explore a multi-sorted algebraic framework for static analysis and formal verification starting from a simple use case extracted from Mitsubishi factory automation documentations. This will serve as a basis to more ambitious research where we intend to leverage recent advance in type theory, SMT solvers for nonlinear real arithmetic (dReal and δ-decidability) and contracts theory (meta-theory of Benveniste et al., Ruchkin's contracts) to provide a general framework of reasoning about heterogeneous factory components.

9. Partnerships and Cooperations

9.1. National Initiatives

9.1.1. ANR

Program: ANR

Project acronym: **Feever**

Project title: Faust Environment Everyware

Duration: 2014-2016

Coordinator: Pierre Jouvelot, Mines ParisTech

Other partners: Grame, Inria Rennes, CIEREC

URL: http://www.feever.fr

Abstract:

The aim of project FEEVER is to ready the Faust music synthesis language for the Web. In this context, we collaborate with Mines ParisTech to define a type system suitable to model music signals timed at multiple rates and to formally support playing music synthesized from different physical locations.

9.1.2. PAI CORAC

Program: CORAC

Project acronym: CORAIL

Project title: Composants pour l'Avionique Modulaire Étendue

Duration: July 2013 - May 2017

Coordinator: Thales Avionics

Other partners: Airbus, Dassault Aviation, Eurocopter, Sagem...

Abstract:

The CORAIL project aims at defining components for Extended Modular Avionics. The contribution of project-team TEA is to define a specification method and to provide a generator of multi-task applications.

9.2. International Initiatives

9.2.1. International Project Grants

9.2.1.1. US Air Force Office for Scientific Research – Grant FA8655-13-1-3049

Title: Co-Modeling of Safety-Critical Multi-threaded Embedded Software for Multi-Core Embedded Platforms

Inria principal investigator: Jean-Pierre Talpin

International Partner (Institution - Laboratory - Researcher):

Virginia Tech Research Laboratories, Arlington (United States)

Embedded Systems Group, Teschnische Universität Kaiserslautern (Germany)

Duration: 2013 - 2016

See also: http://www.irisa.fr/espresso/Polycore

Abstract: The aim of the USAF OSR Grant FA8655-13-1-3049 is to support collaborative research entitled “Co-Modeling of safety-critical multi-threaded embedded software for multi-core embedded platforms” between Inria project-team ESPRESSO, the VTRL Fermat Laboratory and the TUKL embedded system research group, under the program of the Polycore associate-project.

9.2.1.2. Applied Science & Technology Research Institute (ASTRI, Hong Kong)

Title: Virtual Prototyping of Embedded Software Architectures

Inria principal investigator: Jean-Pierre Talpin

International Partner: ASTRI, Hong Kong

Duration: 2015 - 2016

Abstract: the topics of our present collaboration is essentially on heterogeneous time modeling for virtual prototyping in cyber-physical systems. Our project covers a wide spectrum of area of experience developed since 2012 and comprising

- model-based design and analysis of cyber-physical systems;
- system-level virtual prototyping and validation;
- design space exploration and system synthesis;

9.2.2. Inria International Labs

9.2.2.1. SACCADES

Title: Saccades

International Partner:

LIAMA

East China Normal University

Inria project-teams Aoste and Tea

Duration: 2003 - now

The SACCADES project is a LIAMA project hosted by East China Normal University and jointly led by Vania Joloboff (Inria) and Min Zhang (ECNU). The SACCADES project aims at improving the development of reliable cyber physical systems and more generally of distributed systems combining asynchronous with synchronous aspects, with different but complementary angles:

- develop the theoretical support for Models of Computations and Communications (MoCCs) that are the fundamentals basis of the tools.
- develop software tools (a) to enable the development and verification of executable models of the application software, which may be local or distributed and (b) to define and optimize the mapping of software components over the available resources.
- develop virtual prototyping technology enabling the validation of the application software on the target hardware platform.

The ambition of SACCADES project is to develop

- Theoretical Support for Cyber Physical Systems
- Software Tools for design and validation of CPS
- Virtual Prototyping of CPS

9.2.3. Inria International Partners

9.2.3.1. POLYCORE

Title: Models of computation for embedded software design

International Partner:

Virginia Tech Research Laboratories (USA)

University of Kanpur (India)

Duration: 2002 - now

Team TEA collaborates with Sandeep Shukla (now with IIT Kanpur) and his team at Virginia Tech, since 2002 (NSF-Inria BALBOA and Polycore projects, USAF OSR grant).

To date, our fruitful and sustained collaboration has yield the creation of the ACM-IEEE MEMOCODE conference series in 2003, of the ACM-SIGDA FMGALS workshop series, and of a full-day tutorial at ACM-IEEE DATE'09 on formal methods in system design. We have jointly edited two books with Springer [22] [23], two special issues of the IEEE Transactions on Computers and one of the IEEE Transactions on Industrial Informatics, and published more than 40 joint journal articles and conference papers. We published a joint paper at the 52nd. Digital Automation Conference in San Francisco [11].

9.2.3.2. VESA

Title: Virtual Prototyping of embedded software architectures

International Partner:

Applied Science & Technology Research Institute (ASTRI, Hong Kong)

The University of Hong Kong

Duration: 2012 - now

We collaborate with John Koo, now with ASTRI, and LIAMA since 2012 through visiting grants of the Chinese Academy of Science and of the University of Rennes on the topics of heterogeneous time modeling and virtual prototyping in cyber-physical systems.

In the context of project ITF ARD159 (System-Level Virtual Prototyping of Embedded Systems), ASTRI has used Polychrony and AADL to collaboratively develop a platform for conducting the design of an hardware-in-the-loop simulation of an UR5 robot arm, from its physical model described using Matlab/Simulink and powered using an Opal-RT/RT-Lab workstation, structured around an AADL system model, and using Polychrony to orchestrate real-time simulation down to FPGA analog outputs.

9.2.3.3. TIX

Title: Time In Cybernetic Systems

International Partner:

Rajesh Gupta, UCSD

Mani Srivastava, UCLA

Start year: 2015

The first topic of our collaboration is the formal definition of cross-domains clock models in system design and the formal verification of time stabilization and synchronization protocols used in distributed systems (sensor networks, data-bases). In this prospect, the NSF project Roseline is our basis of investigation (https://sites.google.com/site/roselineproject). Roseline aims at enabling robust, secure and efficient knowledge of time across the system stack.

Our second topic of collaboration is the refoundation of time modeling in high-level reactive and scripting languages, for application to the above using uni-kernels to cut through system stacks. We aim at applying the concepts of refinement types to formally specify and infer timing properties in CPS models from different system design view-point (physical, hardware, software) and using different levels of abstraction into multi-sorted 1st order logic (delta-decidability, linear arithmetic, Boolean logic, temporal logic).

9.3. International Research Visitors

9.3.1. Visits of International Scientists

Rajesh Gupta (UC San Diego) visited project TEA in July 2016 in the context of IIP TIX.

[22] *Formal methods and models for system design*, R. Gupta, S. Shukla, J.-P. Talpin, Eds. ISBN 1-4020-8051-4. Springer, 2004.
[23] *Synthesis of embedded systems*. S. Shukla, J.-P. Talpin, Eds. ISBN 978-1-4419-6399-4. Springer, 2010

Brian Larson (FDA) visited project TEA in January and July 2016.

9.3.1.1. Internships

Daian Yue that was selected in the joint program between ENS Rennes and ECNU and joined project TEA for a six month internship in 2016.

9.3.2. Visits to International Teams

Vania Joloboff was invited for two short stays at University of East China Normal University in Shanghai and UC San Diego.

Jean-Pierre Talpin visited ASTRI in May and December, in the context of IIP VESA.

Jean-Pierre Talpin visited UC San Diego in October, in the context of IIP TIX.

Jean-Pierre Talpin visited IIT Kanpur in February and November for the preparation and Chair of MEMOCODE'16.

10. Dissemination

10.1. Promoting Scientific Activities

10.1.1. Scientific events organisation

10.1.1.1. General chair, scientific chair

Jean-Pierre Talpin served as General Chair and Finance Chair of the 14th. ACM-IEEE Conference on Methods and Models for System Design (MEMOCODE'16, IIT Kanpur, October 18-20.).

10.1.1.2. Member of the organizing committees

Jean-Pierre Talpin and Vania Joloboff co-organized the Shonan workshop on "Architecture-Centric Modeling, Analysis, and Verification of Cyber-Physical Systems" in collaboration with Toyota ITC and Denso, March 21-24.

Jean-Pierre Talpin is a member of the steering committee of the ACM-IEEE Conference on Methods and Models for System Design (MEMOCODE).

10.1.2. Scientific events selection

10.1.2.1. Member of the conference program committees

Jean-Pierre Talpin served the program committee of:

- ACVI'16, 3rd. Workshop on Architecture Centric Virtual Integration
- HLDVT'16, 18th. IEEE International High-Level Design Validation and Test Workshop
- ICESS'16, 13th. IEEE International Conference on Embedded Software and Systems
- IDEA'16, 2nd. International Workshop Integrating Data-flow, Embedded computing and Architecture
- LCTES'16, 19th. ACM SIGPLAN-SIGBED Conference on Languages, Compilers, and Tools for Embedded Systems
- MEMOCODE'16, 14th. ACM-IEEE Conference on Methods and Models for System Design
- SAC'16, 31st. ACM SIGAPP Symposium on Applied Computing
- SCOPES'16, 19th. International Workshop on Software and Compilers for Embedded Systems
- TASE'16, 10th. Theoretical Aspects of Software Engineering Conference

10.1.3. Journal

10.1.3.1. Member of the editorial boards

Jean-Pierre Talpin is Associate Editor with the ACM Transactions for Embedded Computing Systems (TECS), with the Springer journal on Frontiers of Computer Science (FCS) and with the EURASIP journal of embedded systems (JES).

10.1.3.2. reviewer

Jean-Pierre Talpin reviewed articles for Acta Informatica.

Thierry Gautier reviewed for Frontiers of Computer Science.

10.2. Teaching - Supervision - Juries

10.2.1. Invited talks

Vania Joloboff gave a talk in the series of the Distinguished Lecturers of the Computer Science and Engineering department at UC San Diego.

Jean-Pierre Talpin gave an invited presentation at the APAC 2016 Summit on Robotics at the HKSTP in Hong Kong https://www.apacinnosummit.net.

10.2.2. Supervision

Vania Joloboff supervised work of Master student Daian Yue that was selected in the joint program between ENS Rennes and ECNU.

Jean-Pierre Talpin is the supervisor of Simon Lunel's thesis on "*Timed contract algebras for correct by construction real-time system design*".

10.2.3. Juries

Jean-Pierre Talpin served as examiner for the Ph.D. Thesis defense of Fatma Jebali on "Formal Framework for modeling and Verifying Globally Asynchronous Locally Synchronous Systems", September 12., in Grenoble.

Jean-Pierre Talpin served as referee for the PhD. Thesis Defence of Amani Khecharem on "Une approche de méta-modélisation pour la représentation multi-vues des architectures hétérogènes embarqués", May 3., in Sophia Antipolis.

11. Bibliography

Major publications by the team in recent years

[1] L. BESNARD, T. GAUTIER, P. LE GUERNIC, J.-P. TALPIN. *Compilation of Polychronous Data Flow Equations*, in "Synthesis of Embedded Software", S. K. SHUKLA, J.-P. TALPIN (editors), Springer, 2010, pp. 1-40 [*DOI :* 10.1007/978-1-4419-6400-7_1], http://hal.inria.fr/inria-00540493

[2] A. BOUAKAZ, J.-P. TALPIN. *Design of Safety-Critical Java Level 1 Applications Using Affine Abstract Clocks*, in "International Workshop on Software and Compilers for Embedded Systems", St. Goar, Germany, June 2013, pp. 58-67 [*DOI :* 10.1145/2463596.2463600], https://hal.inria.fr/hal-00916487

[3] C. BRUNETTE, J.-P. TALPIN, A. GAMATIÉ, T. GAUTIER. *A metamodel for the design of polychronous systems*, in "The Journal of Logic and Algebraic Programming", 2009, vol. 78, n^{o} 4, pp. 233 - 259, IFIP WG1.8 Workshop on Applying Concurrency Research in Industry [*DOI :* 10.1016/J.JLAP.2008.11.005], http://www.sciencedirect.com/science/article/pii/S1567832608000957

[4] A. GAMATIÉ, T. GAUTIER, P. LE GUERNIC, J.-P. TALPIN. *Polychronous Design of Embedded Real-Time Applications*, in "ACM Transactions on Software Engineering and Methodology (TOSEM)", April 2007, vol. 16, n^{o} 2, http://doi.acm.org/10.1145/1217295.1217298

[5] A. GAMATIÉ, T. GAUTIER. *The Signal Synchronous Multiclock Approach to the Design of Distributed Embedded Systems*, in "IEEE Transactions on Parallel and Distributed Systems", 2010, vol. 21, n^{o} 5, pp. 641-657 [*DOI :* 10.1109/TPDS.2009.125], http://hal.inria.fr/inria-00522794

[6] A. GAMATIÉ, T. GAUTIER, P. LE GUERNIC. *Synchronous design of avionic applications based on model refinements*, in "Journal of Embedded Computing (IOS Press)", 2006, vol. 2, n^{o} 3-4, pp. 273-289, http://hal.archives-ouvertes.fr/hal-00541523

[7] T. GAUTIER, C. GUY, A. HONORAT, P. LE GUERNIC, J.-P. TALPIN, L. BESNARD. *Polychronous Automata and their Use for Formal Validation of AADL Models*, in "Frontiers of Computer Science -Springer-", December 2016, https://hal.inria.fr/hal-01411257

[8] P. LE GUERNIC, J.-P. TALPIN, J.-C. LE LANN. *Polychrony for system design*, in "Journal of Circuits, Systems and Computers, Special Issue on Application Specific Hardware Design", June 2003, vol. 12, n^{o} 03, http://hal.inria.fr/docs/00/07/18/71/PDF/RR-4715.pdf

[9] D. POTOP-BUTUCARU, Y. SOREL, R. DE SIMONE, J.-P. TALPIN. *From Concurrent Multi-clock Programs to Deterministic Asynchronous Implementations*, in "Fundamenta Informaticae", January 2011, vol. 108, n^{o} 1-2, pp. 91–118, http://dl.acm.org/citation.cfm?id=2362088.2362094

[10] J.-P. TALPIN, J. OUY, T. GAUTIER, L. BESNARD, P. LE GUERNIC. *Compositional design of isochronous systems*, in "Science of Computer Programming", February 2012, vol. 77, n^{o} 2, pp. 113-128 [*DOI :* 10.1016/J.SCICO.2010.06.006], http://hal.archives-ouvertes.fr/hal-00768341

[11] H. YU, J. PRASHI, J.-P. TALPIN, S. K. SHUKLA, S. SHIRAISHI. *Model-Based Integration for Automotive Control Software*, in "Digital Automation Conference", San Francisco, United States, ACM, June 2015, https://hal.inria.fr/hal-01148905

Publications of the year

Articles in International Peer-Reviewed Journals

[12] T. GAUTIER, C. GUY, A. HONORAT, P. LE GUERNIC, J.-P. TALPIN, L. BESNARD. *Polychronous Automata and their Use for Formal Validation of AADL Models*, in "Frontiers of Computer Science -Springer-", December 2016, https://hal.inria.fr/hal-01411257

International Conferences with Proceedings

[13] D. BAELDE, S. LUNEL, S. SCHMITZ. *A Sequent Calculus for a Modal Logic on Finite Data Trees*, in "CSL 2016", Marseille, France, J.-M. TALBOT, L. REGNIER (editors), Leibniz International Proceedings in Informatics, LZI, September 2016, vol. 62, n^{o} 32, pp. 1–16 [*DOI :* 10.4230/LIPICS.CSL.2016.32], https://hal.inria.fr/hal-01191172

[14] L. BESNARD, T. GAUTIER, C. GUY, P. LE GUERNIC, J.-P. TALPIN, B. R. LARSON, E. BORDE. *Formal semantics of behavior specifications in the architecture analysis and design language standard*, in "18th IEEE International High-Level Design Validation and Test Workshop", Santa Cruz, United States, High-level design,

verification and test, IEEE, October 2016, pp. 30 - 39 [*DOI :* 10.1109/HLDVT.2016.7748252], https://hal.inria.fr/hal-01419968

[15] C. JUNKE, T. GAUTIER, J.-P. TALPIN, L. BESNARD. *Integration of polychrony and QGen model compiler*, in "ERTS'16 - European Congress on Embeddd Real-Rime Software and Systems", Toulouse, France, January 2016, https://hal.inria.fr/hal-01241808

[16] V. C. NGO, A. LEGAY, V. JOLOBOFF. *PSCV: A Runtime Verification Tool for Probabilistic SystemC Models*, in "CAV 2016 - 28th International Conference on Computer Aided Verification", Toronto, Canada, S. CHAUDHURI, A. FARZAN (editors), LNCS - Lecture Notes in Computer Science, Springer, July 2016, vol. 9779, pp. 84 - 91 [*DOI :* 10.1007/978-3-319-41528-4_5], https://hal.inria.fr/hal-01406488

[17] D. YUE, V. JOLOBOFF, F. MALLET. *Flexible Runtime Verification Based On Logical Clock Constraints*, in "FDL 2016 - Forum on specification & Design Languages", Bremen, Germany, September 2016, https://hal.inria.fr/hal-01421890

Conferences without Proceedings

[18] O. AUMAGE, D. BARTHOU, A. HONORAT. *A Stencil DSEL for Single Code Accelerated Computing with SYCL*, in "SYCL 2016 1st SYCL Programming Workshop during the 21st ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming", Barcelone, Spain, March 2016, https://hal.archives-ouvertes.fr/hal-01290099

[19] O. SANKUR, J.-P. TALPIN. *An Abstraction Technique For Parameterized Model Checking of Leader Election Protocols: Application to FTSP*, in "23rd International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)", Uppsala, Sweden, April 2017, https://hal.archives-ouvertes.fr/hal-01431472

Research Reports

[20] L. BESNARD, T. GAUTIER, P. LE GUERNIC, C. GUY, J.-P. TALPIN, B. LARSON, E. BORDE. *Formal semantics of behavior specifications in the architecture analysis and design language standard*, Inria, 2016, n^{o} 8950, pp. 30 - 39 [*DOI :* 10.1109/HLDVT.2016.7748252], https://hal.inria.fr/hal-01419973