Activity Report 2015

Project-Team TEA

Time, Events and Architectures

RESEARCH CENTER
Rennes - Bretagne-Atlantique

THEME
Embedded and Real-time Systems

Table of contents

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
2.1.10. - Domain-specific languages
2.1.6. - Concurrent programming
2.1.8. - Synchronous 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. - Reliability, certification
2.4.1. - Analysis
2.4.2. - Verification
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 [Team leader, Inria, Senior Researcher, HdR]
Thierry Gautier [Inria, Researcher]
Vania Joloboff [Inria, Senior Researcher]
Paul Le Guernic [Inria, Senior Researcher]

Engineers
Clement Guy [Inria]
Alexandre Honorat [Inria]
Christophe Junke [Inria]

PhD Student
Simon Lunel [Mitsubishi Electric R&D, granted by CIFRE]

Administrative Assistant
Stephanie Lemaile [Inria]

2. Overall Objectives

2.1. Introduction

An embedded architecture is an artefact of heterogeneous constituants 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, modelling 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, parallelisation, simulation. The aim of project-team TEA is to define a conceptually unified framework for reasoning on time in cyber-physical system design, and to put this reflection to practice by revisiting analysis and synthesis issues in real-time system design with soundness and compositionality gained from formalisation.

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 revolutionised 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 nothing new. In fact, it 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 modelling formalisms that have appeared in the past ten years with more or less success. The most successful are the *executable* models, models that can be 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 synchronisation, 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 behaviour usually using real numbers (irrational) variables, the models usually have to work with discrete time and approximate floating point variables.

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, parallelisation, 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 modelling 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 behaviour, 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 discretisation 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, or put simply for time reasoning, 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 ressources 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 organise 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 behaviour, 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 [1] and in the CCSL standard [2], 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

[1] *Polychrony on POLARSYS*, an Eclipse project in the POLARSYS Industry Working Group, 2013. https://www.POLARSYS.org/projects/POLARSYS.pop

[2] *Clock Constraints in UML/MARTE CCSL*. C. André, F. Mallet. Technical Report RR-6540. Inria, 2008. http://hal.inria.fr/inria-00280941

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. Modelling Times

To make a sense and eventually formalize the semantcs 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 [3], 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 [4], pre-post condition, value-range, cost, speed, time, temporal logic [5].

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 behaviours of system components, timed relations (abstraction and refinement) represent interaction among components.

Scalability and compositionality dictates the use of assume-guarantee reasoning to represent them, and to facilitate composition by behavioural sub-typing, in the spirit of the (static) contract-based formalism proposed by Passerone et al. [6]. 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 middleware or hardware).

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, bisimulation, equivalence relations) but also to prove more specific properties (synchronisation, determinism, endochrony).

To check conformance between heterogeneously timed specifications, we will consider variants of the abstract interpretation framework proposed by Bertrane et al. [7] to inject properties from one time domain into another, continuous [8] or discrete [9].

[3] *Abstract Refinement Types*. N. Vazou, P. Rondon, and R. Jhala. European Symposium on Programming. Springer, 2013.

[4] *Region-based memory management*. Tofte, M., Talpin, J.-P. Information and Computation, 1997.

[5] *LTL types FRP*. A. Jeffrey. PLPV’12.

[6] *A contract-based formalism for the specification of heterogeneous systems*. L. Benvenistu, A. Ferrari, L. Mangeruca, E. Mazzi, R. Passerone, C. Sofronis. Forum on design languages, 2008

[7] *Temporal Abstract Domains*. J. Bertrane. International Conference on Engineering of Complex Computer Systems. IEEE, 2011

[8] *Abstract Interpretation of the Physical Inputs of Embedded Programs*. O. Bouissou, M. Martel. Verification, Model Checking, and Abstract Interpretation. LNCS 4905, Springer, 2008

All this formalisation 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 onboard 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. Modelling Architectures

To address the formalisation of such cross-domain case studies, modelling 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 modelling languages for embedded systems have emerged in recent years, including the SAE AADL [10], SysML [11], UML MARTE [12].

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 artefacts, 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 modelling time) are domain-specific languages such as Prelude [13] to model the real-time characteristics of embedded software architectures. Conversely, standard architecture description languages could be based on algebraic modelling tools, such as interface theories with the ECDAR tool [14].

In project TEA, it takes form by the normalisation 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. Application to Scheduling Theory

Based on sound formalisation of time and CPS architectures, real-time scheduling theory provides tools for predicting the timing behaviour 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 [15] 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] *Proving the Properties of Communicating Imperfectly-Clocked Synchronous Systems*. J. Bertrane. Static Analysis Symposium. Springer, 2006

[10] *Architecture Analysis and Design Language*, AS-5506. SAE, 2004. http://standards.sae.org/as5506b

[11] *System Modelling Language*. OMG, 2007. http://www.omg.org/spec/SysML

[12] *UML Profile for MARTE*. OMG, 2009. http://www.omg.org/spec/MARTE

[13] *The Prelude language*. LIFL and ONERA, 2012. http://www.lifl.fr/~forget/prelude.html

[14] *PyECDAR, timed games for timed specifications*. Inria, 2013. https://project.inria.fr/pyecdar

[15] *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 [16]. 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 concretise them as real-time schedules for Safety Critical Java programs [17] [18].

Abstract scheduling, or the use of abstraction and refinement techniques in scheduling borrowed to the theory of abstract interpretation [19] 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 formalise 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.

[16] *Buffer minimization in earliest-deadline first scheduling of dataflow graphs*. A. Bouakaz and J.-P. Talpin. Conference on Languages, Compilers and Tools for Embedded Systems. ACM, June 2013.

[17] *Affine data-flow graphs for the synthesis of hard real-time applications*. A. Bouakaz, J.-P. Talpin, and J. Vitek. Application of Concurrency to System Design. IEEE Press, June 2012.

[18] *Design of Safety-Critical Java Level 1 Applications Using Affine Abstract Clocks*. A. Bouakaz and J.-P. Talpin. International Workshop on Software and Compilers for Embedded Systems. ACM, June 2013.

[19] *La vérification de programmes par interprétation abstraite*. P. Cousot. Séminaire au Collège de France, 2008.

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*.

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. Localised 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 discretised 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 modelling, 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 modelling, analysis and software synthesis will greatly contribute to architecture modelling in these domains, with applications to optimised (distributed, parallel, multi-core) code generation for avionics (project Corail with Thales avionics, section 8) as well as modelling 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 centralised 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 centralised 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 Modeller on top of SCADE, which uses the more model-engineering flavoured 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 synthesised, 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 onboard 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 avent of the related automotive standard, RT-Builder then became AUTOSAR-Builder.

4.2. Factory Automation

In the new collaboration with Mitsubishi R&D, started in 2015, 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, electronical, 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

TEA became an Inria project-team in 2015 and developed new and promising collaborations with Mitsubishi, on factory automations, with UCSD on refinement type theory and with UCSD-UCLA again, on time synchronisation protocols verificaton.

We published a paper in the automotive session of the 52nd. Digital Automation Conference (core A*) on our project with Toyota ITC [19] as well as two patents filed with the USPTO.

5.1.1. Awards

Our paper on "Polychronous automata" [13] received the Best Paper Award at the TASE'15 conference.

BEST PAPER AWARD:

[13]
P. LE GUERNIC, T. GAUTIER, J.-P. TALPIN, L. BESNARD. *Polychronous Automata*, in "TASE 2015, 9th International Symposium on Theoretical Aspects of Software Engineering", Nanjing, China, IEEE Computer Society, September 2015, pp. 95-102 [*DOI :* 10.1109/TASE.2015.21], https://hal.archives-ouvertes.fr/hal-01240440

6. New Software and Platforms

6.1. The Eclipse project POP

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

The distribution of project POP [20] 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 finalised 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.

Technically, project POP is composed of the Polychrony toolset, under GPL license, and its Eclipse framework, under EPL license. SSME (Syntactic Signal-Meta under Eclipse), is the metamodel of the Signal language implemented with Eclipse/Ecore. It describes all syntactic elements specified in Signal Reference Manual [21]: all Signal operators (e.g. arithmetic, clock synchronization), model (e.g. process frame, module), and construction (e.g. iteration, type declaration).

The metamodel 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 toolchain. Polychrony now comprises the capability to directly import and export Ecore models instead of textual Signal programs, in order to facilitate interaction between components within such a toolchain.

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.

6.2. The Polychrony toolset

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

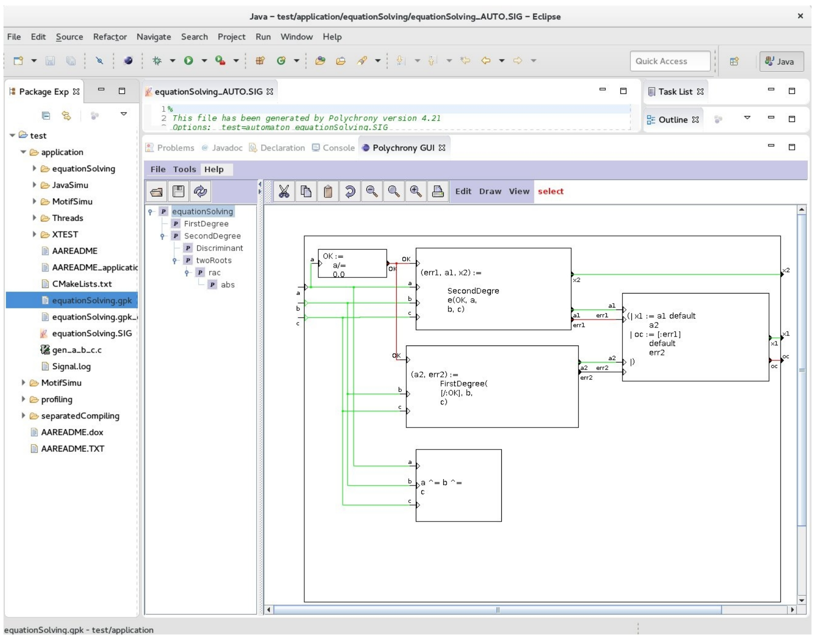
The Polychrony toolset is an Open Source development environment for critical/embedded systems. It is based on Signal, a real-time polychronous dataflow language. It provides a unified model-driven environment to perform design exploration by using top-down and bottom-up design methodologies formally 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 toolset 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,
- assemble heterogeneous predefined components (bottom-up with COTS),
- generate executable code for various architectures.

[20] *Polychrony on POLARSYS (POP)*, an Eclipse project in the POLARSYS Industry Working Group, 2013. https://www.POLARSYS.org/projects/POLARSYS.pop

[21] *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 1. The Eclipse POP Environment

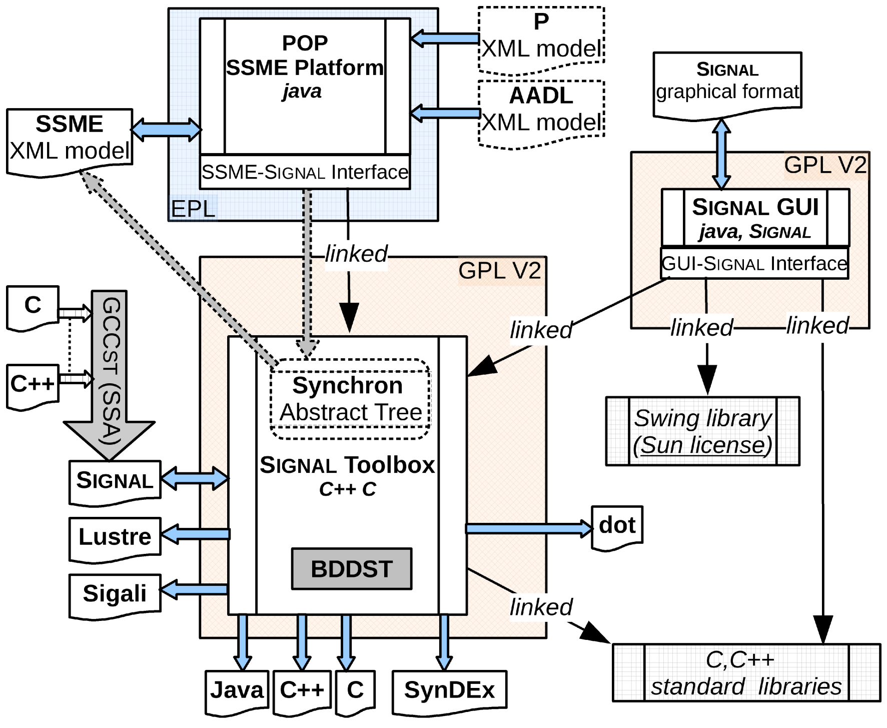


Figure 2. The Polychrony toolset high-level architecture

The Polychrony toolset 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.
- GCCst, a back-end to GCC that generates Signal programs (not yet available for download).

The Polychrony toolset 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 toolset can be freely downloaded on the following web sites:

- The Polychrony toolset public web site: http://polychrony.inria.fr/. This site, intended for users and for developers, contains downloadable executable and source versions of the software for differents 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. This site, intended for internal developers, contains the whole sources of the environment and their documentation.

As part of its open-source release, the Polychrony toolset 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 toolset.

6.3. SigCert: translation validation from Signal to C

Participants: Van-Chan Ngo, Jean-Pierre Talpin, Thierry Gautier, Paul Le Guernic, Loïc Besnard.

Translation validation [22] [23] is a technique that attempts to verify that program transformations preserve the program semantics. It is obvious to prove globally that the source program and its final compiled program have the same semantics. However, we believe that a better approach is to separate concerns and prove each analysis and transformation stage separately with respect to ad-hoc data-structures to carry the semantic information relevant to that phase.

In the case of the Signal compiler [1], [7], the preservation of the semantics can be decomposed into the preservation of clock semantics at the *clock calculation* phase [15] and that of data dependencies at the *static scheduling* phase[16], and, finally, value-equivalence of variables at the *code generation* phase[14].

Translation Validation for Clock Transformations in a Synchronous Compiler. The clock semantics of the source and transformed programs are formally represented as *clock models*. A clock model is a first-order logic formula that characterizes the presence/absence status of all signals in a Signal program at a given instant. Given two clock models, a *clock refinement* between them is defined which expresses the semantic preservation of clock semantics[15]. A method to check the existence of clock refinement is defined as a satisfiability problem which can be automatically and efficiently proved by a SMT solver [24].

[22] *Translation validation.* Pnueli A., Siegel M., and Singerman E. In Proceedings of TACAS'98, 1998.

[23] *Translation validation: From signal to c.* M. Siegel A. Pnueli and E. Singeman. In Correct Sytem Design Recent Insights and Advances, 2000.

[24] *Satisfiability modulo theories: An appetizer.* L. de Moura and N. Bjorner. In Brazilian Symposium on Formal Methods, 2009.

Precise Deadlock Detection for Polychronous Data-flow Specifications. Dependency graphs are a commonly used data structure to encode the streams of values in data-flow programs and play a central role in scheduling instructions during automated code generation from such specifications. We propose a precise and effective method that combines a structure of dependency graph and first order logic formulas to check whether multi-clocked data-flow specifications are deadlock-free before generating code from them. We represent the flow of values in the source programs by means of a dependency graph and attach first-order logic formulas to condition these dependencies. We use an SMT solver to effectively reason about the implied formulas and check deadlock freedom [16].

Implementation and Experiments. At a high level, our prototype tool *SigCert* ([14]) developed in OCaml could check the correctness of the compilation of Signal compiler w.r.t clock semantics, data dependence, and value-equivalence as given in Figure 3. The individual modules designed in the context of this work are now being implemented and integrated in the open-source Polychrony toolset.

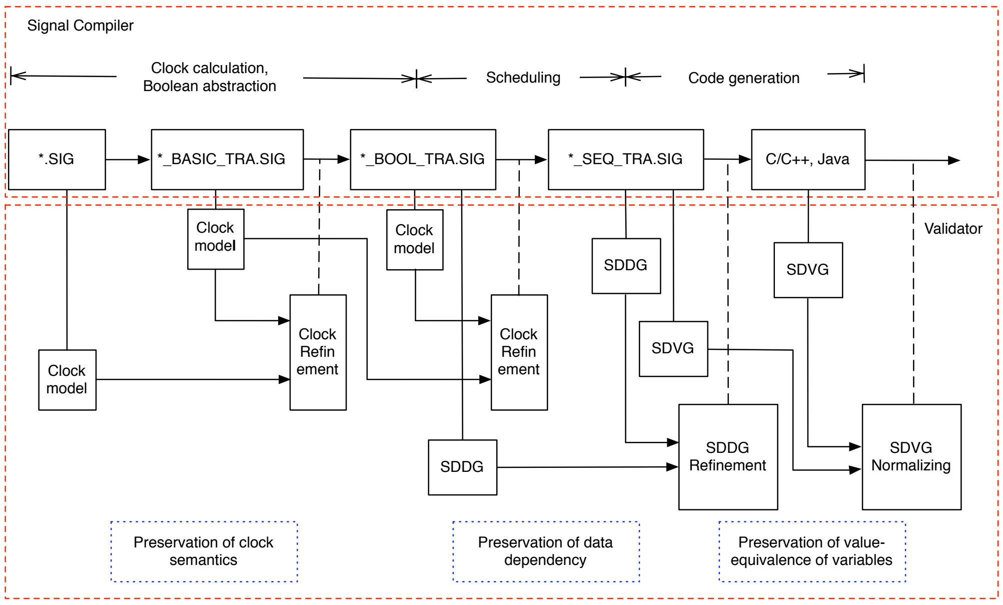


Figure 3. Our Integration within Polychrony Toolset

6.4. ADFG: Affine data-flow graphs scheduler synthesis under Eclipse

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

We have proposed a dataflow design model [2] of SCJ/L1 applications [25] in which handlers (periodic and aperiodic actors) communicate only through lock-free channels. Hence, each mission is modeled as a dataflow graph. The presented dataflow design model 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. Indeed, abstract affine scheduling is first applied on the dataflow 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.

[25] *Safety critical Java technology specification*. JSR-302, Year = 2010

Through a model-to-text transformation, using Acceleo, the 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. The user must provide the SCJ code of all the `handleAsyncEvent()` methods. We have integrated the SCJ memory checker [26] in our tool so that potential dangling pointers can be highlighted at compile-time. To enhance functional determinism, we would like to develop an ownership type system to ensure that actors are strongly isolated and communicate only through buffers.

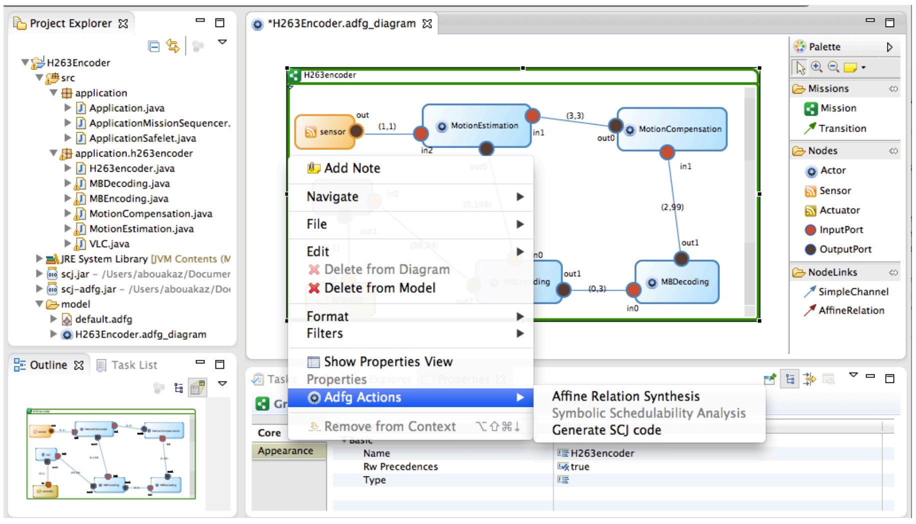


Figure 4. The ADFG Tool

The ADFG tool is being further developed in the context of the ADT "La vie d'AADL" in order to serve both as scheduler synthesis tool from AADL specifications and SCJ tasksets. We plan to further the front end analysis tools from Java task sets in order to build the input CSDF graphs from program analysis, in the context of a future PhD.

7. New Results

7.1. Polychronous automata

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

[26] *Static checking of safety critical Java annotations*. Tang, D. Plsek, A. and Vitek, J. International Workshop on Java Technologies for Real-Time and Embedded Systems, 2010

We have defined a model of *polychronous automata* based on clock relations [13]. 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 formal model of automata also supports the recommendations adopted by the SAE committee on the AADL to implement a timed and synchronous behavioural annex for the standard [27].

A minimal syntactic extension of the Signal language has been defined to integrate polychronous automata in Polychrony. We have added a new syntactic category of *process*, called `automaton`. In such an automaton process, labeled processes represent states, and generic processes such as `Transition` are used to represent the automaton features. Usual equations can be used in these automaton processes to specify constraints or to define computations.

We have also defined and implemented the refinement of Signal processes as automata. A given Signal program may be seen as an automaton which contains one single state and one single transition, labeled by a clock. This clock is the upper bound of all the clocks of the program (the *tick* of the program). The construction of a refined automaton from a Signal program is based on delayed signals, viewed as state variables (in particular Boolean ones). A state of the automaton is a Signal program with some valuation of its state variables. Transitions are labeled by clocks, which represent the events that fire these transitions. The principle of the construction consists in dividing a given state according to the possible values of a state variable (i.e., *true* and *false* for Boolean state variables) in order to get two states, and thus two new Signal programs. Each one of these two states is obtained using a rewriting of the starting program. Moreover, the absence of value for the state variable (which can be considered as another possible value) is taken into account in the clocks labelling the transitions. The construction of the automaton is a hierarchic process. Thanks to the clock hierarchy, this construction, which would be expensive in the worst case (the size of the explicit automaton being an exponential of its number of state variables), may be heavily simplified.

7.2. 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. When building a virtual prototype of the system, including a combination of simulated hardware, firmware and application software, the executable models can be augmented also with property verification, for example in the PSL language, or simply by introducing assertions in the implementation code.

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. Most of time the failure is due to an unexpected sequence of states and transitions that involve several components mixing hardware and software that could not be checked at the model level (e.g. state explosion) or was simply unforeseen. The engineers then have to investigate the cause of failure.

[27] *Logically timed specification in the AADL: a synchronous model of computation and communication (recommendations to the SAE committee on AADL).* L. Besnard, E. Borde, P. Dissaux, T. Gautier, P. Le Guernic, and J.-P. Talpin. Technical Report RT-0446, Inria, 2014.

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 untractable by human manual handling. The engineers use some kind of search tools to identify the cause of failure and after iterative refinement steps, which are very time consuming, eventually identify the reason, most often some unforeseen causality chain of events and state transitions that lead to a failure. A new system property can then be captured and included into the set of verified properties.

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 are investigating a technique for trace analysis 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.

The first idea is to introduce a formal mapping/filtering technique such that the raw data generated by a virtual prototype can be mapped onto a formal trace model. For that, we propose to use a model transformer whose code is generated from a higher level. Using the Eclipse modelling framework, we propose for the virtual prototyping engineers to first describe using a Domain Specific Language how the raw output of the simulator can be filtered and mapped to a formal model. This Domain Specific Language takes as input the description of the simulator output, and the description of the formal output, following fixed meta models. In current version, the meta model of the virtual prototype dictates that it generates 'trace items' where each trace item is specified as a sequence of identified binary data variables (bits, bytes, words..) that carry a timestamp.

The model transformer generates code (in our case C++) that is dynamically invoked by the virtual prototype to dynamical map the trace output. An advantage of doing that is that all irrelevant data with regards to a tested property can be ignored and the size of trace files can be considerably reduced. For our experiment, we have chosen logical clock CCSL as our formal target formalism. The Eclipse EMF tool we have defined allows users to define a mapping model from the local simulation events from the SimSoC simulator to a logical clock format.

The second idea is to hide the complexity of the formal method formulas into a user friendly property specification language. For example, we do not want to expose the end-users engineers to understand the intricacies of CCSL or LTL. The property specification language is translated into CCSL formulas, which in turn generate automata. It should be possible then, to some extent, to change the formalism underneath the language without changing the properties expressed by the user.

The property specification language ultimately compiles into automata that parse the formal trace output generated above. At runtime of the virtual prototype, the mapping library is dynamically loaded by the simulator and generates input for the automata. The verification of the properties can be dynamic, with a true runtime verification, or statically by analyzing the (much smaller) trace file after a failure.

This year we have investigated this approach, designed the architecture described above and carried some experimental work, but a significant part of the implementation still remains to be done. We have started designing a new property specification language where the users can express properties such as causality (e.g. the train must not start if the door is opened) or jittering or clock drift in image processing [11], [10]. There remain some theoretical issues with regards to which properties can be effectively verified.

7.3. Integration of Polychrony with QGen

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

The FUI project P gave birth to the QGen 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. It is currently not known if an architectural description language is going to be integrated in QGen, as originally planned.

We developed a transformation tool named P2S for expressing P system models in Signal, using the EMF (Eclipse Modelling Framework) technology. P2S tool is written in Clojure, a language inspired by Lisp running on the Java Virtual Machine, which helped us define a terse and expressive API for manipulating Signal models while remaining fully interoperable with existing Java libraries (including Eclipse plugins and especially Polychrony ones).

We experimented this transformation tool on small to medium use cases provided by members of the P project. Our work is detailed in a conference paper titled "Integration of Polychrony and QGen Model Compiler", which will appear at ERTSS'16 [28]. A perspective of our work is to convert the intermediate code emitted by QGen as Signal too (under development), in order to produce a fully executable Signal model of Simulink models, and combine them with architectural description of systems in AADL, and/or P's architecture language.

7.4. Formal semantics and model-based analysis of AADL specifications

Participants: Loïc Besnard, Etienne Borde, Thierry Gautier, Paul Le Guernic, Clément Guy, Jean-Pierre Talpin, Huafeng Yu.

Last year, the SAE committee on the AADL adopted our recommendations to implement a timed and synchronous behavioural annex for the standard. We have defined a new model of polychronous constrained automata that has been provided as semantic model for our proposal of an extension of the AADL behavioural annex. An experimental implementation of the semantic features of this "timing annex" will be provided through the Polychrony framework. For that purpose, representations of automata have been introduced in the Signal toolbox of Polychrony. The implementation will enrich the already existing transformation from AADL models to Signal programs to consider behaviour of AADL models, and will be integrated in the POP environment for Eclipse. The transformation from AADL behaviour annex to Signal programs use the Signal extension for polychronous automata, which are used as the common semantic domain. The implementation is currently tested with the adaptive cruise control case study developed with Toyota ITC.

Our work with the SAE committee is sponsored by Toyota, with whom we started a new project in 2014 jointly 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 [19]. The project led to the elaboration of a case study of an adaptive cruise control system, supported through an AADL implementation and a video of demonstration. The case study implementation is an AADL model representing the whole adaptive cruise control system, from car devices (e.g., brakes, throttle or radar) to software behavior, including embedded hardware (buses, processors and memories). It will be used in the future to demonstrate property and constraint analyses through heterogeneous systems. Huafeng Yu, our main collaborator at Toyota ITC, presented the video of demonstration at the annual Toyota show case. Early returns from the show case express a growing interest of Toyota for architecture and timing of car embedded systems, which could lead to new collaborations.

7.5. Refinement types for reactive system models

Participants: Pierre Jouvelot, Sandeep Shukla, Jean-Pierre Talpin.

We introduced a new technique born from the field of functional programming to adapt and extend it to the case of reaction systems, the notion of refinement types of Jahla et al. [29]. Our idea is to formulate the analysis of algebraic properties in synchronous and reactive programs as data-dependent type properties formulated using multi-sorted logic formulas, which we call liquid clocks [20], [18]. Our objectives are to cover the case of several models of concurrency and computation: synchronous, asynchronous, data-parallel; as well as to formulate such algebraic properties for linear, continuous and logical forms of time, all into the same type-theoretical framework. This work, born from two collaborations With USAF/VT and with the ANR Feever project, will be pursued within the TIX international partnership.

[28] *Integration of polychrony in the QGen model compiler*. C. Junke, T. Gautier, L. Besnard, J.-P. Talpin. ERTS'16 - European Congress on Embeddd Real-Rime Software and Systems, 2016.

[29] *Liquid Types*. P. M. Rondon, M. Kawaguchi, R. Jhala. PLDI, 2008

7.6. Formal verification of timing aspects of cyber-physical systems using a contract theory

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

This is a new project in collaboration with Mitsubishi Electronics Research Centre Europe (MERCE). 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.

8. Bilateral Contracts and Grants with Industry

8.1. Bilateral Contracts with Industry

8.1.1. Toyota Info-Technology Centre (2014+)

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 standardisation of a time annex in the SAE standard AADL.

In a second work-package, we aim at elaborating a standardised 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 synthesised from different physical locations.

9.1.2. Competitivity Clusters

Program: FUI

Project acronym: P

Project title: Project P

Duration: March 2011 - Sept. 2015

Coordinator: Continental Automotive France

Other partners: 19 partners (Airbus, Astrium, Rockwell Collins, Safran, Thales Alenia Space, Thales Avionics...)

URL: http://www.open-do.org/projects/p/

Abstract:

The aim of project P is 1/ to aid industrials to deploy model-driven engineering technology for the development of safety-critical embedded applications, 2/ to contribute on initiatives such as ITEA2 OPEES and Artemisia CESAR to develop support for tools inter-operability, and 3/ to provide state-of-the-art automated code generation techniques from multiple, heterogeneous, system-levels models. The focus of project P is the development of a code generation toolchain starting from domain-specific modeling languages for embedded software design and to deliver the outcome of this development as an open-source distribution, in the aim of gaining an impact similar to GCC for general-purpose programming, as well as a kit to aid with the qualification of that code generation toolchain.

The contribution of project-team TEA in project P is to bring the necessary open-source technology of the Polychrony environment to allow for the synthesis of symbolic schedulers for software architectures modeled with P in a manner ensuring global asynchronous deterministic execution..

9.1.3. 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...

URL: http://www.corac-ame.com/

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 modelling 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 [30] 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 [31] [32], 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.

This year, we published a joint paper at the 52nd. Digital Automation Conference in San Francisco [19].

[30] ACM-IEEE MEMOCODE conference series

[31] *Formal methods and models for system design*, R. Gupta, S. Shukla, J.-P. Talpin, Eds. ISBN 1-4020-8051-4. Springer, 2004.

[32] *Synthesis of embedded systems*. S. Shukla, J.-P. Talpin, Eds. ISBN 978-1-4419-6399-4. Springer, 2010

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 modelling and virtual prototyping in cyber-physical systems.

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 stabilisation and synchronisation 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 modelling 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 to International Teams

9.3.1.1. Research stays abroad

Jean-Pierre Talpin was awarded a visiting researcher grant by USAF OSR in 2014. In this context, he visited the Arlington and Falls Church VT campuses in Spring, Summer of 2015, and UC San Diego in Autumn 2015.

Thierry Gautier was invited to visit NUAA (Nanjing University of Aeronautics and Astronautics), Nanjing, China, in September 2015.

10. Dissemination

10.1. Promoting Scientific Activities

10.1.1. Scientific events organisation

10.1.1.1. Member of the organizing committees

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. Chair of conference program committees

Jean-Pierre Talpin co-chaired the AVICPS'15 workshop at RTSS'15

10.1.2.2. Member of the conference program committees

Jean-Pierre Talpin served the program committee of:

- MEMOCODE'15, 13th ACM-IEEE International Conference on Formal Methods and Models for System Design
- ACVI'15, 2nd Workshop Architecture Centric Virtual Integration
- AVICPS'15, 5th Analytic Virtual Integration of Cyber-Physical Systems Workshop
- FACS'15, 12th International Conference on Formal Aspects of Component Software
- FTSCS'15, 8th International Workshop on Formal Techniques for Safety-Critical Systems
- ICESS'15, 12th International Conference on Embedded Software and Systems
- SCOPES'15, 18th International Workshop on Software and Compilers for Embedded Systems
- SAC'15, 30th ACM/SIGAPP Symposium on Applied Computing

Thierry Gautier served as program committee member for the 2015 Electronic System Level Synthesis Conference, ESLsyn 2015 (http://ecsi.org//eslsyn).

10.1.2.3. reviewer

Thierry Gautier reviewed for ICCAD 2015 and MEMOCODE 2015.

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).

Jean-Pierre Talpin was Guest Editor of two special issues of the ACM TECS on *Architecture-Centric Virtual Integration* and on *Formal Methods for System Design*.

10.1.3.2. reviewer

Jean-Pierre Talpin reviewed articles for Acta Informatica.

Thierry Gautier reviewed for Frontiers of Computer Science.

10.1.4. Invited talks

Jean-Pierre Talpin gave a lecture on "Sémantique formelle de modèles d'architecture" at the Ecole temps-réel 2015 in Rennes

10.1.5. Scientific expertise

Jean-Pierre Talpin is co-author with Huafeng Yu and Sandeep Shukla of two patents filed with the USPTO:

- T1834.10134US01 – A timing-oriented and architecture-centric system design using contracts
- T1834.10131US01 – Bottom-up approach for integrating models for software components using contracts

10.2. Teaching - Supervision - Juries

10.2.1. Supervision

- Jean-Pierre Talpin is the supervisor of Simon Lunel's thesis on "*Timed contract algebras for correct by construction real-time system design*" and co-supervisor of Imré Frotier de la Mésselière with Mines ParisTech until June 2015.

10.2.2. Juries

Jean-Pierre Talpin served as Referee (rapporteur) for the HDR Thesis defense of Claire Paggetti , entitled "*Programmation sûre de plates-formes embarquées de type multi/pluri-cœurs*".

Jean-Pierre Talpin served as Referee (rapporteur) for the PhD. Thesis Defence of

- Elie Richa, Telecom ParisTech, entitled "*Qualification of Source Code Generators in the Avionics Domain : Automated Testing of Model Transformation Chains*".
- Ahlem Triki, Université de Grenoble , entitled "*Distributed Implementations of Timed Component-based Systems*".
- Yu Bai, TU Kaiserslautern , entitled "*Model-based Design of Embedded Systems by Desynchronization*".

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] 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

[8] 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

[9] 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

Publications of the year

Articles in International Peer-Reviewed Journals

[10] V. JOLOBOFF, S. WANG, Y. DENG. *Fast approximately timed simulation*, in "WIT Transactions on Information and Communication Technologies", March 2015, vol. 978-1-78466-054-3, n^{o} 68, 756 p. , https://hal.archives-ouvertes.fr/hal-01081104

International Conferences with Proceedings

[11] V. JOLOBOFF, J.-F. MONIN, X. SHI. *Towards Verified Faithful Simulation*, in "Dependable Software Engineering: Theories, Tools, and Applications", Nanjing, China, S. VERLAG (editor), November 2015, https://hal.inria.fr/hal-01242963

[12] C. JUNKE, T. GAUTIER, L. BESNARD, J.-P. TALPIN. *Integration of polychrony in the 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

[13] *Best Paper*
P. LE GUERNIC, T. GAUTIER, J.-P. TALPIN, L. BESNARD. *Polychronous Automata*, in "TASE 2015, 9th International Symposium on Theoretical Aspects of Software Engineering", Nanjing, China, IEEE Computer Society, September 2015, pp. 95-102 [*DOI :* 10.1109/TASE.2015.21], https://hal.archives-ouvertes.fr/hal-01240440.

[14] V. C. NGO, J.-P. TALPIN, T. GAUTIER, L. BESNARD, P. LE GUERNIC. *Modular translation validation of a full-sized synchronous compiler using off-the-shelf verification tools (abstract)*, in "International Workshop on Software and Compilers for Embedded Systems", St Goar, Germany, ACM, June 2015, https://hal.inria.fr/hal-01148919

[15] V. C. NGO, J.-P. TALPIN, T. GAUTIER, P. LE GUERNIC. *Translation Validation for Clock Transformations in a Synchronous Compiler*, in "FASE - ETAPS 2015", London, United Kingdom, Springer, April 2015, https://hal.inria.fr/hal-01087795

[16] V. C. NGO, J.-P. TALPIN, T. GAUTIER. *Translation Validation for Synchronous Data-flow Specification in the SIGNAL Compiler*, in "International Conference on Formal Techniques for Distributed Objects, Components and Systems", Grenoble, France, Formal Techniques for Distributed Objects, Components, and Systems,

Springer, June 2015, vol. 9039, pp. 66-80 [*DOI :* 10.1007/978-3-319-19195-9_5], https://hal.inria.fr/hal-01148901

[17] J. PRASHI, S. KUMAR SHUKLA, J.-P. TALPIN, H. YU. *Mapping Functional Behavior onto Architectural Model in a Model Driven Embedded System Design*, in "Symposium On Applied Computing", Salamanca, Spain, April 2015, https://hal.inria.fr/hal-01148908

[18] J.-P. TALPIN, P. JOUVELOT, S. KUMAR SHUKLA. *Towards refinement types for time-dependent data-flow networks*, in "ACM-IEEE Conference on Methods and Models for System Design", Austin, United States, I. C. SOCIETY (editor), September 2015, https://hal.inria.fr/hal-01241806

[19] 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

Research Reports

[20] J.-P. TALPIN, P. JOUVELOT, S. KUMAR SHUKLA. *Liquid Clocks - Refinement Types for Time-Dependent Stream Functions*, Inria Rennes - Bretagne Atlantique ; Inria, June 2015, n^{o} RR-8747, https://hal.inria.fr/hal-01166350