IN PARTNERSHIP WITH:
Institut polytechnique de Grenoble

Activity Report 2016

Project-Team SPADES

Sound Programming of Adaptive Dependable Embedded Systems

IN COLLABORATION WITH: Laboratoire d'Informatique de Grenoble (LIG)

RESEARCH CENTER
Grenoble - Rhône-Alpes

THEME
Embedded and Real-time Systems

Table of contents

Project-Team SPADES

Creation of the Team: 2013 January 01, updated into Project-Team: 2015 July 01

Keywords:

Computer Science and Digital Science:

1.1.1. - Multicore
1.1.9. - Fault tolerant systems
1.3. - Distributed Systems
2.1.1. - Semantics of programming languages
2.1.6. - Concurrent programming
2.1.8. - Synchronous languages
2.3. - Embedded and cyber-physical systems
2.3.1. - Embedded systems
2.3.2. - Cyber-physical systems
2.3.3. - Real-time systems
2.4.1. - Analysis
2.4.3. - Proofs
2.5.2. - Component-based Design

Other Research Topics and Application Domains:

6.6. - Embedded systems

1. Members

Research Scientists

Alain Girault [Team leader, Inria, Senior Researcher, HDR]
Pascal Fradet [Inria, Researcher, HDR]
Gregor Goessler [Inria, Researcher, HDR]
Sophie Quinton [Inria, Researcher]
Jean-Bernard Stefani [Inria, Senior Researcher]

Faculty Member

Xavier Nicollin [Grenoble INP, Associate Professor]

PhD Students

Yoann Geoffroy [Inria, until Dec. 2016]
Xiaojie Guo [UGA & PERSYVAL-Lab, from Dec. 2016]
Stephan Plassart [UGA & PERSYVAL-Lab, from Sep. 2016]
Christophe Prévot [Thales, granted by CIFRE]

Post-Doctoral Fellow

Lijun Shan [Inria, from Nov. 2016]

Visiting Scientists

Athena Abdi [Amirkabir U., until Jul. 2016]
Leonie Ahrendts [TU Braunschweig, Jan. and Jun. 2016]
Ismail Assayad [Casablanca U., Sep. 2016]
Zain Hammadeh [TU Braunschweig, Aug. 2016]
Eugene Yip [Bamberg U., Oct. 2016]

Hamid Zarandi [Amirkabir U., Jul. 2016]

Administrative Assistant

Helen Pouchot-Rouge-Blanc [Inria]

Others

Aurelie Kong Win Chang [Université Claude Bernard Lyon 1, Master Student, from Feb. 2016 until Jul. 2016]
Lina Marsso [UGA, Master Student, from Feb. 2016 until Jul. 2016]
Baptiste Pollien [UGA, L1 Polytech Internship, from Jun. 2016 until Jul. 2016]
Martin Vassor [EPFL, Summer Internship, from Jun. 2016 until Sep. 2016]

2. Overall Objectives

2.1. Overall Objectives

The SPADES project-team aims at contributing to meet the challenge of designing and programming dependable embedded systems in an increasingly distributed and dynamic context. Specifically, by exploiting formal methods and techniques, SPADES aims to answer three key questions:

1. How to program open networked embedded systems as dynamic adaptive modular structures?
2. How to program reactive systems with real-time and resource constraints on multicore architectures?
3. How to program reliable, fault-tolerant embedded systems with different levels of criticality?

These questions above are not new, but answering them in the context of modern embedded systems, which are increasingly distributed, open and dynamic in nature [31], makes them more pressing and more difficult to address: the targeted system properties – dynamic modularity, time-predictability, energy efficiency, and fault-tolerance – are largely antagonistic (*e.g.*, having a highly dynamic software structure is at variance with ensuring that resource and behavioral constraints are met). Tackling these questions together is crucial to address this antagonism, and constitutes a key point of the SPADES research program.

A few remarks are in order:

- We consider these questions to be central in the construction of future embedded systems, dealing as they are with, roughly, software architecture and the provision of real-time and fault-tolerance guarantees. Building a safety-critical embedded system cannot avoid dealing with these three concerns.
- The three questions above are highly connected. For instance, composability along time, resource consumption and reliability dimensions are key to the success of a component-based approach to embedded systems construction.
- For us, “Programming” means any constructive process to build a running system. It can encompass traditional programming as well as high-level design or “model-based engineering” activities, provided that the latter are supported by effective compiling tools to produce a running system.
- We aim to provide semantically sound programming tools for embedded systems. This translates into an emphasis on formal methods and tools for the development of provably dependable systems.

3. Research Program

3.1. Introduction

The SPADES research program is organized around three main themes, *Components and contracts*, *Real-time multicore programming*, and *Language-based fault tolerance*, that seek to answer the three key questions identified in Section 2.1. We plan to do so by developing and/or building on programming languages and techniques based on formal methods and formal semantics (hence the use of *“sound programming”* in the project-team title). In particular, we seek to support design where correctness is obtained by construction, relying on proven tools and verified constructs, with programming languages and programming abstractions designed with verification in mind.

3.2. Components and Contracts

Component-based construction has long been advocated as a key approach to the "correct-by-construction" design of complex embedded systems [65]. Witness component-based toolsets such as UC Berkeley's PTOLEMY [53], Verimag's BIP [36], or the modular architecture frameworks used, for instance, in the automotive industry (AUTOSAR) [28]. For building large, complex systems, a key feature of component-based construction is the ability to associate with components a set of *contracts*, which can be understood as rich behavioral types that can be composed and verified to guarantee a component assemblage will meet desired properties. The goal in this theme is to study the formal foundations of the component-based construction of embedded systems, to develop component and contract theories dealing with real-time, reliability and fault-tolerance aspects of components, and to develop proof-assistant-based tools for the computer-aided design and verification of component-based systems.

Formal models for component-based design are an active area of research (see *e.g.*, [29], [30]). However, we are still missing a comprehensive formal model and its associated behavioral theory able to deal *at the same time* with different forms of composition, dynamic component structures, and quantitative constraints (such as timing, fault-tolerance, or energy consumption). Notions of contracts and interface theories have been proposed to support modular and compositional design of correct-by-construction embedded systems (see *e.g.*, [40], [41] and the references therein), but having a comprehensive theory of contracts that deals with all the above aspects is still an open question [71]. In particular, it is not clear how to accomodate different forms of composition, reliability and fault-tolerance aspects, or to deal with evolving component structures in a theory of contracts.

Dealing in the same component theory with heterogeneous forms of composition, different quantitative aspects, and dynamic configurations, requires to consider together the three elements that comprise a component model: behavior, structure and types. *Behavior* refers to behavioral (interaction and execution) models that characterize the behavior of components and component assemblages (*e.g.*, transition systems and their multiple variants – timed, stochastic, etc.). *Structure* refers to the organization of component assemblages or configurations, and the composition operators they involve. *Types* refer to properties or contracts that can be attached to components and component interfaces to facilitate separate development and ensure the correctness of component configurations with respect to certain properties. Taking into account dynamicity requires to establish an explicit link between behavior and structure, as well as to consider higher-order systems, both of which have a direct impact on types.

We plan to develop our component theory by progressing on two fronts: component calculi, and semantical framework. The work on typed component calculi aims to elicit process calculi that capture the main insights of component-based design and programming and that can serve as a bridge towards actual architecture description and programming language developments. The work on the semantical framework should, in the longer term, provide abstract mathematical models for the more operational and linguistic analysis afforded by component calculi. Our work on component theory will find its application in the development of a COQ-based toolchain for the certified design and construction of dependable embedded systems, which constitutes our third main objective for this axis.

3.3. Real-Time Multicore Programming

Programming real-time systems (*i.e.*, systems whose correct behavior depends on meeting timing constraints) requires appropriate languages (as exemplified by the family of synchronous languages [39]), but also the support of efficient scheduling policies, execution time and schedulability analyses to guarantee real-time constraints (*e.g.*, deadlines) while making the most effective use of available (processing, memory, or networking) resources. Schedulability analysis involves analyzing the worst-case behavior of real-time tasks under a given scheduling algorithm and is crucial to guarantee that time constraints are met in any possible execution of the system. Reactive programming and real-time scheduling and schedulability for multiprocessor systems are old subjects, but they are nowhere as mature as their uniprocessor counterparts, and still feature a number of open research questions [35], [48], in particular in relation with mixed criticality systems. The main goal in this theme is to address several of these open questions.

We intend to focus on two issues: multicriteria scheduling on multiprocessors, and schedulability analysis for real-time multiprocessor systems. Beyond real-time aspects, multiprocessor environments, and multicore ones in particular, are subject to several constraints *in conjunction*, typically involving real-time, reliability and energy-efficiency constraints, making the scheduling problem more complex for both the offline and the online cases. Schedulability analysis for multiprocessor systems, in particular for systems with mixed criticality tasks, is still very much an open research area.

Distributed reactive programming is rightly singled out as a major open issue in the recent, but heavily biased (it essentially ignores recent research in synchronous and dataflow programming), survey by Bainomugisha et al. [35]. For our part, we intend to focus on two questions: devising synchronous programming languages for distributed systems and precision-timed architectures, and devising dataflow languages for multiprocessors supporting dynamicity and parametricity while enjoying effective analyses for meeting real-time, resource and energy constraints in conjunction.

3.4. Language-Based Fault Tolerance

Tolerating faults is a clear and present necessity in networked embedded systems. At the hardware level, modern multicore architectures are manufactured using inherently unreliable technologies [43], [58]. The evolution of embedded systems towards increasingly distributed architectures highlighted in the introductory section means that dealing with partial failures, as in Web-based distributed systems, becomes an important issue. While fault-tolerance is an old and much researched topic, several important questions remain open: automation of fault-tolerance provision, composable abstractions for fault-tolerance, fault diagnosis, and fault isolation.

The first question is related to the old question of "system structure for fault-tolerance" as originally discussed by Randell for software fault tolerance [77], and concerns in part our ability to clearly separate fault-tolerance aspects from the design and programming of purely "functional" aspects of an application. The classical arguments in favor of a clear separation of fault-tolerance concerns from application code revolve around reduced code and maintenance complexity [49]. The second question concerns the definition of appropriate abstractions for the modular construction of fault-tolerant embedded systems. The current set of techniques available for building such systems spans a wide range, including exception handling facilities, transaction management schemes, rollback/recovery schemes, and replication protocols. Unfortunately, these different techniques do not necessarily compose well – for instance, combining exception handling and transactions is non trivial, witness the flurry of recent work on the topic, see *e.g.*, [64] and the references therein –, they have no common semantical basis, and they suffer from limited programming language support. The third question concerns the identification of causes for faulty behavior in component-based assemblages. It is directly related to the much researched area of fault diagnosis, fault detection and isolation [66].

We intend to address these questions by leveraging programming language techniques (programming constructs, formal semantics, static analyses, program transformations) with the goal to achieve provable fault-tolerance, *i.e.*, the construction of systems whose fault-tolerance can be formally ensured using verification tools and proof assistants. We aim in this axis to address some of the issues raised by the above open questions by using aspect-oriented programming techniques and program transformations to automate the inclusion of fault-tolerance in systems (software as well as hardware), by exploiting reversible programming models to investigate composable recovery abstractions, and by leveraging causality analyses to study fault-ascription in component-based systems. Compared to the huge literature on fault-tolerance in general, in particular in the systems area (see *e.g.*, [59] for an interesting but not so recent survey), we find by comparison much less work exploiting formal language techniques and tools to achieve or support fault-tolerance. The works reported in [42], [44], [47], [54], [67], [76], [81] provide a representative sample of recent such works.

A common theme in this axis is the use and exploitation of causality information. Causality, *i.e.*, the logical dependence of an effect on a cause, has long been studied in disciplines such as philosophy [72], natural sciences, law [73], and statistics [74], but it has only recently emerged as an important focus of research in computer science. The analysis of logical causality has applications in many areas of computer science. For instance, tracking and analyzing logical causality between events in the execution of a concurrent system is

required to ensure reversibility [70], to allow the diagnosis of faults in a complex concurrent system [61], or to enforce accountability [69], that is, designing systems in such a way that it can be determined without ambiguity whether a required safety or security property has been violated, and why. More generally, the goal of fault-tolerance can be understood as being to prevent certain causal chains from occurring by designing systems such that each causal chain either has its premises outside of the fault model (*e.g.*, by introducing redundancy [59]), or is broken (*e.g.*, by limiting fault propagation [78]).

4. Application Domains

4.1. Industrial Applications

Our applications are in the embedded system area, typically: transportation, energy production, robotics, telecommunications, systems on chip (SoC). In some areas, safety is critical, and motivates the investment in formal methods and techniques for design. But even in less critical contexts, like telecommunications and multimedia, these techniques can be beneficial in improving the efficiency and the quality of designs, as well as the cost of the programming and the validation processes.

Industrial acceptance of formal techniques, as well as their deployment, goes necessarily through their usability by specialists of the application domain, rather than of the formal techniques themselves. Hence, we are looking to propose domain-specific (but generic) realistic models, validated through experience (*e.g.*, control tasks systems), based on formal techniques with a high degree of automation (*e.g.*, synchronous models), and tailored for concrete functionalities (*e.g.*, code generation).

4.2. Industrial Design Tools

The commercially available design tools (such as UML with real-time extensions, MATLAB/ SIMULINK/ dSPACE [1]) and execution platforms (OS such as VXWORKS, QNX, real-time versions of LINUX ...) start now to provide besides their core functionalities design or verification methods. Some of them, founded on models of reactive systems, come close to tools with a formal basis, such as for example STATEMATE by ILOGIX.

Regarding the synchronous approach, commercial tools are available: SCADE [2] (based on LUSTRE), CONTROLBUILD and RT-BUILDER (based on SIGNAL) from GEENSYS [3] (part of DASSAULT SYSTEMES), specialized environments like CELLCONTROL for industrial automatism (by the INRIA spin-off ATHYS– now part of DASSAULT SYSTEMES). One can observe that behind the variety of actors, there is a real consistency of the synchronous technology, which makes sure that the results of our work related to the synchronous approach are not restricted to some language due to compatibility issues.

4.3. Current Industrial Cooperations

Regarding applications and case studies with industrial end-users of our techniques, we cooperate with Thales on schedulability analysis for evolving or underspecified real-time embedded systems, with Orange Labs on software architecture for cloud services and with Daimler on reduction of nondeterminism and analysis of deadline miss models for the design of automotive systems.

5. New Software and Platforms

5.1. pyCPA_TWCA: A pyCPA plugin for computing deadline miss models

FUNCTIONAL DESCRIPTION

[1] http://www.dspaceinc.com
[2] http://www.esterel-technologies.com
[3] http://www.geensoft.com

We are developing pyCPA_TWCA, a pyCPA plugin for Typical Worst-Case Analysis as described in Section 6.2.5. pyCPA is an open-source Python implementation of Compositional Performance Analysis developed at TU Braunschweig, which allows in particular response-time analysis. pyCPA_TWCA is an extension of this tool that is co-developed by Sophie Quinton, Zain Hammadeh (TU Braunschweig) and Leonie Ahrendts (TU Braunschweig). It allows in particular the computation of weakly-hard guarantees for real-time tasks, *i.e.*, the number of deadline misses out of a sequence of executions. This year, pyCPA_TWCA has been extended to task chains but remains limited to uniprocessor systems, scheduled according to static priority scheduling. A public release is planned but has not yet taken place.

- Authors: Zain Hammadeh and Leonie Ahrendts and Sophie Quinton.
- Contact: Sophie Quinton.

6. New Results

6.1. Components and contracts

Participants: Alain Girault, Christophe Prévot, Sophie Quinton, Jean-Bernard Stefani.

6.1.1. Contracts for the negotiation of embedded software updates

We address the issue of change after deployment in safety-critical embedded system applications in collaboration with Thales and also in the context of the CCC project (http://ccc-project.org/).

The goal of CCC is to substitute lab-based verification with in-field formal analysis to determine whether an update may be safely applied. This is challenging because it requires an automated process able to handle multiple viewpoints such as functional correctness, timing, etc. For this purpose, we propose an original methodology for contract-based negotiation of software updates. The use of contracts allows us to cleanly split the verification effort between the lab and the field. In addition, we show how to rely on existing viewpoint-specific methods for update negotiation. We have validated our approach on a concrete example inspired by the automotive domain in collaboration with our German partners from TU Braunschweig [19].

In collaboration with Thales we mostly focus on timing aspects with the objective to anticipate at design time future software evolutions and identify potential schedulability bottlenecks. This year we have presented an approach to quantify the flexibility of a system with respect to timing. In particular we have shown that it is possible under certain conditions to identify the task that will directly induce the limitations on a possible software update. If performed at design time, such a result can be used to adjust the system design by giving more slack to the limiting task [21].

6.1.2. Location graphs

The design of configurable systems can be streamlined and made more systematic by adopting a component-based structure, as demonstrated with the FRACTAL component model [2]. However, the formal foundations for configurable component-based systems, featuring higher-order capabilities where components can be dynamically instantiated and passivated, and non-hierarchical structures where components can be contained in different composites at the same time, are still an open topic. We have recently introduced the location graph model [79], where components are understood as graphs of locations hosting higher-order processes, and where component structures can be arbitrary graphs.

We have continued the development of location graphs, revisiting the underlying structural model (hypergraphs instead of graphs), and simplifying its operational semantics while preserving the model expressivity. Towards the development of a behavioral theory of location graphs, we have defined different notions of bisimilarity for location graphs and shown them to be congruences, although a fully fledged co-inductive characterization of contextual equivalence for location graphs is still in the works. This work has not yet been published.

6.2. Real-Time multicore programming

Participants: Pascal Fradet, Alain Girault, Gregor Goessler, Xavier Nicollin, Sophie Quinton.

6.2.1. Time predictable programming languages

Time predictability (PRET) is a topic that emerged in 2007 as a solution to the ever increasing unpredictability of today's embedded processors, which results from features such as multi-level caches or deep pipelines [52]. For many real-time systems, it is mandatory to compute a strict bound on the program's execution time. Yet, in general, computing a tight bound is extremely difficult [82]. The rationale of PRET is to simplify both the programming language and the execution platform to allow more precise execution times to be easily computed [34].

Following our past results on the PRET-C programming language [32], we have proposed a time predictable synchronous programming language for multicores, called FOREC. It extends C with a small set of ESTEREL-like synchronous primitives to express concurrency, interaction with the environment, looping, and a synchronization barrier [83] (like the pause statement in ESTEREL). FOREC threads communicate with each other via shared variables, the values of which are *combined* at the end of each tick to maintain deterministic execution. We provide several deterministic combine policies for shared variables, in a way similar as concurrent revisions [45]. Thanks to this, it benefits from a deterministic semantics. FOREC is compiled into threads that are then statically scheduled for a target multicore chip. Our WCET analysis takes into account the access to the shared TDMA bus and the necessary administration for the shared variables. We achieve a very precise WCET (the over-approximation being less than 2%) thanks to a reachable space exploration of the threads' states [15]. We have published a research report presenting the complete semantics and the compiler [27], and submitted it to a journal.

Furthermore, we have extended the PRET-C compiler [32] in order to make it energy aware. To achieve this, we use dynamic voltage and frequency scaling (DFVS) and we insert DVFS control points in the control flow graph of the PRET-C program. The difficulty is twofold: first the control flow graph is concurrent, and second resulting optimization problem is in the 2D space (time,energy). Thanks to a novel ILP formulation and to a bicriteria heuristic, we are able to address the two objectives jointly and to compute, for each PRET-C program, the Pareto front of the non-dominated solutions in the 2D space (time, energy) [20].

This is a collaboration with Eugene Yip from Bamberg University, and with Partha Roop and Jiajie Wang from the University of Auckland.

6.2.2. Modular distribution of synchronous programs

Synchronous programming languages describe functionally centralized systems, where every value, input, output, or function is always directly available for every operation. However, most embedded systems are nowadays composed of several computing resources. The aim of this work is to provide a language-oriented solution to describe *functionally distributed reactive systems*. This research started within the Inria large scale action SYNCHRONICS and is a joint work with Marc Pouzet (ENS, PARKAS team from Rocquencourt) and Gwenaël Delaval (UGA, CTRL-A team from Grenoble).

We are working on defining a *fully-conservative* extension of a synchronous data-flow programming language (the HEPTAGON language, inspired from LUCID SYNCHRONE [46]). The extension, by means of *annotations* adds *abstract location parameters* to functions, and *communications* of values between locations. At deployment, every abstract location is assigned an actual one; this yields an executable for each actual computing resource. Compared to the PhD of Gwenaël Delaval [50], [51], the goal here is to achieve *modular* distribution even in the presence of non-static clocks, *i.e.*, clocks defined according to the value of inputs.

By *fully-conservative*, we have three aims in mind:

1. A non-annotated (*i.e.*, centralized) program will be compiled exactly as before;
2. An annotated program eventually deployed onto only one computing location will behave exactly as its centralized couterpart;
3. The input-output semantics of a distributed program is the same as its centralized counterpart.

By *modular*, we mean that we want to compile each function of the program into a single function capable of running on any computing location. At deployment, the program of each location may be optimized (by simple Boolean-constant-propagation, dead-code and unused-variable elimination), yielding different optimized code for each computing location.

We have formalized the type-system for inferring the location of each variable and computation. In the presence of local clocks, added information is computed from the existing clock-calculus and the location-calculus, to infer necessary communication of clocks between location. All pending theorical and technical issues have been answered, and the new compiler is being implemented, with new algorithms for deployment (and code optimization), achieving the three aims detailed above.

6.2.3. Parametric dataflow models

Recent data-flow programming environments support applications whose behavior is characterized by dynamic variations in resource requirements. The high expressive power of the underlying models (*e.g.*, Kahn Process Networks or the CAL actor language) makes it challenging to ensure predictable behavior. In particular, checking *liveness* (*i.e.*, no part of the system will deadlock) and *boundedness* (*i.e.*, the system can be executed in finite memory) is known to be hard or even undecidable for such models. This situation is troublesome for the design of high-quality embedded systems.

Recently, we have introduced the *Schedulable Parametric Data-Flow* (SPDF) MoC for dynamic streaming applications [55], which extends the standard dataflow model by allowing rates to be parametric, and the *Boolean Parametric Data Flow* (BPDF) MoC [38], [37] which combines integer parameters (to express dynamic rates) and boolean parameters (to express the activation and deactivation of communication channels). In the past years, several other parametric dataflow MoCs have been presented. All these models aim at providing an interesting trade-off between analyzability and expressiveness. They offer a controlled form of dynamism under the form of parameters (*e.g.*, parametric rates), along with run-time parameter configuration.

We have written a survey which provides a comprehensive description of the existing parametric dataflow MoCs (constructs, constraints, properties, static analyses) and compares them using a common example [11]. The main objectives are to help designers of streaming applications to choose the most suitable model for their needs and to pave the way for the design of new parametric MoCs.

We have also studied *symbolic* analyses of data-flow graphs [24], [16], [17], [12]. Symbolic analyses express the system performance as a function of parameters (*i.e.*, input and output rates, execution times). Such functions can be quickly evaluated for each different configuration or checked *w.r.t.* different quality-of-service requirements. These analyses are useful for parametric MoCs, partially specified graphs, and even for completely static SDF graphs. We provide symbolic analyses for computing the maximal throughput of acyclic synchronous dataflow graphs, the minimum required buffers for which as soon as possible (asap) scheduling achieves this throughput, and finally the corresponding input-output latency of the graph. We first investigate these problems for a single parametric edge. The results are then extended to general acyclic graphs using linear approximation techniques. We assess the proposed analyses experimentally on both synthetic and real benchmarks.

6.2.4. Synthesis of switching controllers using approximately bisimilar multiscale abstractions

The use of discrete abstractions for continuous dynamics has become standard in hybrid systems design (see *e.g.*, [80] and the references therein). The main advantage of this approach is that it offers the possibility to leverage controller synthesis techniques developed in the areas of supervisory control of discrete-event systems [75]. The first attempts to compute discrete abstractions for hybrid systems were based on traditional systems behavioral relationships such as simulation or bisimulation, initially proposed for discrete systems most notably in the area of formal methods. These notions require inclusion or equivalence of observed behaviors which is often too restrictive when dealing with systems observed over metric spaces. For such systems, a more natural abstraction requirement is to ask for closeness of observed behaviors. This leads to the notions of approximate simulation and bisimulation introduced in [56].

These approaches are based on sampling of time and space where the sampling parameters must satisfy some relation in order to obtain abstractions of a prescribed precision. In particular, the smaller the time sampling parameter, the finer the lattice used for approximating the state-space; this may result in abstractions with a very large number of states when the sampling period is small. However, there are a number of applications where sampling has to be fast; though this is generally necessary only on a small part of the state-space. We have been exploring two approaches to overcome this state-space explosion [5].

We are currently investigating an approach using mode sequences of given length as symbolic states for our abstractions. By using mode sequences of variable length we are able to adapt the granularity of our abstraction to the dynamics of the system, so as to automatically trade off precision against controllability of the abstract states.

6.2.5. Schedulability of weakly-hard real-time systems

We focus on the problem of computing tight deadline miss models for real-time systems, which bound the number of potential deadline misses in a given sequence of activations of a task. In practical applications, such guarantees are often sufficient because many systems are in fact not hard real-time [4].

Our major contribution this year is the extension of our method for computing deadline miss models, called Typical Worst-Case Analysis (TWCA), to systems with task dependencies. This allows us to provide bounds on deadline misses for systems which until now could not be analyzed [18].

In parallel, we have developed an extension of sensitivity analysis for budgeting in the design of weakly-hard real-time systems. During design, it often happens that some parts of a task set are fully specified while other parameters, e.g. regarding recovery or monitoring tasks, will be available only much later. In such cases, sensitivity analysis can help anticipate how these missing parameters can influence the behavior of the whole system so that a resource budget can be allocated to them. We have developed an extension of sensitivity analysis for deriving task budgets for systems with hard and weakly-hard requirements. This approach has been validated on synthetic test cases and a realistic case study given by our partner Thales. This work will be submitted soon.

Finally, in collaboration with TU Braunschweig and Daimler we have investigated the use of TWCA in conjunction with the Logical Execution Time paradigm [68] according to which data are read and written at predefined time instants. In particular, we have extended TWCA to different deadline miss handling strategies. This work has not been published yet.

6.3. Language Based Fault-Tolerance

Participants: Pascal Fradet, Alain Girault, Yoann Geoffroy, Gregor Goessler, Jean-Bernard Stefani, Martin Vassor, Athena Abdi.

6.3.1. Fault Ascription in Concurrent Systems

The failure of one component may entail a cascade of failures in other components; several components may also fail independently. In such cases, elucidating the exact scenario that led to the failure is a complex and tedious task that requires significant expertise.

The notion of causality *(did an event e cause an event e'?)* has been studied in many disciplines, including philosophy, logic, statistics, and law. The definitions of causality studied in these disciplines usually amount to variants of the counterfactual test "e is a cause of e' if both e and e' have occurred, and in a world that is as close as possible to the actual world but where e does not occur, e' does not occur either". In computer science, almost all definitions of logical causality — including the landmark definition of [63] and its derivatives — rely on a causal model that may not be known, for instance in presence of black-box components. For such systems, we have been developing a framework for blaming that helps us establish the causal relationship between component failures and system failures, given an observed system execution trace. The analysis is based on a formalization of counterfactual reasoning [7].

In his PhD thesis, Yoann Geoffroy proposed a generalization of our fault ascription technique to systems composed of black-box and white-box components. For the latter a faithful behavioral model is given but no specification. The approach leverages results from game theory and discrete controller synthesis to define several notions of causality.

We are currently working on an instantiation of our general semantic framework for fault ascription in [60] to acyclic models of computation, in order to compare our approach with the standard definition of *actual causality* proposed by Halpern and Pearl.

6.3.2. Tradeoff exploration between energy consumption and execution time

We have continued our work on multi-criteria scheduling, in two directions. First, in the context of dynamic applications that are launched and terminated on an embedded homogeneous multi-core chip, under execution time and energy consumption constraints, we have proposed a two layer adaptive scheduling method. In the first layer, each application (represented as a DAG of tasks) is scheduled statically on subsets of cores: 2 cores, 3 cores, 4 cores, and so on. For each size of these sets (2, 3, 4, ...), there may be only one topology or several topologies. For instance, for 2 or 3 cores there is only one topology (a "line"), while for 4 cores there are three distinct topologies ("line", "square", and "T shape"). Moreover, for each topology, we generate statically several schedules, each one subject to a different total energy consumption constraint, and consequently with a different Worst-Case Reaction Time (WCRT). Coping with the energy consumption constraints is achieved thanks to Dynamic Frequency and Voltage Scaling (DVFS). In the second layer, we use these pre-generated static schedules to reconfigure dynamically the applications running on the multi-core each time a new application is launched or an existing one is stopped. The goal of the second layer is to perform a dynamic global optimization of the configuration, such that each running application meets a pre-defined quality-of-service constraint (translated into an upper bound on its WCRT) and such that the total energy consumption be minimized. For this, we *(i)* allocate a sufficient number of cores to each active application, *(ii)* allocate the unassigned cores to the applications yielding the largest gain in energy, and *(iii)* choose for each application the best topology for its subset of cores (*i.e.*, better than the by default "line" topology). This is a joint work with Ismail Assayad (U. Casablanca, Morocco) who visited the team in September 2015.

Second, in the context of a static application (again represented a DAG of tasks) running on an homogeneous multi-core chip, we have worked on the static scheduling minimizing the WCRT of the application under the multiple constraints that the reliability, the power consumption, and the temperature remain below some given thresholds. There are multiple difficulties: *(i)* the reliability is not an invariant measure w.r.t. time, which makes it impossible to use backtrack-free scheduling algorithms such as list scheduling [33]; to overcome this, we adopt instead the Global System Failure Rate (GSFR) as a measure of the system's reliability, which is invariant with time [57]; *(ii)* keeping the power consumption under a given threshold requires to lower the voltage and frequency, but this has a negative impact both on the WCRT and on the GSFR; keeping the GSFR below a given threshold requires to replicate the tasks on multiple cores, but this has a negative impact both on the WCRT, on the power consumption, and on the temperature; *(iii)* keeping the temperature below a given threshold is even more difficult because the temperature continues to increase even after the activity stops, so each scheduling decision must be assessed not based on the current state of the chip (*i.e.*, the temperature of each core) but on the state of the chip at the end of the candidate task, and cooling slacks must be inserted. We have proposed a multi-criteria scheduling heuristics to address these challenges. It produces a static schedule of the given application graph and the given architecture description, such that the GSFR, power, and temperature thresholds are satisfied, and such that the execution time is minimized. We then combine our heuristic with a variant of the ε-constraint method [62] in order to produce, for a given application graph and a given architecture description, its entire Pareto front in the 4D space (exec. time, GSFR, power, temp.). This is a joint work with Athena Abdi and Hamid Zarandi from Amirkabir U., Iran, who have visited the team in 2016.

6.3.3. Automatic transformations for fault tolerant circuits

In the past years, we have studied the implementation of specific fault tolerance techniques in real-time embedded systems using program transformation [1]. We are now investigating the use of automatic transformations to ensure fault-tolerance properties in digital circuits. To this aim, we consider program transformations for

hardware description languages (HDL). We consider both single-event upsets (SEU) and single-event transients (SET) and fault models of the form *"at most 1 SEU or SET within n clock cycles"*.

We have expressed several variants of triple modular redundancy (TMR) as program transformations. We have proposed a verification-based approach to minimize the number of voters in TMR [25]. Our technique guarantees that the resulting circuit *(i)* is fault tolerant to the soft-errors defined by the fault model and *(ii)* is functionally equivalent to the initial one. Our approach operates at the logic level and takes into account the input and output interface specifications of the circuit. Its implementation makes use of graph traversal algorithms, fixed-point iterations, and BDDs. Experimental results on the ITC'99 benchmark suite indicate that our method significantly decreases the number of inserted voters which entails a hardware reduction of up to 55% and a clock frequency increase of up to 35% compared to full TMR. We address scalability issues arising from formal verification with approximations and assess their efficiency and precision. As our experiments show, if the SEU fault-model is replaced with the stricter fault-model of SET, it has a minor impact on the number of removed voters. On the other hand, BDD-based modeling of SET effects represents a more complex task than the modeling of an SEU as a bit-flip. We propose solutions for this task and explain the nature of encountered problems. We discuss scalability issues arising from formal verification with approximations and assess their efficiency and precision.

6.3.4. Concurrent flexible reversibility

Reversible concurrent models of computation provide natively what appears to be very fine-grained checkpoint and recovery capabilities. We have made this intuition clear by formally comparing a distributed algorithm for checkpointing and recovery based on causal information, and the distributed backtracking algorithm that lies at the heart of our reversible higher-order pi-calculus. We have shown that (a variant of) the reversible higher-order calculus with explicit rollback can faithfully encode a distributed causal checkpoint and recovery algorithm. The reverse is also true but under precise conditions, which restrict the ability to rollback a computation to an identified checkpoint. This work has currently not been published.

7. Bilateral Contracts and Grants with Industry

7.1. Bilateral Contracts with Industry

- INRIA and Orange Labs have established this year a joint virtual research laboratory, called I/O LAB. We have been heavily involved in the creation of the laboratory and are actively involved in its operation (Jean-Bernard Stefani is one of the two co-directors of the lab). I/O LAB focuses on the network virtualization and cloudification. As part of the work of I/O LAB, we have cooperated with Orange Lab, as part of a cooperative research contract funded by Orange, on defining architectural principles and frameworks for network cloud infrastructures encompassing control and management of computing, storage and network resources.
- With Daimler (subcontracting via iUTBS): We have shown how to extend our current method for computing deadline miss models to real-time systems designed according to the Logical Execution Time paradigm.

7.2. Bilateral Grants with Industry

With Thales: Early Performance assessment for evolving and variable Cyber-Physical Systems. This CIFRE grant funds the PhD of Christophe Prévot.

8. Partnerships and Cooperations

8.1. Regional Initiatives

8.1.1. CASERM (PERSYVAL-Lab project)

Participants: Pascal Fradet, Alain Girault, Gregor Goessler, Xiaojie Guo, Xavier Nicollin, Stephan Plassart, Sophie Quinton, Jean-Bernard Stefani.

Despite recent advances, there exist currently no integrated formal methods and tools for the design and analysis of reconfigurable multi-view embedded systems. This is the goal of the CASERM project.

The CASERM project represents a significant effort towards a COQ-based design method for reconfigurable multi-view embedded systems, in order to formalize the structure and behavior of systems and to prove their main properties. The use of a proof assistant to support such a framework is motivated by the fact that the targeted systems are both extremely complex and critical. The challenges addressed are threefold:

1. to model software architectures for embedded systems taking into account their dynamicity and multiple constraints (functional as well as non functional);
2. to propose novel scheduling techniques for dynamically reconfiguring embedded systems; and
3. to advance the state of the art in automated proving for such systems.

The objectives of CASERM that address these challenges are organized in three tasks. They consist respectively in designing an architecture description framework based on a process calculus, in proposing online optimization methods for dynamic reconfiguration systems (this is the topic of Stephan Plassart's PhD), and in developing a formal framework for real-time analysis in the COQ proof assistant (this is the topic of Xiaojie Guo's PhD). A fourth task focuses on common case studies for the evaluation of the obtained results.

The CASERM consortium gathers researchers from the G-SCOP, LIG and VERIMAG laboratories who are reknown specialists in these fields. The project started in November 2016 and will last three years.

8.2. European Initiatives

8.2.1. Collaborations with Major European Organizations

We have a strong collaboration with the Technische Universität Braunschweig in Germany. In particular, Sophie Quinton is involved in the CCC project (http://ccc-project.org/) to provide methods and mechanisms for the verification of software updates after deployment in safety-critical systems and in the TypicalCPA project which aims at computing deadline miss models for distributed systems.

We also a recent collaboration with the MPI-SWS in Kaiserslautern (Germany) on formal proofs for real-time systems.

8.3. International Initiatives

8.3.1. Inria Associate Teams Not Involved in an Inria International Labs

8.3.1.1. Causalysis

Title: Causality Analysis for Safety-Critical Embedded Systems

International Partner (Institution - Laboratory - Researcher):

University of Pennsylvania (United States) - PRECISE center - Oleg Sokolsky

Start year: 2015

See also: https://team.inria.fr/causalysis/

Today's embedded systems become more and more complex, while an increasing number of safety-critical functions rely on them. Determining the cause(s) of a system-level failure and elucidating the exact scenario that led to the failure is today a complex and tedious task that requires significant expertise. The CAUSALYSIS project will develop automated approaches to causality analysis on execution logs.

8.4. International Research Visitors

8.4.1. Visits of International Scientists

8.4.1.1. Internships

- Athena Abdi has been a visitor in the team from October 2015 to June 2016. She is doing her PhD at the Amirkabir University of Technology in Teheran, Iran. In the SPADES team, she is working on multi-criteria scheduling for real-time embedded systems, addressing the complex interplay between reliability, power consumption, temperature, and execution time (see 6.3.2).
- Ismail Assayad has been a visitor in the team in September 2015. He is assistant professor at the University of Casablanca, Morocco. In the SPADES team, he is working on adaptive scheduling methods and admission control for dynamic embedded applications (see 6.3.2).

9. Dissemination

9.1. Promoting Scientific Activities

9.1.1. Scientific events organisation

9.1.1.1. Member of organizing committees

- Sophie Quinton was artifact evaluation chair of the 24th International Conference on Real-Time Networks and Systems (RTNS'16).
- Sophie Quinton was demo chair of the 22nd IEEE Real-Time Embedded Technology & Applications Symposium (RTAS'16)
- Sophie Quinton was co-chair of the 1st Tutorial on Tools for Real-Time Systems (TuToR'16), held as a satellite event of CPSWeek'16. http://tutor2016.inria.fr/
- Sophie Quinton was co-organizer of the 1st Workshop on Collaboration of Academia and Industry for Real World Embedded Systems (CAIRES'16), held as a satellite event of ESWeek'16. http://caires2016.inria.fr/

9.1.2. Scientific events selection

9.1.2.1. Chair of conference program committees

- Gregor Gössler was co-chair of the 1st international Workshop on Causal Reasoning for Embedded and safety-critical Systems Technologies (CREST'16) [22], held as a satellite event of ETAPS'16. http://crest2016.inria.fr
- Sophie Quinton was co-chair of the 7th International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems (WATERS'16), held as a satellite event of ECRTS'16. http://waters2016.inria.fr

9.1.2.2. Member of conference program committees

- Pascal Fradet served in the program committee of the 15th International Conference on Modularity (MODULARITY'16).
- Alain Girault served in the program committees of the International Conference on Design and Test in Europe (DATE'16), the Embedded Software conference (EMSOFT'16), and the International Symposium on Industrial Embedded Systems (SIES'16).
- Sophie Quinton served in the program committees of the 28th Euromicro Conference on Real-Time Systems (ECRTS'16), the 24th International Conference on Real-Time Networks and Systems (RTNS'16), the 4th International Workshop on Mixed Criticality Systems (WMC'16), the 10th Junior Researcher Workshop on Real-Time Computing (JRWRTC'16), and in the artifact evaluation committees of ECRTS'16 and the IEEE Real-Time Systems Symposium (RTSS'16).
- Jean-Bernard Stefani served on the program committees of the 36th IFIP International Conference on Formal Techniques for Distributed Objects, Components and Systems (FORTE) and the 8th Conference on Reversible Computation.

9.1.2.3. Reviewer

- Alain Girault reviewed an article for ECRTS'16.
- Gregor Gössler reviewed articles for EMSOFT'16, FACS'16, and RTNS'16.
- Xavier Nicollin reviewed an article for SIES'16.
- Sophie Quinton reviewed articles for EMSOFT'16 and DATE'17.

9.1.3. Journal

9.1.3.1. Member of the editorial boards

- Alain Girault is a member of the editorial board of the EURASIP Journal on Embedded Systems.
- Jean-Bernard Stefani is a member of the editorial board of Annals of Telecommunications.

9.1.3.2. Reviewer - Reviewing activities

- Alain Girault reviewed articles for ACM TECS, Parallel Computing, Embedded Systems Letters, and Microprocessors and Microsystems.
- Gregor Gössler reviewed articles for Formal Methods in System Design (FMSD) and IEEE Transactions on Automatic Control (TAC).
- Jean-Bernard Stefani reviewed articles for Theoretical Computer Science (TCS) and Science of Computer Programming (SCP).

9.1.4. Research administration

- Pascal Fradet is head of the committee for doctoral studies ("Responsable du comité des études doctorales") of the INRIA Grenoble – Rhône-Alpes research center and local correspondent for the young researchers INRIA mission (mission jeunes chercheurs).
- Alain Girault is Vice Chair of the INRIA Evaluation Committee. As such, he co-organizes in particular the evaluation seminars of the INRIA teams (twice a year) and all the juries for the hiring and promotion of INRIA's researchers (CR2, CR1, DR2, DR1, and DR0).
- Jean-Bernard Stefani is Head of science of the INRIA Grenoble – Rhône-Alpes research center. As such, he manages with the research center director all aspects of the scientific life of the research center (creation of the research teams and their evaluation by international panels, scientific

Master : Sophie Quinton, Performance and Quantitative Properties, 6h, MOSIG, Univ. Grenoble Alpes, France

9.2.2. Supervision

- PhD: Yoann Geoffroy, “A general trace-based causality framework for component-based systems”, Univ. Grenoble Alpes, defended on December 7th 2016, advised by Gregor Gössler.
- PhD in progress: Sihem Cherrared, “Fault Management in Multi-Tenant Programmable Networks”, Univ. Rennes 1, since October 2016, co-advised by Eric Fabre and Gregor Gössler.
- PhD in progress: Christophe Prévot, “Early Performance assessment for evolving and variable Cyber-Physical Systems”, Univ. Grenoble Alpes, since November 2015, co-advised by Alain Girault and Sophie Quinton.
- PhD in progress: Xiaojie Guo, “Formal Proofs for the Analysis of Real-Time Systems in Coq”, Univ. Grenoble Alpes, since December 2016, co-advised by Pascal Fradet, Jean-François Monin, and Sophie Quinton.
- PhD in progress: Stephan Plassart, “On-line optimization in dynamic real-time systems”, Univ. Grenoble Alpes, since September 2016, co-advised by Alain Girault and Bruno Gaujal.

9.2.3. Juries

- Alain Girault was president of the HDR jury of Goran Frehse (Univ. Grenoble Alpes).
- Sophie Quinton was member of the PhD jury of Houssam Zahaf (U. Lille).
- Jean-Bernard Stefani was president of the HDR jury of Tom Hirschowitz (U. Savoie).

9.3. Popularization

Alain Girault gave a lecture to high school math professors, titled “Multi-core architectures, reliability, and optimization” (ISN conference cycle, Grenoble, February 2016). http://www.canal-u.tv/video/inria/architectures_multi_coeurs_ fiabilite_et_optimisation.20829

10. Bibliography

Major publications by the team in recent years

[1] T. Ayav, P. Fradet, A. Girault. *Implementing Fault-Tolerance in Real-Time Programs by Automatic Program Transformations*, in "ACM Trans. Embedd. Comput. Syst.", July 2008, vol. 7, n^0 4, pp. 1–43

[2] E. Bruneton, T. Coupaye, M. Leclercq, V. Quema, J.-B. Stefani. *The Fractal Component Model and its Support in Java*, in "Software - Practice and Experience", 2006, vol. 36, n^0 11-12

[3] S. Djoko Djoko, R. Douence, P. Fradet. *Aspects preserving properties*, in "Science of Computer Programming", 2012, vol. 77, n^0 3, pp. 393-422

[4] G. Frehse, A. Hamann, S. Quinton, M. Wöhrle. *Formal Analysis of Timing Effects on Closed-loop Properties of Control Software*, in "35th IEEE Real-Time Systems Symposium 2014 (RTSS)", Rome, Italy, December 2014, https://hal.inria.fr/hal-01097622

[5] A. Girard, G. Gössler, S. Mouelhi. *Safety Controller Synthesis for Incrementally Stable Switched Systems Using Multiscale Symbolic Models*, in "IEEE Transactions on Automatic Control", 2016, vol. 61, n^0 6, pp. 1537-1549 [*DOI :* 10.1109/TAC.2015.2478131], https://hal.archives-ouvertes.fr/hal-01197426

[6] A. GIRAULT, H. KALLA. *A Novel Bicriteria Scheduling Heuristics Providing a Guaranteed Global System Failure Rate*, in "IEEE Trans. Dependable Secure Comput.", December 2009, vol. 6, n^{o} 4, pp. 241–254, Research report Inria 6319, http://hal.inria.fr/inria-00177117

[7] G. GÖSSLER, D. LE MÉTAYER. *A general framework for blaming in component-based systems*, in "Science of Computer Programming", 2015, vol. 113, Part 3 [*DOI :* 10.1016/J.SCICO.2015.06.010], https://hal.inria.fr/hal-01211484

[8] S. LENGLET, A. SCHMIT(a)-652396.679 -11.955 Td [(hal-01211484)]TJ 0 0 0 rg 0 0 0 RG (3) 0.77 TJ 0 0 Lp 0.718 TR inriPublicationsRC

[17] A. BOUAKAZ, P. FRADET, A. GIRAULT. *Symbolic computation of the latency for dataflow graphs*, in "Integrating Dataflow, Embedded computing and Architecture (IDEA'2016)", Vienne, Austria, April 2016, https://hal.inria.fr/hal-01417111

[18] Z. A. H. HAMMADEH, E. ROLF, S. QUINTON, R. HENIA, L. RIOUX. *Bounding Deadline Misses in Weakly-Hard Real-Time Systems with Task Dependencies*, in "Design, Automation and Test in Europe", Lausanne, Switzerland, March 2017, https://hal.inria.fr/hal-01426632

[19] S. HOLTHUSEN, S. QUINTON, I. SCHAEFER, J. SCHLATOW, M. WEGNER. *Using Multi-Viewpoint Contracts for Negotiation of Embedded Software Updates*, in "Workshop on Pre- and Post-Deployment Verification Techniques", Reykjavik, Iceland, June 2016 [*DOI :* 10.4204/EPTCS.208.3], https://hal.inria.fr/hal-01426654

[20] J. WANG, P. S. ROOP, A. GIRAULT. *Energy and timing aware synchronous programming*, in "International Conference on Embedded Software, EMSOFT'16", Pittsburgh, United States, ACM, October 2016, 10 p. [*DOI :* 10.1145/2968478.2968500], https://hal.inria.fr/hal-01412100

Conferences without Proceedings

[21] R. HENIA, A. GIRAULT, C. PRÉVOT, S. QUINTON, L. RIOUX. *Quantifying the Flexibility of Real-Time Systems*, in "10th Junior Researcher Workshop on Real-Time Computing", Brest, France, October 2016, https://hal.inria.fr/hal-01426658

Scientific Books (or Scientific Book chapters)

[22] G. GÖSSLER, O. SOKOLSKY (editors). *Proceedings First Workshop on Causal Reasoning for Embedded and safety-critical Systems Technologies*, EPTCS, August 2016, vol. 224 [*DOI :* 10.4204/EPTCS.224], https://hal.inria.fr/hal-01378792

Books or Proceedings Editing

[23] V. NÉLIS, S. QUINTON (editors). *Work-in-Progress and Demo Proceedings - 2016 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, April 2016, https://hal.inria.fr/hal-01305183

Research Reports

[24] A. BOUAKAZ, P. FRADET, A. GIRAULT. *Symbolic Analysis of Dataflow Graphs (Extended Version)*, Inria - Research Centre Grenoble – Rhône-Alpes, January 2016, n^0 8742, https://hal.inria.fr/hal-01166360

[25] D. BURLYAEV, P. FRADET, A. GIRAULT. *A static analysis for the minimization of voters in fault-tolerant circuits*, Inria - Research Centre Grenoble – Rhône-Alpes, December 2016, n^0 RR-9004, pp. 1-27, https://hal.inria.fr/hal-01417164

[26] L. SHAN, S. GRAF, S. QUINTON. *RTLib: A Library of Timed Automata for Modeling Real-Time Systems*, Grenoble 1 UGA - Université Grenoble Alpe ; Inria Grenoble - Rhone-Alpes, November 2016, https://hal.archives-ouvertes.fr/hal-01393888

[27] E. YIP, P. S. ROOP, A. GIRAULT, M. BIGLARI-ABHARI. *Synchronous Deterministic Parallel Programming for Multicores with ForeC: Programming Language, Semantics, and Code Generation*, Inria - Research Centre Grenoble – Rhône-Alpes, August 2016, n^0 RR-8943, https://hal.inria.fr/hal-01351552

References in notes

[28] *Automotive Open System Architecture*, 2003, http://www.autosar.org

[29] G. LEAVENS, M. SITARAMAN (editors). *Foundations of Component-Based Systems*, Cambridge University Press, 2000

[30] Z. LIU, H. JIFENG (editors). *Mathematical Frameworks for Component Software - Models for Analysis and Synthesis*, World Scientific, 2006

[31] ARTEMIS JOINT UNDERTAKING. *ARTEMIS Strategic Research Agenda*, 2011

[32] S. ANDALAM, P. ROOP, A. GIRAULT. *Predictable Multithreading of Embedded Applications Using PRET-C*, in "International Conference on Formal Methods and Models for Codesign, MEMOCODE'10", Grenoble, France, IEEE, July 2010, pp. 159–168

[33] I. ASSAYAD, A. GIRAULT, H. KALLA. *Tradeoff Exploration between Reliability, Power Consumption, and Execution Time for Embedded Systems*, in "Int. J. Software Tools for Technology Transfer", June 2013, vol. 15, n^0 3, pp. 229–245

[34] P. AXER, R. ERNST, H. FALK, A. GIRAULT, D. GRUND, N. GUAN, B. JONSSON, P. MARWEDEL, J. REINEKE, C. ROCHANGE, M. SEBATIAN, R. VON HANXLEDEN, R. WILHELM, W. YI. *Building Timing Predictable Embedded Systems*, in "ACM Trans. Embedd. Comput. Syst.", 2014, To appear

[35] E. BAINOMUGISHA, A. CARRETON, T. VAN CUTSEM, S. MOSTINCKX, W. DE MEUTER. *A Survey on Reactive Programming*, in "ACM Computing Surveys", 2013, vol. 45, n^0 4

[36] A. BASU, S. BENSALEM, M. BOZGA, J. COMBAZ, M. JABER, T.-H. NGUYEN, J. SIFAKIS. *Rigorous Component-Based System Design Using the BIP Framework*, in "IEEE Software", 2011, vol. 28, n^0 3

[37] V. BEBELIS, P. FRADET, A. GIRAULT. *A Framework to Schedule Parametric Dataflow Applications on Many-Core Platforms*, in "International Conference on Languages, Compilers and Tools for Embedded Systems, LCTES'14", Edinburgh, UK, ACM, June 2014

[38] V. BEBELIS, P. FRADET, A. GIRAULT, B. LAVIGUEUR. *BPDF: A Statically Analyzable Dataflow Model with Integer and Boolean Parameters*, in "International Conference on Embedded Software, EMSOFT'13", Montreal, Canada, ACM, September 2013

[39] A. BENVENISTE, P. CASPI, S. A. EDWARDS, N. HALBWACHS, P. LE GUERNIC, R. DE SIMONE. *The synchronous languages 12 years later*, in "Proceedings of the IEEE", 2003, vol. 91, n^0 1

[40] A. BENVENISTE, J. RACLET, B. CAILLAUD, D. NICKOVIC, R. PASSERONE, A. SANGIOVANNI-VICENTELLI, T. HENZINGER, K. LARSEN. *Contracts for the Design of Embedded Systems Part I: Methodology and Use Cases*, in "Proceedings of the IEEE", 2012

[41] A. BENVENISTE, J. RACLET, B. CAILLAUD, D. NICKOVIC, R. PASSERONE, A. SANGIOVANNI-VICENTELLI, T. HENZINGER, K. LARSEN. *Contracts for the Design of Embedded Systems Part II: Theory*, in "Proceedings of the IEEE", 2012

[42] B. BONAKDARPOUR, S. S. KULKARNI, F. ABUJARAD. *Symbolic synthesis of masking fault-tolerant distributed programs*, in "Distributed Computing", 2012, vol. 25, n^0 1

[43] S. BORKAR. *Designing Reliable Systems from Unreliable Components: The Challenges of Transistor Variability and Degradation*, in "IEEE Micro", 2005, vol. 25, n^0 6

[44] R. BRUNI, H. C. MELGRATTI, U. MONTANARI. *Theoretical foundations for compensations in flow composition languages*, in "32nd ACM Symposium on Principles of Programming Languages (POPL)", ACM, 2005

[45] S. BURCKHARDT, D. LEIJEN. *Semantics of Concurrent Revisions*, in "European Symposium on Programming, ESOP'11", Saarbrucken, Germany, LNCS, Springer, March 2011, n^0 6602, pp. 116–135

[46] P. CASPI, M. POUZET. *Synchronous Kahn Networks*, in "ACM SIGPLAN International Conference on Functional Programming, ICFP'96", Philadelphia (PA), USA, ACM, May 1996

[47] T. CHOTHIA, D. DUGGAN. *Abstractions for fault-tolerant global computing*, in "Theor. Comput. Sci.", 2004, vol. 322, n^0 3

[48] R. DAVIS, A. BURNS. *A Survey of Hard Real-Time Scheduling for Multiprocessor Systems*, in "ACM Computing Surveys", 2011, vol. 43, n^0 4

[49] V. DE FLORIO, C. BLONDIA. *A Survey of Linguistic Structures for Application-Level Fault-Tolerance*, in "ACM Computing Surveys", 2008, vol. 40, n^0 2

[50] G. DELAVAL. *Répartition modulaire de programmes synchrones*, INPG, Inria Grenoble Rhône-Alpes, July 2008, PhD thesis

[51] G. DELAVAL, A. GIRAULT, M. POUZET. *A Type System for the Automatic Distribution of Higher-order Synchronous Dataflow Programs*, in "International Conference on Languages, Compilers, and Tools for Embedded Systems, LCTES'08", Tucson (AZ), USA, ACM, June 2008, pp. 101–110, ftp://ftp.inrialpes.fr/pub/bip/pub/girault/Publications/Lctes08/main.pdf

[52] S. A. EDWARDS, E. A. LEE. *The Case for the Precision Timed (PRET) Machine*, in "44th Design Automation Conference (DAC)", IEEE, 2007

[53] J. EKER, J. W. JANNECK, E. A. LEE, J. LIU, X. LIU, J. LUDVIG, S. NEUENDORFFER, S. SACHS, Y. XIONG. *Taming heterogeneity - the Ptolemy approach*, in "Proceedings of the IEEE", 2003, vol. 91, n^0 1

[54] J. FIELD, C. A. VARELA. *Transactors: a programming model for maintaining globally consistent distributed state in unreliable environments*, in "32nd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL)", ACM, 2005

[55] P. FRADET, A. GIRAULT, P. POPLAVKO. *SPDF: A Schedulable Parametric Data-Flow MoC*, in "Design Automation and Test in Europe, DATE'12", Dresden, Germany, 2012, http://hal.inria.fr/hal-00744376

[56] A. GIRARD, G. PAPPAS. *Approximation metrics for discrete and continuous systems*, in "IEEE Trans. on Automatic Control", 2007, vol. 52, n^0 5, pp. 782–798

[57] A. GIRAULT, H. KALLA. *A Novel Bicriteria Scheduling Heuristics Providing a Guaranteed Global System Failure Rate*, in "IEEE Trans. Dependable Secure Comput.", December 2009, vol. 6, n° 4, pp. 241–254, Research report Inria 6319, http://www.computer.org/portal/web/csdl/doi/10.1109/TDSC.2008.50

[58] D. GIZOPOULOS, M. PSARAKIS, S. V. ADVE, P. RAMACHANDRAN, S. K. S. HARI, D. SORIN, A. MEIXNER, A. BISWAS, X. VERA. *Architectures for Online Error Detection and Recovery in Multicore Processors*, in "Design Automation and Test in Europe (DATE)", 2011

[59] F. C. GÄRTNER. *Fundamentals of Fault-Tolerant Distributed Computing in Asynchronous Environments*, in "ACM Computing Surveys", 1999, vol. 31, n° 1

[60] G. GÖSSLER, J.-B. STEFANI. *Fault Ascription in Concurrent Systems*, in "Proc. Trustworthy Global Computing - 10th International Symposium, TGC 2015", P. GANTY, M. LORETI (editors), LNCS, Springer, 2016, vol. 9533

[61] S. HAAR, E. FABRE. *Diagnosis with Petri Net Unfoldings*, in "Control of Discrete-Event Systems", Lecture Notes in Control and Information Sciences, Springer, 2013, vol. 433, chap. 15

[62] Y. HAIMES, L. LASDON, D. WISMER. *On a Bicriterion Formulation of the Problems of Integrated System Identification and System Optimization*, in "IEEE Trans. Systems, Man, and Cybernetics", 1971, vol. 1, pp. 296–297

[63] J. HALPERN, J. PEARL. *Causes and Explanations: A Structural-Model Approach. Part I: Causes*, in "British Journal for the Philosophy of Science", 2005, vol. 56, n° 4, pp. 843-887

[64] D. HARMANCI, V. GRAMOLI, P. FELBER. *Atomic Boxes: Coordinated Exception Handling with Transactional Memory*, in "25th European Conference on Object-Oriented Programming (ECOOP)", Lecture Notes in Computer Science, 2011, vol. 6813

[65] T. HENZINGER, J. SIFAKIS. *The Embedded Systems Design Challenge*, in "Formal Methods 2006", Lecture Notes in Computer Science, Springer, 2006, vol. 4085

[66] I. HWANG, S. KIM, Y. KIM, C. E. SEAH. *A Survey of Fault Detection, Isolation and Reconfiguration Methods*, in "IEEE Trans. on Control Systems Technology", 2010, vol. 18, n° 3

[67] V. IZOSIMOV, P. POP, P. ELES, Z. PENG. *Scheduling and Optimization of Fault-Tolerant Embedded Systems with Transparency/Performance Trade-Offs*, in "ACM Trans. Embedded Comput. Syst.", 2012, vol. 11, n° 3, 61 p.

[68] C. M. KIRSCH, A. SOKOLOVA. *The Logical Execution Time Paradigm*, in "Advances in Real-Time Systems (to Georg Färber on the occasion of his appointment as Professor Emeritus at TU München after leading the Lehrstuhl für Realzeit-Computersysteme for 34 illustrious years)", 2012, pp. 103–120

[69] R. KÜSTERS, T. TRUDERUNG, A. VOGT. *Accountability: definition and relationship to verifiability*, in "ACM Conference on Computer and Communications Security", 2010, pp. 526-535

[70] I. LANESE, C. A. MEZZINA, J.-B. STEFANI. *Reversing Higher-Order Pi*, in "21th International Conference on Concurrency Theory (CONCUR)", Lecture Notes in Computer Science, Springer, 2010, vol. 6269

[71] E. A. LEE, A. L. SANGIOVANNI-VINCENTELLI. *Component-based design for the future*, in "Design, Automation and Test in Europe, DATE 2011", IEEE, 2011

[72] P. MENZIES. *Counterfactual Theories of Causation*, in "Stanford Encyclopedia of Philosophy", E. ZALTA (editor), Stanford University, 2009, http://plato.stanford.edu/entries/causation-counterfactual

[73] M. MOORE. *Causation and Responsibility*, Oxford, 1999

[74] J. PEARL. *Causal inference in statistics: An overview*, in "Statistics Surveys", 2009, vol. 3, pp. 96-146

[75] P. RAMADGE, W. WONHAM. *Supervisory Control of a Class of Discrete Event Processes*, in "SIAM Journal on control and optimization", January 1987, vol. 25, n^0 1, pp. 206–230

[76] G. RAMALINGAM, K. VASWANI. *Fault tolerance via idempotence*, in "40th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL)", ACM, 2013

[77] B. RANDELL. *System Structure for Software Fault Tolerance*, in "IEEE Trans. on Software Engineering", 1975, vol. 1, n^0 2

[78] J. RUSHBY. *Partitioning for Safety and Security: Requirements, Mechanisms, and Assurance*, NASA Langley Research Center, 1999, n^0 CR-1999-209347

[79] J.-B. STEFANI. *Components as Location Graphs*, in "11th International Symposium on Formal Aspects of Component Software", Bertinoro, Italy, Lecture Notes in Computer Science, September 2014, vol. 8997, https://hal.inria.fr/hal-01094208

[80] P. TABUADA. *Verification and Control of Hybrid Systems - A Symbolic Approach*, Springer, 2009

[81] D. WALKER, L. W. MACKEY, J. LIGATTI, G. A. REIS, D. I. AUGUST. *Static typing for a faulty lambda calculus*, in "11th ACM SIGPLAN International Conference on Functional Programming (ICFP)", ACM, 2006

[82] R. WILHELM, J. ENGBLOM, A. ERMEDAHL, N. HOLSTI, S. THESING, D. B. WHALLEY, G. BERNAT, C. FERDINAND, R. HECKMANN, T. MITRA, F. MUELLER, I. PUAUT, P. P. PUSCHNER, J. STASCHULAT, P. STENSTRÖM. *The Determination of Worst-Case Execution Times — Overview of the Methods and Survey of Tools*, in "ACM Trans. Embedd. Comput. Syst.", April 2008, vol. 7, n^0 3

[83] E. YIP, P. ROOP, M. BIGLARI-ABHARI, A. GIRAULT. *Programming and Timing Analysis of Parallel Programs on Multicores*, in "International Conference on Application of Concurrency to System Design, ACSD'13", Barcelona, Spain, IEEE, July 2013, pp. 167–176, https://hal.inria.fr/hal-00842402