Activity Report 2015

Project-Team SPADES

Sound Programming of Adaptive Dependable Embedded Systems

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:

1.1.9. - Fault tolerant systems
1.3. - Distributed Systems
2.1.1. - Semantics of programming languages
2.1.3. - Functional programming
2.1.8. - Synchronous languages
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

Other Research Topics and Application Domains:

6.6. - Embedded systems

1. Members

Research Scientists

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

Faculty Member

Xavier Nicollin [Univ. Grenoble Alpes - Grenoble INP, Associate Professor]

PhD Students

Vagelis Bebelis [Inria, until Feb 2015]
Dmitry Burlyaev [Univ. Grenoble Alpes, until Nov 2015]
Yoann Geoffroy [Univ. Grenoble Alpes]
Christophe Prévot [Thales, from Oct 2015, CIFRE grant]

Post-Doctoral Fellows

Clement Aubert [Inria, until Sep 2015]
Adnan Bouakaz [Inria]

Visiting Scientists

Atena Abdi [Amirkabir University of Technology, from Oct 2015]
Ismail Assayad [Univ. Casablanca, Sep 2015]
Soenke Holthusen [Technische Universität Braunschweig, until Apr 2015]

Administrative Assistant

Helen Pouchot-Rouge-Blanc [Inria]

Other

Martin Vassor [Inria, from Jul 2015 until Aug 2015]

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 [35], 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 [73]. Witness component-based toolsets such as UC Berkeley's PTOLEMY [60], Verimag's BIP [40], or the modular architecture frameworks used, for instance, in the automotive industry (AUTOSAR) [32]. 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.*, [33], [34]). 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.*, [44], [45] and the references therein), but having a comprehensive theory of contracts that deals with all the above aspects is still an open question [78]. 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 [43]), 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 [39], [54], 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. [39]. 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 [47], [66]. 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 [85], 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 [55]. 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.*, [72] 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 [74].

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.*, [67] 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 [46], [48], [52], [61], [75], [84], [91] 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 [80], natural sciences, law [81], and statistics [82], 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 [77], to allow the diagnosis of faults in a complex concurrent system [68], or to enforce accountability [76], 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 [67]), or is broken (*e.g.*, by limiting fault propagation [86]).

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. COSYMA: Controller synthesis using multi-scale abstractions

FUNCTIONAL DESCRIPTION

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

CoSyMA is a tool for automatic controller synthesis for incrementally stable switched systems based on multi-scale discrete abstractions. The tool accepts as input a switched system defined by differential equations indexed by a set of modes, time and space sampling parameters used to define an approximation of the continuous state-space, and a safety or a time-bounded reachability specification. CoSyMA computes and refines discrete abstractions of the state space so as to generate a controller, if one exists, for the system that enforces the specification.

- Authors: Antoine Girard, Gregor Gössler, and Sebti Mouelhi.
- Partner: LJK.
- Contact: Gregor Gössler.

5.2. LoCa: Logical Causality Analyzer

FUNCTIONAL DESCRIPTION

Based on an execution trace, the component specifications, and a required property P, LoCA analyzes the causes of a violation of P in a component-based system. LoCA currently supports causality analysis in BIP and networks of timed automata. The core analysis engine is implemented as an abstract class, such that support for other models of computation (MoC) can be added by instantiating the class with the basic operations of the MoC.

- Authors: Lacramioara Astefanoaei, Yoann Geoffroy, and Gregor Gössler.
- Contact: Gregor Gössler.

5.3. LDDL: Coq proofs of circuit transformations for fault-tolerance

FUNCTIONAL DESCRIPTION

We have been developing a COQ-based framework to formally verify the functional and fault-tolerance properties of circuit transformations. Circuits are described at the gate level using LDDL, a Low-level Dependent Description Language inspired from μFP [87]. Our combinator language, equipped with dependent types, ensures that circuits are well-formed by construction (gates correctly plugged, no dangling wires, no combinational loops, ...). Faults like Single-Event Upsets (SEUs) (*i.e.*, bit-flips in flipflops) and SETs (*i.e.*, glitches propagating in the combinational circuit) and fault-models like *"at most 1 SEU or SET within n clock cycles"* are described in the operational semantics of LDDL. Fault-tolerance techniques are described as transformations of LDDL circuits.

The framework has been used to prove the correctness of three fault-tolerance techniques: TMR, TTR and DTR (see Section 6.3.3). The size of specifications and proofs for the common part (LDDL syntax and semantics, libraries) is 5000 lines of COQ (excluding comments and blank lines), 700 for TMR, 3500 for TTR and 7000 for DTR.

- Authors: Dmitry Burlyaev and Pascal Fradet.
- Contact: Pascal Fradet.
- URL: https://team.inria.fr/spades/fthwproofs

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

FUNCTIONAL DESCRIPTION

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 and Zain Hammadeh (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. So far, pyCPA_TWCA is restricted to uniprocessor systems of independent tasks, scheduled according to static priority scheduling. A public release is planned for 2016.

- Contact: Sophie Quinton.

6. New Results

6.1. Components and contracts

Participants: Sophie Quinton, Jean-Bernard Stefani.

6.1.1. Multi-viewpoint contracts for the negotiation of embedded software updates

In the context of the CCC project (http://ccc-project.org/) we address the issue of change after deployment in safety-critical embedded system applications. Our goal 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 started validating our approach on a concrete example inspired by the automotive domain in collaboration with our German partners from TU Braunschweig.

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 [88], 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 the location graph model and extended it in several directions. First we have introduced basic capabilities and predicate parameters in the model to allow for different forms of architectural invariants, such as different forms of encapsulation, to be maintained even in presence of dynamic graph modifications. Second, we have started developing the premises of a refinement theory for location graphs, showing in particular how one could refine a location process into a whole graph. Finally, we have shown how to handle heterogeneous forms of composition in the same location graph, turning each location into a composition operator. This work has not yet been published.

6.2. Real-Time multicore programming

Participants: Vagelis Bebelis, Adnan Bouakaz, Pascal Fradet, Alain Girault, Gregor Goessler, Xavier Nicollin, Jean-Bernard Stefani.

6.2.1. A time predictable programming language for multicores

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 [59]. 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 [92]. 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 [38].

Following our past results on the PRET-C programming language [36], 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 [93] (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. 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.

Recent results have addressed the semantics, the compiler, and the experiments. In particular, we have seeked to provide several combine policies for shared variables, in a way similar as concurrent revisions [49].

This work has been conducted within the RIPPES associated team.

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 [51]). 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 [56], [57], 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. The overall structure of the new compiler is defined, including new algorithms for deployment (and code optimization), achieving the three aims detailed above.

6.2.3. Analysis and scheduling of 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 [62], which extends the standard dataflow model by allowing rates to be parametric, and the *Boolean Parametric Data Flow* (BPDF) MoC [42], [41] which combines integer parameters (to express dynamic rates) and boolean parameters (to express the activation and deactivation of communication channels). High dynamicity is provided by integer parameters which can change at each basic iteration and boolean parameters which can change even within the iteration. We have presented static analyses that ensure the liveness and the boundedness of BDPF graphs.

This year, we have focused on the *symbolic* analysis of parametric data-flow graphs. This work has been conducted within the RIPPES associated team.

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.*, [90] 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 [83]. 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 [63].

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.

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. Typical Worst-Case Analysis of 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. Our major contribution this year is a general formulation of that problem in the context of systems where some tasks occasionally experience sporadic overload [26]. Based on this new formulation, we present an algorithm that can take into account fine-grained effects of overload at the input of different tasks when computing deadline miss bounds. We show in experiments with synthetic as well as industrial data that our algorithm produces bounds that are much tighter than in previous work, in sufficiently short time. In addition, we improve, in the preemptive case, the criterion proposed in [71] for establishing how much overload can be tolerated in a time window while still guaranteeing absence of deadline misses: our new criterion is a necessary and sufficient condition (as opposed to the sufficient condition of [71]) and therefore yields better results.

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. It is, however, sufficient in many application contexts to budget these tasks in order to preserve weakly-hard rather than hard guarantees. We have thus developed an extension of sensitivity analysis for deriving task budgets for systems with hard and weakly-hard requirements. We currently validate our approach on synthetic test cases and a realistic case study given by our partner Thales.

6.3. Language Based Fault-Tolerance

Participants: Dmitry Burlyaev, Pascal Fradet, Alain Girault, Yoann Geoffroy, Gregor Goessler, Jean-Bernard Stefani, Atena Abdi, Ismail Assayad.

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 [70] 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].

We have instantiated our approach to a synchronous data flow framework defined by a subset of the LUSTRE [69] language, and implemented the analysis in LOCA (see Section 5.2).

In [25] we have shown that we can improve precision of the analysis if (1) we can emulate execution of components instead of relying on their specifications, and (2) take into consideration input/output dependencies between components to avoid blaming components for faults induced by other components. We have demonstrated the utility of the extended analysis with a case study for a closed-loop patient-controlled analgesia system.

We have further proposed in [23] a general semantic framework for fault ascription. Our framework relies on configuration structures to handle concurrent systems, partial and distributed observations in a uniform way. It defines basic conditions for a counterfactual analysis of necessary and sufficient causes, and it presents a refined analysis that conforms to our basic conditions while avoiding various infelicities.

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 (1) allocate a sufficient number of cores to each active application, (2) allocate the unassigned cores to the applications yielding the largest gain in energy, and (3) 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 threshold. There are multiple difficulties: (1) 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 [37]; to overcome this, we adopt instead the Global System Failure Rate (GSFR) as a measure of the system's reliability that is invariant with time [64]; (2) 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; (3) 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. This is a joint work with Atena Abdi (Amirkabir U., Iran) who is a PhD visitor in the team.

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 proposed novel fault-tolerance transformations based on time-redundancy. In particular, we have presented a transformation using double-time redundancy (DTR) coupled with micro-checkpointing, rollback and a speedup mode [19]. The approach is capable to mask any SET every 10 cycles and keeps the same input/output behavior regardless error occurrences. Usually transparent masking requires triple redundancy and voting. Experimental results on the ITC'99 benchmark suite indicate that the hardware overhead of DTR is 2.7 to 6.1 times smaller than full TMR with a double loss in throughput. The method does not require any specific hardware support and is an interesting alternative to Triple Modular Redundancy (TMR) for logic intensive designs.

We have also designed a transformation that allows the circuit to change its level of time-redundancy. This feature allows the circuit to dynamically and temporarily lower (resp. increase) fault-tolerance and speed up (resp. slow down) its computation without interruption [20]. The motivations for such changes can be based on the current radiation environment or the processing of critical data. When hardware size is limited and fault-tolerance is only occasionally needed, that scheme is a better choice than static TMR, which involves a constant high area overhead

These time redundancy transformations (DTR and adaptive fault-tolerance) have been patented [50]

We have described how to formally certify fault-tolerant transformations using the Coq proof assistant [53] (see Section 5.3). The transformations are described on a simple gate-level hardware description language LDDL (Low-level Dependent Description Language). This combinator language is equiped with dependent types and ensures that circuits are well-formed by construction (gates correctly plugged, no dangling wires, no combinational loops, ...). Fault-models are specified in the operational semantics of the language. The main theorem states that, for any circuit, for any input stream and for any SET allowed by the fault-model, its transformed version produces a correct output [18]. The primary motivation of this work was to certify DTR whose intricacy requested a formal proof to make sure that no single-point of failure existed. We have first applied this approach to the correctness proofs of TMR, TTR (Triple Time Redundancy) and finally DTR.

This research is part of Dmitry Burlyaev's pHD thesis [11] defended in November 2015.

6.3.4. A formal approach for the synthesis and implementation of fault-tolerant embedded systems

We have been working for several years on the usage of discrete controller synthesis (DCS) [83] to provide the automated addition of fault-tolerance in embedded systems with formal guarantees [65]. The first key idea is that the initial system model (usually an LTS) includes both the expected behaviors, the unexpected ones (that is, the failures), and the reconfigurations (typically repair actions). The second key idea is that the failures are modeled as *uncontrollable* events. Then, thanks to an exhaustive state space traversal, DCS is able to generate a *controller* that will prevent the system from entering a "bad" state (*e.g.*, a configuration of the system where a task is active on a faulty processor). From the point of view of fault-tolerance, this approach combines the advantages of static guarantees with that of dynamic reconfiguration (hence without the penalty of static redundancy).

Through this new work, we have demonstrated the feasibility of a complete workflow to synthesize and implement correct-by-construction fault tolerant distributed embedded systems consisting of real-time periodic tasks [24]. Correct-by-construction is provided by the use of DCS, which allows us to guarantee that the synthesized controlled system guarantees the functionality of its tasks even in the presence of processor failures. For this step, our workflow uses the HEPTAGON domain specific language [58] and the SIGALI DCS tool [79]. The correct implementation of the resulting distributed system is a challenge, all the more since the controller itself must be tolerant to the processor failures. We achieve this step thanks to the libDGALS real-time library [89] (1) to generate the glue code that will migrate the tasks upon processor failures, maintaining their internal state through migration, and (2) to make the synthesized controller itself fault-tolerant.

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 applied our recent improvements regarding the analysis of deadline miss models for real-time systems to the specific needs of Daimler in the context of CAN buses.

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. National Initiatives

8.1.1. ANR Projects

8.1.1.1. REVER (ANR project)

Participant: Jean-Bernard Stefani.

The REVER project aims to develop semantically well-founded and composable abstractions for dependable distributed computing on the basis of a reversible programming model, where reversibility means the ability to undo any program execution and to revert it to a state consistent with the past execution. The critical assumption behind REVER is that by combining reversibility with notions of compensation and modularity, one can develop systematic and composable abstractions for dependable programming.

The REVER work program is articulated around three major objectives:

- To investigate the semantics of reversible concurrent processes.
- To study the combination of reversibility with notions of compensation, isolation and modularity in a concurrent and distributed setting.
- To investigate how to support these features in a practical (typically, object-oriented and functional) programming language design.

The project partners are Inria (FOCUS and SPADES teams), Université de Paris VII (PPS laboratory), and CEA (List laboratory). The project ended in November 2015.

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 actively participates 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.

8.3. International Initiatives

8.3.1. Inria International Labs

Inria@SiliconValley
Associate Team involved in the International Lab:

8.3.1.1. RIPPES

Title: RIgorous Programming of Predictable Embedded Systems

International Partner (Institution – Laboratory – Researcher):

University of California Berkeley (United States) – Electrical Engineering and Computer Science Department (EECS) – Edward Lee

University of Auckland (New Zealand) – Electrical Computer Engineering Department (ECE) – Partha Roop

Start year: 2013

See also: https://wiki.inria.fr/rippes

The RIPPES associated teams gathers the SPADES team from Inria Grenoble Rhône-Alpes, the PTOLEMY group from UC Berkeley (EECS Department), and the Embedded Systems Research group from U. of Auckland (ECE Department). The planned research seeks to reconcile two contradictory objectives of embedded systems, more predictability and more adaptivity. We have addressed these issues by exploring two complementary research directions: (1) by starting from a classical concurrent C or Java programming language and enhancing it to provide more predictability (see Section 6.2.1), and (2) by starting from a very predictable model of computation (SDF) and enhancing it to provide more adaptivity (see Section 6.2.3).

8.3.2. Inria Associate Teams not involved in an Inria International Labs

8.3.2.1. 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

- Atena 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. General chair, scientific chair

- Alain Girault was general chair of the 10th International Federated Conference on Distributed Computing Techniques (DisCoTec'15), held in Grenoble in June 2015. http://discotec2015.inria.fr
- Jean-Bernard Stefani was the organization chair of the 7th Reversible Computation (RC'15) conference, held in Grenoble in July 2015. http://www.reversible-computation.org/2015

9.1.1.2. Member of the organizing committees

- Jean-Bernard Stefani was organiser chair for the 10th International Federated Conference on Distributed Computing Techniques (DisCoTec'15), held in Grenoble in June 2015. http://discotec2015.inria.fr
- Sophie Quinton was workshops chair for the 10th International Federated Conference on Distributed Computing Techniques (DisCoTec'15), held in Grenoble in June 2015. http://discotec2015.inria.fr

9.1.2. Scientific events selection

9.1.2.1. Chair of conference program committees

- Alain Girault was technical program committee co-chair for the ACM/IEEE International Conference on Embedded Software (EMSOFT'15), held in Amsterdam in October 2015. http://esweek.acm.org/esweek2015/emsoft
- Jean-Bernard Stefani was program committee co-chair for the 5th Reversible Computation conference (RC'15), held in Grenoble in July 2015. http://www.reversible-computation.org/2015
- Sophie Quinton was co-chair of the 6th International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems (WATERS'15), held in Lund in July 2015. http://waters2015.inria.fr

9.1.2.2. Member of the 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'15), the Design Automation Conference (DAC'15), and the International Symposium on Industrial Embedded Systems (SIES'15).
- Gregor Gössler served in the program committee of the 10th International Federated Conference on Distributed Computing Techniques (FORTE'15).

- Sophie Quinton served in the program committees of the 27th Euromicro Conference on Real-Time Systems (ECRTS'15), the 20th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA'15), the 2nd International Workshop on Multi-Objective Many-Core Design (MOMAC'15), and the 3rd International Workshop on Mixed Criticality Systems (WMC'15).
- Jean-Bernard Stefani served in the program committee of the 12th International Colloquium on Theoretical Aspects of Computing (ICTAC'15).

9.1.2.3. Reviewer

- Alain Girault reviewed an article for ECRTS'15.
- Gregor Gössler reviewed articles for ECRTS'15
- Xavier Nicollin reviewed an article for DATE'16.
- Sophie Quinton reviewed articles for DAC'15, FORTE'15, SIES'15, DATE'16.

9.1.3. Journal

9.1.3.1. Member of the editorial boards

- Alain Girault is 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

- Pascal Fradet reviewed articles for LNCS Transactions on Aspect-Oriented Software Development.
- Alain Girault reviewed an article for the Journal on Advances in Signal Processing.
- Gregor Gössler reviewed articles for ACM Transactions on Embedded Computing Systems (TECS), Acta Informatica, and the EURASIP Journal on Embedded Systems.
- Sophie Quinton reviewed articles for the journal Real-time Systems and the Transactions on Design Automation of Electronic Systems,
- Jean-Bernard Stefani reviewed articles for Theoretical Computer Science.

9.1.4. Scientific expertise

- Alain Girault was reviewer for an ERC grant.

9.1.5. 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.
- Alain Girault was head of science ("Délégué scientifique") of the INRIA Grenoble – Rhône-Alpes research center until June 2015.
- Alain Girault has been Vice Chair of the INRIA Evaluation Committee since September 2015. 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).
- Gregor Gössler has been leading a working group with the goal of fostering collaborations on the topic of formal methods for system design between the INRIA Grenoble – Rhône-Alpes research center and CEA Tech since September 2015.
- Jean-Bernard Stefani has been head of science of the INRIA Grenoble – Rhône-Alpes research center from July 2015. 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 relationships with our academic and industrial partners, hiring of the new junior researchers, ...).
- Jean-Bernard Stefani is co-director of I/O LAB, the joint research laboratory with Orange Lab.

9.2. Teaching - Supervision - Juries

9.2.1. Teaching

Licence : Pascal Fradet, Théorie des Langages 1, 18 HeqTD, niveau L3, Grenoble INP (Ensimag), France

Licence : Xavier Nicollin, Algorithmique et Structures de données 1, 22,5 HeqTD, niveau L3, Grenoble INP (Ensimag), France

Licence : Xavier Nicollin, Théorie des Langages 2, 18 HeqTD, niveau L3, Grenoble INP (Ensimag), France

Master : Jean-Bernard Stefani, Formal aspects of component software, 6h, MOSIG, Univ. Grenoble Alpes, France

9.2.2. Supervision

- PhD: Vagelis Bebelis, "Boolean Parametric Data Flow – Modeling – Analyses – Implementation", Univ. Grenoble Alpes, defended on February 26th 2015, co-advised by Pascal Fradet and Alain Girault.
- PhD: Dmitry Burlyaev, "Design, Optimization, and Formal Verification of Circuit Fault-Tolerance Techniques", Univ. Grenoble Alpes, defended on November 26th 2015, co-advised by Pascal Fradet and Alain Girault.
- PhD in progress: Yoann Geoffroy, "Towards a general causality analysis framework", Univ. Grenoble Alpes, since October 2013, advised by Gregor Gössler.
- PhD in progress: Christophe Prévot, "Early Performance assessment for evolving and variable Cyber-Physical Systems", Univ. Grenoble Alpes, since November 2015, advised by Alain Girault and Sophie Quinton.

9.2.3. Juries

- Alain Girault was member of the PhD jury of Paul-Antoine Arras (Univ. Bordeaux).
- Alain Girault was referee for the PhD of Julien Heulot (INSA Rennes).
- Jean-Bernard Stefani was president of the PhD jury of Jakub Zwolakowski (U. Paris-Diderot).
- Jean-Bernard Stefani was member of the HDR jury of Philippe Merle (U. Lille).
- Jean-Bernard Stefani was referee for the HDR of Luc Fabresse (U. Lille).
- Jean-Bernard Stefani was referee for the PhD of Vincent Lanore (ENS Lyon).
- Sophie Quinton was member of the jury for a "Maître de Conférences" position in Nancy.

9.3. Popularization

Gregor Gössler was the scientific organizer of an In'Tech seminar on *Embedded systems and liability. Failures, accidents: who's to blame* [4] featuring seven invited speakers and a panel. Led by the INRIA Grenoble - Rhône-Alpes research centre, the In'Tech seminars strive to offer an opportunity for SMEs, industrial operators and researchers to come together, thus promoting technology monitoring and the exchange of information among the various participants in the development of the region's Information Sciences and Technologies.

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° 4, pp. 1–43

[4] http://www.inria.fr/centre/grenoble/actualites/logiciel-embarque-et-responsabilite

[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° 11-12

[3] S. DJOKO DJOKO, R. DOUENCE, P. FRADET. *Aspects preserving properties*, in "Science of Computer Programming", 2012, vol. 77, n° 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. 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://hal.inria.fr/inria-00177117

[6] G. GÖSSLER, L. ASTEFANOAEI. *Blaming in component-based real-time systems*, in "Proceedings of the 14th International Conference on Embedded Software - EMSOFT'14", Delhi, India, ACM, October 2014 [*DOI :* 10.1145/2656045.2656048], https://hal.inria.fr/hal-01078214

[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. SCHMITT, J.-B. STEFANI. *Characterizing Contextual Equivalence in Calculi with Passivation*, in "Inf. Comput.", 2011, vol. 209, n° 11, pp. 1390-1433

[9] S. QUINTON, M. HANKE, R. ERNST. *Formal analysis of sporadic overload in real-time systems*, in "2012 Design, Automation & Test in Europe Conference & Exhibition, DATE 2012, Dresden, Germany, March, 2012", 2012, pp. 515–520, http://dx.doi.org/10.1109/DATE.2012.6176523

Publications of the year

Doctoral Dissertations and Habilitation Theses

[10] E. BEMPELIS. *Boolean Parametric Data Flow Modeling - Analyses - Implementation*, Université Grenoble Alpes, February 2015, https://tel.archives-ouvertes.fr/tel-01148698

[11] D. BURLYAEV. *Design, Optimization, and Formal Verification of Circuit Fault-Tolerance Techniques*, Inria Grenoble Rhône-Alpes, Université de Grenoble, November 2015, https://tel.archives-ouvertes.fr/tel-01253368

Articles in International Peer-Reviewed Journals

[12] 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, pp. 1-12 [*DOI :* 10.1109/TAC.2015.2478131], https://hal.archives-ouvertes.fr/hal-01197426

[13] S. GRAF, S. QUINTON. *Knowledge-based construction of distributed constrained systems*, in "International Journal on Software and Systems Modeling", January 2015 [*DOI :* 10.1007/S10270-014-0451-Z], https://hal.inria.fr/hal-01257059

[14] 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

International Conferences with Proceedings

[15] C. AUBERT. *An in-between "implicit" and "explicit" complexity: Automata*, in "DICE 2015 - Developments in Implicit Computational Complexity", Londres, United Kingdom, April 2015, https://hal.archives-ouvertes.fr/hal-01111737

[16] C. AUBERT, I. CRISTESCU. *Reversible Barbed Congruence on Configuration Structures*, in "8th Interaction and Concurrency Experience (ICE 2015) Satellite workshop of DisCoTec 2015", Grenoble, France, June 2015, https://hal.archives-ouvertes.fr/hal-01157974

[17] A. BOUAKAZ, P. FRADET, A. GIRAULT. *Symbolic Buffer Sizing for Throughput-Optimal Scheduling of Dataflow Graphs*, in "22nd IEEE Real-Time Embedded Technology & Applications Symposium (RTAS 2016)", Vienne, Austria, April 2016, https://hal.inria.fr/hal-01253168

[18] D. BURLYAEV, P. FRADET. *Formal Verification of Automatic Circuit Transformations for Fault-Tolerance*, in "Formal Methods in Computer-Aided Design (FMCAD 2015)", Austin, Texas, United States, September 2015, https://hal.inria.fr/hal-01253127

[19] D. BURLYAEV, P. FRADET, A. GIRAULT. *Automatic Time-Redundancy Transformation for Fault-Tolerant Circuits*, in "23rd ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, FPGA'15", Monterey, United States, February 2015 [*DOI :* 10.1145/2684746.2689058], https://hal.inria.fr/hal-01095747

[20] D. BURLYAEV, P. FRADET, A. GIRAULT. *Time-redundancy transformations for adaptive fault-tolerant circuits*, in "2015 NASA/ESA Conference on Adaptive Hardware and Systems (AHS)", Montreal, Canada, June 2015 [*DOI :* 10.1109/AHS.2015.7231164], https://hal.inria.fr/hal-01253111

[21] B. CLAUDEL, Q. SABAH, J.-B. STEFANI. *Simple Isolation for an Actor Abstract Machine*, in "35th IFIP International Conference on Formal Techniques for Distributed Objects, Components and Systems", Montbonnot, France, Lecture Notes in Computer Science, Springer, June 2015, vol. 9039 [*DOI :* 10.1007/978-3-319-19195-9_14], https://hal.inria.fr/hal-01219656

[22] O. GETTINGS, S. QUINTON, R. DAVIS. *Mixed criticality systems with weakly-hard constraints*, in "International Conference on Real Time and Networks Systems", Lille, France, November 2015 [*DOI :* 10.1145/2834848.2834850], https://hal.inria.fr/hal-01257067

[23] G. GÖSSLER, J.-B. STEFANI. *Fault Ascription in Concurrent Systems*, in "Trustworthy Global Computing", Madrid, Spain, P. GANTY, M. LORETI (editors), TGC 2015, Springer, 2015, vol. 9533, 16 p. [*DOI :* 10.1007/978-3-319-28766-9 6], https://hal.inria.fr/hal-01246485

[24] W.-T. SUN, A. GIRAULT, G. DELAVAL. *A formal approach for the synthesis and implementation of fault-tolerant industrial embedded systems*, in "SIES'2015: 10th IEEE International Symposium on Industrial Embedded Systems", Siegen, Germany, June 2015, https://hal.inria.fr/hal-01165686

[25] S. WANG, Y. GEOFFROY, G. GÖSSLER, O. SOKOLSKY, I. LEE. *A Hybrid Approach to Causality Analysis*, in "RV 2015 - 6th International Conference on Runtime Verification", Vienna, Austria, LNCS, September 2015, vol. 9333 [*DOI :* 10.1007/978-3-319-23820-3_16], https://hal.inria.fr/hal-01211607

[26] X. WENBO, Z. A. H. HAMMADEH, K. ALEXANDER, S. QUINTON, R. ERNST. *Improved Deadline Miss Models for Real-Time Systems Using Typical Worst-Case Analysis*, in "Euromicro Conference on Real-Time Systems", Lund, Sweden, July 2015 [*DOI :* 10.1109/ECRTS.2015.29], https://hal.inria.fr/hal-01257065

Books or Proceedings Editing

[27] A. GIRAULT, G. NAN (editors). *2015 International Conference on Embedded Software, EMSOFT'15*, IEEE, Amsterdam, Netherlands, October 2015, https://hal.archives-ouvertes.fr/hal-01271544

[28] J. KRIVINE, J.-B. STEFANI (editors). *Reversible Computation*, Lecture Notes in Computer Science, Springer, Grenoble, France, July 2015, vol. 9138 [*DOI :* 10.1007/978-3-319-20860-2], https://hal.inria.fr/hal-01246644

Research Reports

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

[30] G. GÖSSLER, J.-B. STEFANI. *Fault Ascription in Concurrent Systems*, Inria Grenoble - Rhône-Alpes, September 2015, n^{o} RR-8772, https://hal.inria.fr/hal-01197486

Other Publications

[31] C. AUBERT, I. CRISTESCU. *Contextual equivalences in configuration structures and reversibility*, November 2015, working paper or preprint, https://hal.archives-ouvertes.fr/hal-01229408

References in notes

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

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

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

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

[36] 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

[37] 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^{o} 3, pp. 229–245

[38] 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

[39] 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^{o} 4

[40] 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^{o} 3

[41] 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

[42] 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

[43] 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^{o} 1

[44] 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

[45] 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

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

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

[48] 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

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

[50] D. BURLYAEV, P. FRADET, A. GIRAULT. *Procédé de fabrication automatisée d'un circuit électronique adapté pour détecter ou masquer des fautes par redondance temporelle, programme d'ordinateur et circuit électronique associés*, June 2014, n^{o} 1456080, https://hal.inria.fr/hal-01096054

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

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

[53] COQ DEVELOPMENT TEAM. *The Coq proof assistant, 1989-2014*, http://coq.inria.fr/

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

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

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

[57] 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

[58] G. DELAVAL, H. MARCHAND, É. RUTTEN. *Contracts for Modular Discrete Controller Synthesis*, in "ACM International Conference on Languages, Compilers, and Tools for Embedded Systems (LCTES 2010)", Stockholm, Sweden, April 2010, http://pop-art.inrialpes.fr/people/delaval/pub/lctes2010.pdf

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

[60] 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^{o} 1

[61] 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

[62] 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

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

[64] 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://www.computer.org/portal/web/csdl/doi/10.1109/TDSC.2008.50

[65] A. GIRAULT, É. RUTTEN. *Automating the Addition of Fault Tolerance with Discrete Controller Synthesis*, in "Formal Methods in System Design", October 2009, vol. 35, n^{o} 2, pp. 190–225, http://www.springerlink.com/content/w726262156h4822j

[66] 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

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

[68] 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

[69] N. HALBWACHS, P. CASPI, P. RAYMOND, D. PILAUD. *The Synchronous Data-Flow Programming Language Lustre*, in "Proceedings of the IEEE", September 1991, vol. 79, n° 9, pp. 1305–1320

[70] 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

[71] Z. A. H. HAMMADEH, S. QUINTON, R. ERNST. *Extending typical worst-case analysis using response-time dependencies to bound deadline misses*, in "14th International Conference on Embedded Software 2014 (EMSOFT)", New Delhi, India, October 2014 [*DOI :* 10.1145/2656045.2656059], https://hal.inria.fr/hal-01097621

[72] 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

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

[74] 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

[75] 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.

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

[77] 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

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

[79] H. MARCHAND, P. BOURNAI, M. LE BORGNE, P. LE GUERNIC. *Synthesis of Discrete-Event Controllers based on the Signal Environment*, in "Discrete Event Dynamical System: Theory and Applications", October 2000, vol. 10, n° 4, pp. 325–346

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

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

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

[83] 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^{o} 1, pp. 206–230

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

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

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

[87] M. SHEERAN. *muFP, A Language for VLSI Design*, in "LISP and Functional Programming", 1984, pp. 104–112

[88] 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

[89] W.-T. SUN, Z. A. SALCIC, A. GIRAULT, A. MALIK. *libDGALS: A library-based approach to design dynamic GALS systems*, in "International Symposium on Industrial Embedded Systems, SIES'14", Pisa, Italy, IEEE, June 2014, pp. 104–111

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

[91] 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

[92] 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^{o} 3

[93] 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