IN PARTNERSHIP WITH:
Institut national des sciences appliquées de Rennes

Université Rennes 1

Activity Report 2016

Project-Team DIVERSE

Diversity-centric Software Engineering

IN COLLABORATION WITH: Institut de recherche en informatique et systèmes aléatoires (IRISA)

RESEARCH CENTER
Rennes - Bretagne-Atlantique

THEME
Distributed programming and Software engineering

Table of contents

Project-Team DIVERSE

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

Keywords:

Computer Science and Digital Science:

1.2.1. - Dynamic reconfiguration
2.1.2. - Object-oriented programming
2.1.10. - Domain-specific languages
2.5. - Software engineering
2.5.2. - Component-based Design
2.5.3. - Empirical Software Engineering
2.6.2. - Middleware
4.4. - Security of equipment and software
4.8. - Privacy-enhancing technologies

Other Research Topics and Application Domains:

3.1. - Sustainable development
3.1.1. - Resource management
6.1. - Software industry
6.1.1. - Software engineering
6.4. - Internet of things
6.5. - Information systems
6.6. - Embedded systems
8.1.2. - Sensor networks for smart buildings
9.4.1. - Computer science

1. Members

Research Scientist

Benoit Baudry [Team leader, Inria, Researcher, HDR]

Faculty Members

Mathieu Acher [Univ. Rennes I, Associate Professor]
Olivier Barais [Univ. Rennes I, Professor, HDR]
Arnaud Blouin [INSA Rennes, Associate Professor]
Johann Bourcier [Univ. Rennes I, Associate Professor]
Benoit Combemale [Univ. Rennes I, Associate Professor, HDR]
Jean-Marc Jezequel [Univ. Rennes I, Professor, HDR]
Noel Plouzeau [Univ. Rennes I, Associate Professor]

Engineers

Simon Allier [Inria, until Nov 2016]
Fabien Coulon [Univ. Rennes I]
Andre Elie [Univ. Rennes I]
Jose Angel Galindo Duarte [Inria, until Sep 2016]
Caroline Landry [Inria, from Dec 2016]
Manuel Leduc [Inria]

Dorian Leroy [Univ. Rennes I]
Jean Parpaillon [Inria, until Aug 2016]
Maxime Tricoire [Inria]

PhD Students

Sana Ben Nasr [Inria, until Apr 2016]
Mohamed Boussaa [Univ. Rennes I]
Kevin Corre [Orange Labs, granted by CIFRE]
Thomas Degueule [Univ. Rennes I, until Nov 2016]
Alejandro Gomez Boix [Inria, from Nov 2016]
Nicolas Harrand [Inria, from Nov 2016]
Pierre Laperdrix [Univ. Rennes I]
Gwendal Le Moulec [INSA Rennes]
David Mendez Acuna [Inria, until Oct 2016]
Johan Pelay [Inst. de Recherche Technologique B-COM, until Mar 2016]
Quentin Plazar [Inria]
Oscar Luis Vera Perez [Inria]
Alexandre Rio [OKWind]
Jean-Emile Dartois [B-COM]
Youssou Ndaye [Orange]
Marcelino Rodriguez Cancio [Univ. Rennes I]
Paul Temple [Univ. Rennes I]
Kwaku Yeboah-Antwi [Inria, until Sep 2016]
Guillaume Becan [Univ. Rennes I, intern]

Visiting Scientists

Marwa Hentati [National School of Engineers of Sfax (ENIS), from Apr 2016 until May 2016]
Akram Kammoun ["Research laboratory on Development and Control of Distributed Applications (ReD-CAD)" Sfax Tunisia, from May 2016 until Jun 2016]

Administrative Assistant

Tifenn Donguy [CNRS]

Others

Nassim Aliche [Univ. Rennes I, intern, from Jul 2016 until Aug 2016]
Marvin Billaud [Univ. Rennes I, intern, from May 2016 until Aug 2016]
Clementine Delambily [Inria, intern, from May 2016 until Aug 2016]
Pierre Duros [Univ. Rennes I, intern, from Jul 2016 until Sep 2016]
Laureline Guerin [Inria, intern, until Aug 2016]
Axel Halin [Univ. Rennes I, intern, from Sep 2016]
Remi Hauperpin Bonneau [Univ. Rennes I, intern, from May 2016 until Aug 2016]
Pierre Le Gall [Univ. Rennes I, intern, from Feb 2016 until Jun 2016]
Gurvan Le Guernic [DGA, research engineer]
Stephane Mangin [Univ. Rennes I, intern, until Aug 2016]
Malo Manini [Univ. Rennes I, intern, from Jun 2016 until Aug 2016]
Bruno Merciol [Univ. Rennes I, intern, until Jun 2016]
Vikas Mishra [Inria, intern, from May 2016 until Jul 2016]
Leo Noel-Baron [Univ. Rennes I, intern, from May 2016 until Jul 2016]
Alexandre Nuttinck [Univ. Rennes I, intern, from Sep 2016]

2. Overall Objectives

2.1. Overall objectives

DIVERSE's research agenda is in the area of software engineering. In this broad domain we develop models, methodologies and theories to address the challenges raised by the emergence of several forms of diversity in the design, deployment and evolution of software-intensive systems. The emergence of software diversity is an essential phenomenon in all application domains that we investigate with our industrial partners. These application domains range from complex systems such as systems of systems (in collaboration with Thales and DGA) and Instrumentation and Control (with EDF) to pervasive combinations of Internet of Things and Internet of Services (with TellU and Software AG) and tactical information systems (with the firefighter department). Even if today these systems are apparently radically different, we envision a strong convergence of the scientific principles underpinning their construction and validation towards **flexible and open yet dependable systems**. In particular, we see that the required flexibility and openness raise challenges for the software layer of these systems that must deal with four dimensions of diversity: the **diversity of languages** used by the stakeholders involved in the construction of these systems; the **diversity of features** required by the different customers; the **diversity of runtime environments** in which software has to run and adapt; the **diversity of implementations** that are necessary for resilience through redundancy.

In this context, the major software engineering challenge consists in handling **diversity** from variability in requirements and design to heterogeneous and dynamic execution environments. In particular this requires considering that the software system must adapt, in unpredictable ways, to changes in the requirements and environment. Conversely, explicitly handling of diversity is a great opportunity to allow software to spontaneously explore alternative design solutions. Concretely, we want to provide software engineers with the ability:

- to characterize an 'envelope' of possible variations
- to compose 'envelopes' (to discover new macro envelopes in an opportunistic manner)
- to dynamically synthesize software inside a given envelop

The major scientific objective that we must achieve to provide such mechanisms for software engineering is synthesized below

Scientific objective for DIVERSE: Automatically **compose and synthesize software diversity** from design to runtime to **address unpredictable evolutions of software-intensive systems**

Software product lines and associated variability modeling formalisms represent an essential aspect of software diversity, which we already explored in the past and that represent a major foundation of DIVERSE's research agenda. However, DIVERSE also exploits other foundations to handle new forms of diversity: type theory and models of computation for the composition of languages; distributed algorithms and pervasive computation to handle the diversity of execution platforms; functional and qualitative randomized transformations to synthesize diversity for robust systems.

3. Research Program

3.1. Scientific background

3.1.1. Model-driven engineering

Model-Driven Engineering (MDE) aims at reducing the accidental complexity associated with developing complex software-intensive systems (e.g., use of abstractions of the problem space rather than abstractions of the solution space) [117]. It provides DIVERSE with solid foundations to specify, analyze and reason about the different forms of diversity that occur through the development lifecycle. A primary source of accidental complexity is the wide gap between the concepts used by domain experts and the low-level abstractions

provided by general-purpose programming languages [88]. MDE approaches address this problem through modeling techniques that support separation of concerns and automated generation of major system artifacts from models (*e.g.,* test cases, implementations, deployment and configuration scripts). In MDE, a model describes an aspect of a system and is typically created or derived for specific development purposes [70]. Separation of concerns is supported through the use of different modeling languages, each providing constructs based on abstractions that are specific to an aspect of a system. MDE technologies also provide support for manipulating models, for example, support for querying, slicing, transforming, merging, and analyzing (including executing) models. Modeling languages are thus at the core of MDE, which participates to the development of a sound *Software Language Engineering* [1], including an unified typing theory that integrate models as first class entities [120].

Incorporating domain-specific concepts and high-quality development experience into MDE technologies can significantly improve developer productivity and system quality. Since the late nineties, this realization has led to work on MDE language workbenches that support the development of domain-specific modeling languages (DSMLs) and associated tools (*e.g.,* model editors and code generators). A DSML provides a bridge between the field in which domain experts work and the implementation (programming) field. Domains in which DSMLs have been developed and used include, among others, automotive, avionics, and the emerging cyber-physical systems. A study performed by Hutchinson et al. [94] provides some indications that DSMLs can pave the way for wider industrial adoption of MDE.

More recently, the emergence of new classes of systems that are complex and operate in heterogeneous and rapidly changing environments raises new challenges for the software engineering community. These systems must be adaptable, flexible, reconfigurable and, increasingly, self-managing. Such characteristics make systems more prone to failure when running and thus the development and study of appropriate mechanisms for continuous design and run-time validation and monitoring are needed. In the MDE community, research is focused primarily on using models at design, implementation, and deployment stages of development. This work has been highly productive, with several techniques now entering a commercialization phase. As software systems are becoming more and more dynamic, the use of model-driven techniques for validating and monitoring run-time behavior is extremely promising [102].

3.1.2. Variability modeling

While the basic vision underlying *Software Product Lines* (SPL) can probably be traced back to David Parnas seminal article [110] on the Design and Development of Program Families, it is only quite recently that SPLs are emerging as a paradigm shift towards modeling and developing software system families rather than individual systems [108]. SPL engineering embraces the ideas of mass customization and software reuse. It focuses on the means of efficiently producing and maintaining multiple related software products, exploiting what they have in common and managing what varies among them.

Several definitions of the *software product line* concept can be found in the research literature. Clements *et al.* define it as *a set of software-intensive systems sharing a common, managed set of features that satisfy the specific needs of a particular market segment or mission and are developed from a common set of core assets in a prescribed way* [107]. Bosch provides a different definition [76]: *A SPL consists of a product line architecture and a set of reusable components designed for incorporation into the product line architecture. In addition, the PL consists of the software products developed using the mentioned reusable assets*. In spite of the similarities, these definitions provide different perspectives of the concept: *market-driven*, as seen by Clements *et al.*, and *technology-oriented* for Bosch.

SPL engineering is a process focusing on capturing the *commonalities* (assumptions true for each family member) and *variability* (assumptions about how individual family members differ) between several software products [82]. Instead of describing a single software system, a SPL model describes a set of products in the same domain. This is accomplished by distinguishing between elements common to all SPL members, and those that may vary from one product to another. Reuse of core assets, which form the basis of the product line, is key to productivity and quality gains. These core assets extend beyond simple code reuse and may

[1] See http://planet-sl.org

include the architecture, software components, domain models, requirements statements, documentation, test plans or test cases.

The SPL engineering process consists of two major steps:

1. **Domain Engineering**, or *development for reuse*, focuses on core assets development.
2. **Application Engineering**, or *development with reuse*, addresses the development of the final products using core assets and following customer requirements.

Central to both processes is the management of **variability** across the product line [90]. In common language use, the term *variability* refers to *the ability or the tendency to change*. Variability management is thus seen as the key feature that distinguishes SPL engineering from other software development approaches [77]. Variability management is thus growingly seen as the cornerstone of SPL development, covering the entire development life cycle, from requirements elicitation [122] to product derivation [127] to product testing [106], [105].

Halmans *et al.* [90] distinguish between *essential* and *technical* variability, especially at requirements level. Essential variability corresponds to the customer's viewpoint, defining what to implement, while technical variability relates to product family engineering, defining how to implement it. A classification based on the dimensions of variability is proposed by Pohl *et al.* [112]: beyond **variability in time** (existence of different versions of an artifact that are valid at different times) and **variability in space** (existence of an artifact in different shapes at the same time) Pohl *et al.* claim that variability is important to different stakeholders and thus has different levels of visibility: **external variability** is visible to the customers while **internal variability**, that of domain artifacts, is hidden from them. Other classification proposals come from Meekel *et al.* [100] (feature, hardware platform, performances and attributes variability) or Bass *et al.* [68] who discuss about variability at the architectural level.

Central to the modeling of variability is the notion of *feature*, originally defined by Kang *et al.* as: *a prominent or distinctive user-visible aspect, quality or characteristic of a software system or systems* [96]. Based on this notion of *feature*, they proposed to use a *feature model* to model the variability in a SPL. A feature model consists of a *feature diagram* and other associated information: *constraints* and *dependency rules*. Feature diagrams provide a *graphical tree-like notation depicting the hierarchical organization of high level product functionalities* represented as features. The root of the tree refers to the complete system and is progressively decomposed into more refined features (tree nodes). Relations between nodes (features) are materialized by *decomposition edges* and *textual constraints*. Variability can be expressed in several ways. Presence or absence of a feature from a product is modeled using *mandatory* or *optional features*. Features are graphically represented as rectangles while some graphical elements (e.g., unfilled circle) are used to describe the variability (e.g., a feature may be optional).

Features can be organized into *feature groups*. Boolean operators *exclusive alternative (XOR)*, *inclusive alternative (OR)* or *inclusive (AND)* are used to select one, several or all the features from a feature group. Dependencies between features can be modeled using *textual constraints*: *requires* (presence of a feature requires the presence of another), *mutex* (presence of a feature automatically excludes another). Feature attributes can be also used for modeling quantitative (e.g., numerical) information. Constraints over attributes and features can be specified as well.

Modeling variability allows an organization to capture and select which version of which variant of any particular aspect is wanted in the system [77]. To implement it cheaply, quickly and safely, redoing by hand the tedious weaving of every aspect is not an option: some form of automation is needed to leverage the modeling of variability [72], [84]. Model Driven Engineering (MDE) makes it possible to automate this weaving process [95]. This requires that models are no longer informal, and that the weaving process is itself described as a program (which is as a matter of facts an executable meta-model [103]) manipulating these models to produce for instance a detailed design that can ultimately be transformed to code, or to test suites [111], or other software artifacts.

3.1.3. Component-based software development

Component-based software development [121] aims at providing reliable software architectures with a low cost of design. Components are now used routinely in many domains of software system designs: distributed systems, user interaction, product lines, embedded systems, etc. With respect to more traditional software artifacts (e.g., object oriented architectures), modern component models have the following distinctive features [83]: description of requirements on services required from the other components; indirect connections between components thanks to ports and connectors constructs [98]; hierarchical definition of components (assemblies of components can define new component types); connectors supporting various communication semantics [80]; quantitative properties on the services [75].

In recent years component-based architectures have evolved from static designs to dynamic, adaptive designs (e.g., SOFA [80], Palladio [73], Frascati [104]). Processes for building a system using a statically designed architecture are made of the following sequential lifecycle stages: requirements, modeling, implementation, packaging, deployment, system launch, system execution, system shutdown and system removal. If for any reason after design time architectural changes are needed after system launch (e.g., because requirements changed, or the implementation platform has evolved, etc) then the design process must be reexecuted from scratch (unless the changes are limited to parameter adjustment in the components deployed).

Dynamic designs allow for *on the fly* redesign of a component based system. A process for dynamic adaptation is able to reapply the design phases while the system is up and running, without stopping it (this is different from stop/redeploy/start). This kind of process supports *chosen adaptation*, when changes are planned and realized to maintain a good fit between the needs that the system must support and the way it supports them [97]. Dynamic component-based designs rely on a component meta-model that supports complex life cycles for components, connectors, service specification, etc. Advanced dynamic designs can also take platform changes into account at run-time, without human intervention, by adapting themselves [81], [124]. Platform changes and more generally environmental changes trigger *imposed adaptation*, when the system can no longer use its design to provide the services it must support. In order to support an eternal system [74], dynamic component based systems must separate architectural design and platform compatibility. This requires support for heterogeneity, since platform evolutions can be partial.

The Models@runtime paradigm denotes a model-driven approach aiming at taming the complexity of dynamic software systems. It basically pushes the idea of reflection one step further by considering the reflection layer as a real model "something simpler, safer or cheaper than reality to avoid the complexity, danger and irreversibility of reality [115]". In practice, component-based (and/or service-based) platforms offer reflection APIs that make it possible to introspect the system (which components and bindings are currently in place in the system) and dynamic adaptation (by applying CRUD operations on these components and bindings). While some of these platforms offer rollback mechanisms to recover after an erroneous adaptation, the idea of Models@runtime is to prevent the system from actually enacting an erroneous adaptation. In other words, the "model at run-time" is a reflection model that can be uncoupled (for reasoning, validation, simulation purposes) and automatically resynchronized.

Heterogeneity is a key challenge for modern component based system. Until recently, component based techniques were designed to address a specific domain, such as embedded software for command and control, or distributed Web based service oriented architectures. The emergence of the Internet of Things paradigm calls for a unified approach in component based design techniques. By implementing an efficient separation of concern between platform independent architecture management and platform dependent implementations, *Models@runtime* is now established as a key technique to support dynamic component based designs. It provides DIVERSE with an essential foundation to explore an adaptation envelop at run-time.

Search Based Software Engineering [92] has been applied to various software engineering problems in order to support software developers in their daily work. The goal is to automatically explore a set of alternatives and assess their relevance with respect to the considered problem. These techniques have been applied to craft software architecture exhibiting high quality of services properties [89]. Multi Objectives Search based techniques [86] deal with optimization problem containing several (possibly conflicting) dimensions to optimize. These techniques provide DIVERSE with the scientific foundations for reasoning and efficiently exploring an envelope of software configurations at run-time.

3.1.4. Validation and verification

Validation and verification (V&V) theories and techniques provide the means to assess the validity of a software system with respect to a specific correctness envelop. As such, they form an essential element of DIVERSE's scientific background. In particular, we focus on model-based V&V in order to leverage the different models that specify the envelop at different moments of the software development lifecycle.

Model-based testing consists in analyzing a formal model of a system (*e.g.*, activity diagrams, which capture high-level requirements about the system, statecharts, which capture the expected behavior of a software module, or a feature model, which describes all possible variants of the system) in order to generate test cases that will be executed against the system. Model-based testing [123] mainly relies on model analysis, constraint solving [85] and search-based reasoning [99]. DIVERSE leverages in particular the applications of model-based testing in the context of highly-configurable systems and [125] interactive systems [101] as well as recent advances based on diversity for test cases selection [93].

Nowadays, it is possible to simulate various kinds of models. Existing tools range from industrial tools such as Simulink, Rhapsody or Telelogic to academic approaches like Omega [109], or Xholon [2]. All these simulation environments operate on homogeneous environment models. However, to handle diversity in software systems, we also leverage recent advances in heterogeneous simulation. Ptolemy [79] proposes a common abstract syntax, which represents the description of the model structure. These elements can be decorated using different directors that reflect the application of a specific model of computation on the model element. Metropolis [69] provides modeling elements amenable to semantically equivalent mathematical models. Metropolis offers a precise semantics flexible enough to support different models of computation. ModHel'X [91] studies the composition of multi-paradigm models relying on different models of computation.

Model-based testing and simulation are complemented by runtime fault-tolerance through the automatic generation of software variants that can run in parallel, to tackle the open nature of software-intensive systems. The foundations in this case are the seminal work about N-version programming [67], recovery blocks [113] and code randomization [71], which demonstrated the central role of diversity in software to ensure runtime resilience of complex systems. Such techniques rely on truly diverse software solutions in order to provide systems with the ability to react to events, which could not be predicted at design time and checked through testing or simulation.

3.1.5. Empirical software engineering

The rigorous, scientific evaluation of DIVERSE's contributions is an essential aspect of our research methodology. In addition to theoretical validation through formal analysis or complexity estimation, we also aim at applying state-of-the-art methodologies and principles of empirical software engineering. This approach encompasses a set of techniques for the sound validation contributions in the field of software engineering, ranging from statistically sound comparisons of techniques and large-scale data analysis to interviews and systematic literature reviews [118], [116]. Such methods have been used for example to understand the impact of new software development paradigms [78]. Experimental design and statistical tests represent another major aspect of empirical software engineering. Addressing large-scale software engineering problems often requires the application of heuristics, and it is important to understand their effects through sound statistical analyses [66].

3.2. Research axis

Figure 1 illustrates the four dimensions of software diversity, which form the core research axis of DIVERSE: the **diversity of languages** used by the stakeholders involved in the construction of these systems; the **diversity of features** required by the different customers; the **diversity of runtime environments** in which software has to run and adapt; the **diversity of implementations** that are necessary for resilience through redundancy. These four axis share and leverage the scientific and technological results developed in the area of model-driven engineering in the last decade. This means that all our research activities are founded on sound abstractions to

[2] http://www.primordion.com/Xholon/

reason about specific aspects of software systems, compose different perspectives and automatically generate parts of the system.

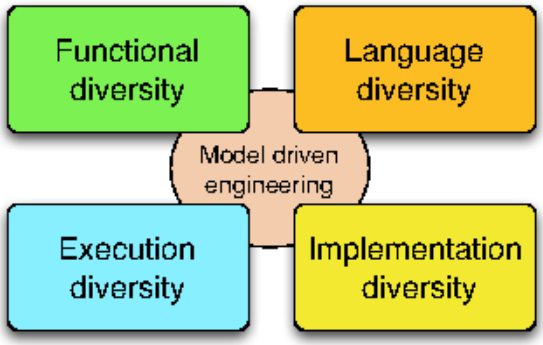

Figure 1. The four research axis of DIVERSE, which rely on a MDE scientific background

3.2.1. Software Language Engineering

The engineering of systems involves many different stakeholders, each with their own domain of expertise. Hence more and more organizations are adopting Domain Specific Modeling Languages (DSMLs) to allow domain experts to express solutions directly in terms of relevant domain concepts [117], [88]. This new trend raises new challenges about designing DSMLs, evolving a set of DSMLs and coordinating the use of multiple DSLs for both DSL designers and DSL users.

3.2.1.1. Challenges

Reusability of software artifacts is a central notion that has been thoroughly studied and used by both academics and industrials since the early days of software construction. Essentially, designing reusable artifacts allows the construction of large systems from smaller parts that have been separately developed and validated, thus reducing the development costs by capitalizing on previous engineering efforts. However, it is still hardly possible for language designers to design typical language artifacts (e.g. language constructs, grammars, editors or compilers) in a reusable way. The current state of the practice usually prevents the reusability of language artifacts from one language to another, consequently hindering the emergence of real engineering techniques around software languages. Conversely, concepts and mechanisms that enable artifacts reusability abound in the software engineering community.

Variability in modeling languages occur in the definition of the abstract and concrete syntax as well as in the specification of the language's semantics. The major challenges met when addressing the need for variability are: (i) set principles for modeling language units that support the modular specification of a modeling language; and (ii) design mechanisms to assemble these units in a complete language, according to the set of authorized variation points for the modeling language family.

A new generation of complex software-intensive systems (for example smart health support, smart grid, building energy management, and intelligent transportation systems) presents new opportunities for leveraging modeling languages. The development of these systems requires expertise in diverse domains. Consequently, different types of stakeholders (e.g., scientists, engineers and end-users) must work in a coordinated manner on various aspects of the system across multiple development phases. DSMLs can be used to support the work of domain experts who focus on a specific system aspect, but they can also provide the means for coordinating work across teams specializing in different aspects and across development phases. The support and integration of DSMLs leads to what we call **the globalization of modeling languages**, *i.e.* the use of multiple languages

for the coordinated development of diverse aspects of a system. One can make an analogy with world globalization in which relationships are established between sovereign countries to regulate interactions (e.g., travel and commerce related interactions) while preserving each country's independent existence.

3.2.1.2. Scientific objectives

We address reuse and variability challenges through the investigation of the time-honored concepts of substitutability, inheritance and components, evaluate their relevance for language designers and provide tools and methods for their inclusion in software language engineering. We will develop novel techniques for the modular construction of language extensions with the support of model syntactical variability. From the semantics perspective, we investigate extension mechanisms for the specification of variability in operational semantics, focusing on static introduction and heterogeneous models of computation. The definition of variation points for the three aspects of the language definition provides the foundations for the novel concept Language Unit (LU) as well as suitable mechanisms to compose such units.

We explore the necessary breakthrough in software languages to support modeling and simulation of heterogeneous and open systems. This work relies on the specification of executable domain specific modeling languages (DSMLs) to formalize the various concerns of a software-intensive system, and of models of computation (MoCs) to explicitly model the concurrency, time and communication of such DSMLs. We develop a framework that integrates the necessary foundations and facilities for designing and implementing executable and concurrent domain-specific modeling languages. It also provides unique features to specify composition operators between (possibly heterogeneous) DSMLs. Such specifications are amenable to support the edition, execution, graphical animation and analysis of heterogeneous models. The objective is to provide both a significant improvement of MoCs and DSMLs design and implementation; and the simulation based validation and verification of complex systems.

We see an opportunity for the automatic diversification of programs' computation semantics, for example through the diversification of compilers or virtual machines. The main impact of this artificial diversity is to provide flexible computation and thus ease adaptation to different execution conditions. A combination of static and dynamic analysis could support the identification of what we call *plastic computation zones* in the code. We identify different categories of such zones: (i) areas in the code in which the order of computation can vary (e.g., the order in which a block of sequential statements is executed); (ii) areas that can be removed, keeping the essential functionality [119] (e.g., skip some loop iterations); (iii) areas that can replaced by alternative code (e.g., replace a try-catch by a return statement). Once we know which zones in the code can be randomized, it is necessary to modify the model of computation to leverage the computation plasticity. This consists in introducing variation points in the interpreter to reflect the diversity of models of computation. Then, the choice of a given variation is performed randomly at run-time.

3.2.2. Variability Modeling and Engineering

The systematic modeling of variability in software systems has emerged as an effective approach to document and reason about software evolutions and heterogeneity (*cf.* Section 3.1.2). Variability modeling characterizes an "envelope" of possible software variations. The industrial use of variability models and their relation to software artifact models require a complete engineering framework, including composition, decomposition, analysis, configuration and artifact derivation, refactoring, re-engineering, extraction, and testing. This framework can be used both to tame imposed diversity and to manage chosen diversity.

3.2.2.1. Challenges

A fundamental problem is that the **number of variants** can be exponential in the number of options (features). Already with 300 boolean configuration options, approximately 10^{90} configurations exist – more than estimated count of atoms in the universe. Domains like automotive or operating systems have to manage more than 10000 options (e.g., Linux). Practitioners face the challenge of developing billions of variants. It is easy to forget a necessary constraint, leading to the synthesis of unsafe variants, or to under-approximate the capabilities of the software platform. Scalable modelling techniques are therefore crucial to specify and reason about a very large set of variants.

Model-driven development supports two ways to deal with the increasing number of concerns in complex systems: (1) multi-view modeling, *i.e.* when modeling each concern separately, and variability modeling. However, there is little support to combine both approaches consistently. Techniques to integrate both approaches will enable the construction of a consistent set of views and variation points in each view.

The design, construction and maintenance of software families have a major impact on **software testing**. Among the existing challenges, we can cite: the selection of test cases for a specific variant; the evolution of test suites with integration of new variants; the combinatorial explosion of the number of software configurations to be tested. Novel model-based techniques for test generation and test management in a software product line context are needed to overcome state-of-the-art limits we already observed in some projects.

3.2.2.2. Scientific objectives

We aim at developing scalable techniques to automatically analyze variability models and their interactions with other views on the software intensive system (requirements, architecture, design). These techniques provide two major advancements in the state of the art: (1) an extension of the semantics of variability models in order to enable the definition of attributes (*e.g.*, cost, quality of service, effort) on features and to include these attributes in the reasoning; (2) an assessment of the consistent specification of variability models with respect to system views (since variability is orthogonal to system modeling, it is currently possible to specify the different models in ways that are semantically meaningless). The former aspect of analysis is tackled through constraint solving and finite-domain constraint programming, while the latter aspect is investigated through automatic search-based techniques (similar to genetic algorithms) for the exploration of the space of interaction between variability and view models.

We aim to develop procedures to reverse engineer dependencies and features' sets from existing software artefacts – be it source code, configuration files, spreadsheets (e.g., product comparison matrices) or requirements. We expect to scale up (e.g., for extracting a very large number of variation points) and guarantee some properties (e.g., soundness of configuration semantics, understandability of ontological semantics). For instance, when building complex software-intensive systems, textual requirements are captured in very large quantities of documents. In this context, adequate models to formalize the organization of requirements documents and automated techniques to support impact analysis (in case of changes in the requirements) have to be developed.

We aim at developing sound methods and tools to integrate variability management in model-based testing activities. In particular, we will leverage requirement models as an essential asset to establish formal relations between variation points and test models. These relations will form the basis for novel algorithms that drive the systematic selection of test configurations that satisfy well-defined test adequacy criteria as well as the generation of test cases for a specific product in the product line.

3.2.3. Heterogeneous and dynamic software architectures

Flexible yet dependable systems have to cope with heterogeneous hardware execution platforms ranging from smart sensors to huge computation infrastructures and data centers. Evolutions range from a mere change in the system configuration to a major architectural redesign, for instance to support addition of new features or a change in the platform architecture (new hardware is made available, a running system switches to low bandwidth wireless communication, a computation node battery is running low, etc). In this context, we need to devise formalisms to reason about the impact of an evolution and about the transition from one configuration to another. It must be noted that this axis focuses on the use of models to drive the evolution from design time to run-time. Models will be used to (i) systematically define predictable configurations and variation points through which the system will evolve; (ii) develop behaviors necessary to handle unpredicted evolutions.

3.2.3.1. Challenges

The main challenge is to provide new homogeneous architectural modelling languages and efficient techniques that enable continuous software reconfiguration to react to changes. This work handles the challenges of handling the diversity of runtime infrastructures and managing the cooperation between different stakeholders. More specifically, the research developed in this axis targets the following dimensions of software diversity.

Platform architectural heterogeneity induces a first dimension of imposed diversity (type diversity). Platform reconfigurations driven by changing resources define another dimension of diversity (deployment diversity). To deal with these imposed diversity problems, we will rely on model based runtime support for adaptation, in the spirit of the dynamic distributed component framework developed by the Triskell team. Since the runtime environment composed of distributed, resource constrained hardware nodes cannot afford the overhead of traditional runtime adaptation techniques, we investigate the design of novel solutions relying on models@runtime and on specialized tiny virtual machines to offer resource provisioning and dynamic reconfigurations. In the next two years this research will be supported by the InfraJVM project.

Diversity can also be an asset to optimize software architecture. Architecture models must integrate multiple concerns in order to properly manage the deployment of software components over a physical platform. However, these concerns can contradict each other (*e.g.*, accuracy and energy). In this context, we investigate automatic solutions to explore the set of possible architecture models and to establish valid trade-offs between all concerns in case of changes.

3.2.3.2. Scientific objectives

Automatic synthesis of optimal software architectures. Implementing a service over a distributed platform (*e.g.*, a pervasive system or a cloud platform) consists in deploying multiple software components over distributed computation nodes. We aim at designing search-based solutions to (i) assist the software architect in establishing a good initial architecture (that balances between different factors such as cost of the nodes, latency, fault tolerance) and to automatically update the architecture when the environment or the system itself change. The choice of search-based techniques is motivated by the very large number of possible software deployment architectures that can be investigated and that all provide different trade-offs between qualitative factors. Another essential aspect that is supported by multi-objective search is to explore different architectural solutions that are not necessarily comparable. This is important when the qualitative factors are orthogonal to each other, such as security and usability for example.

Flexible software architecture for testing and data management. As the number of platforms on which software runs increases and different software versions coexist, the demand for testing environments also increases. For example, to test a software patch or upgrade, the number of testing environments is the product of the number of running environments the software supports and the number of coexisting versions of the software. Based on our first experiment on the synthesis of cloud environment using architectural models, our objective is to define a set of domain specific languages to catch the requirement and to design cloud environments for testing and data management of future internet systems from data centers to things. These languages will be interpreted to support dynamic synthesis and reconfiguration of a testing environment.

Runtime support for heterogeneous environments. Execution environments must provide a way to account or reserve resources for applications. However, current execution environments such as the Java Virtual Machine do not clearly define a notion of application: each framework has its own definition. For example, in OSGi, an application is a component, in JEE, an application is most of the time associated to a class loader, in the Multi-Tasking Virtual machine, an application is a process. The challenge consists in defining an execution environment that provides direct control over resources (CPU, Memory, Network I/O) independently from the definition of an application. We propose to define abstract resource containers to account and reserve resources on a distributed network of heterogeneous devices.

3.2.4. Diverse implementations for resilience

Open software-intensive systems have to evolve over their lifetime in response to changes in their environment. Yet, most verification techniques assume a closed environment or the ability to predict all changes. Dynamic changes and evolutions thus represent a major challenge for these techniques that aim at assessing the correctness and robustness of the system. On the one hand, DIVERSE will adapt V&V techniques to handle diversity imposed by the requirements and the execution environment, on the other hand we leverage diversity to increase the robustness of software in face of unpredicted situations. More specifically, we address the following V&V challenges.

3.2.4.1. Challenges

One major challenge to build flexible and open yet dependable systems is that current software engineering techniques require architects to foresee all possible situations the system will have to face. However, openness and flexibility also mean unpredictability: unpredictable bugs, attacks, environmental evolutions, etc. Current fault-tolerance [113] and security [87] techniques provide software systems with the capacity of detecting accidental and deliberate faults. However, existing solutions assume that the set of bugs or vulnerabilities in a system does not evolve. This assumption does not hold for open systems, thus it is essential to revisit fault-tolerance and security solutions to account for diverse and unpredictable faults.

Diversity is known to be a major asset for the robustness of large, open, and complex systems (*e.g.*, economical or ecological systems). Following this observation, the software engineering literature provides a rich set of work that choose to implement diversity in software systems in order to improve robustness to attacks or to changes in quality of service. These works range from N-version programming to obfuscation of data structures or control flow, to randomization of instruction sets. An essential remaining challenge is to support the automatic synthesis and evolution of software diversity in open software-intensive systems. There is an opportunity to further enhance these techniques in order to cope with a wider diversity of faults, by multiplying the levels of diversity in the different software layers that are found in software-intensive systems (system, libraries, frameworks, application). This increased diversity must be based on artificial program transformations and code synthesis, which increase the chances of exploring novel solutions, better fitted at one point in time. The biological analogy also indicates that diversity should emerge as a side-effect of evolution, to prevent over-specialization towards one kind of diversity.

3.2.4.2. Scientific objectives

The main objective is to address one of the main limitations of N-version programming for fault-tolerant systems: the manual production and management of software diversity. Through automated injection of artificial diversity we aim at systematically increasing failure diversity and thus increasing the chances of early error detection at run-time. A fundamental assumption for this work is that software-intensive systems can be “good enough” [114], [126].

Proactive program diversification. We aim at establishing novel principles and techniques that favor the emergence of multiple forms of software diversity in software-intensive systems, in conjunction with the software adaptation mechanisms that leverage this diversity. The main expected outcome is a set of meta-design principles that maintain diversity in systems and the experimental demonstration of the effects of software diversity on the adaptive capacities of CASs. Higher levels of diversity in the system provide a pool of software solutions that can eventually be used to adapt to situations unforeseen at design time (bugs, crash, attacks, etc.). Principles of automated software diversification rely on the automated synthesis of variants in a software product line, as well as finer-grained program synthesis combining unsound transformations and genetic programming to explore the space of mutational robustness.

Multi-tier software diversification. We call multi-tier diversification the fact of diversifying several application software components simultaneously. The novelty of our proposal, with respect to the software diversity state of the art, is to diversify the application-level code (for example, diversify the business logics of the application), focusing on the technical layers found in web applications. The diversification of application software code is expected to provide a diversity of failures and vulnerabilities in web server deployment. Web server deployment usually adopts a form of the Reactor architecture pattern, for scalability purposes: multiple copies of the server software stack, called request handlers, are deployed behind a load balancer. This architecture is very favorable for diversification, since by using the multiplicity of request handlers running in a web server we can simultaneously deploy multiple combinations of diverse software components. Then, if one handler is hacked or crashes the others should still be able to process client requests.

4. Application Domains

4.1. From Embedded Systems to Service Oriented Architectures

From small embedded systems such as home automation products or automotive systems to medium sized systems such as medical equipment, office equipment, household appliances, smart phones; up to large Service Oriented Architectures (SOA), building a new application from scratch is no longer possible. Such applications reside in (group of) machines that are expected to run continuously for years without unrecoverable errors. Special care has then to be taken to design and validate embedded software, making the appropriate trade-off between various extra-functional properties such as reliability, timeliness, safety and security but also development and production cost, including resource usage of processor, memory, bandwidth, power, etc.

Leveraging ongoing advances in hardware, embedded software is playing an evermore crucial role in our society, bound to increase even more when embedded systems get interconnected to deliver ubiquitous SOA. For this reason, embedded software has been growing in size and complexity at an exponential rate for the past 20 years, pleading for a component based approach to embedded software development. There is a real need for flexible solutions allowing to deal at the same time with a wide range of needs (product lines modeling and methodologies for managing them), while preserving quality and reducing the time to market (such as derivation and validation tools).

We believe that building flexible, reliable and efficient embedded software will be achieved by reducing the gap between executable programs, their models, and the platform on which they execute, and by developing new composition mechanisms as well as transformation techniques with a sound formal basis for mapping between the different levels.

Reliability is an essential requirement in a context where a huge number of softwares (and sometimes several versions of the same program) may coexist in a large system. On one hand, software should be able to evolve very fast, as new features or services are frequently added to existing ones, but on the other hand, the occurrence of a fault in a system can be very costly, and time consuming. While we think that formal methods may help solving this kind of problems, we develop approaches where they are kept "behind the scene" in a global process taking into account constraints and objectives coming from user requirements.

Software testing is another aspect of reliable development. Testing activities mostly consist in trying to exhibit cases where a system implementation does not conform to its specifications. Whatever the efforts spent for development, this phase is of real importance to raise the confidence level in the fact that a system behaves properly in a complex environment. We also put a particular emphasis on on-line approaches, in which test and observation are dynamically computed during execution.

5. Highlights of the Year

5.1. Highlights of the Year

H2020 project accepted in the call ICT-10-2016 'Software Technologies', as coordinator.

The book "Engineering Modeling Languages" has been published by CRC Press. This book, co-authored by Benoit Combemale, Robert B. France, Jean-Marc Jézéquel, Bernhard Rumpe, Didier Vojtisek and Jim Steel, is the result of our respective expertise in model-driven engineering and software language engineering.

5.1.1. Awards

Silver Medal of the CNRS for Jean-Marc Jézéquel.

Second position for the ACM Student Research Competition: Thomas Degueule.

6. New Software and Platforms

6.1. FAMILIAR

KEYWORDS: Software line product - Configators - Customisation

SCIENTIFIC DESCRIPTION

FAMILIAR (for FeAture Model scrIpt Language for manIpulation and Automatic Reasoning) is a language for importing, exporting, composing, decomposing, editing, configuring, computing "diffs", refactoring, reverse engineering, testing, and reasoning about (multiple) feature models. All these operations can be combined to realize complex variability management tasks. A comprehensive environment is proposed as well as integration facilities with the Java ecosystem.

FUNCTIONAL DESCRIPTION

Familiar is an environment for large-scale product customisation. From a model of product features (options, parameters, etc.), Familiar can automatically generate several million variants. These variants can take many forms: software, a graphical interface, a video sequence or even a manufactured product (3D printing). Familiar is particularly well suited for developing web configurators (for ordering customised products online), for providing online comparison tools and also for engineering any family of embedded or software-based products.

- Participants: Mathieu Acher, Guillaume Becan, Olivier Barais
- Contact: Mathieu Acher
- URL: http://familiar-project.github.com

6.2. GEMOC Studio

KEYWORDS: Model-driven engineering - Meta model - MDE - DSL - Model-driven software engineering - Dedicated langage - Language workbench - Meta-modelisation - Modeling language - Meta-modeling

SCIENTIFIC DESCRIPTION

The language workbench put together the following tools seamlessly integrated to the Eclipse Modeling Framework (EMF):

- Melange, a tool-supported meta-language to modularly define executable modeling languages with execution functions and data, and to extend (EMF-based) existing modeling languages.
- MoCCML, a tool-supported meta-language dedicated to the specification of a Model of Concurrency and Communication (MoCC) and its mapping to a specific abstract syntax and associated execution functions of a modeling language.
- GEL, a tool-supported meta-language dedicated to the specification of the protocol between the execution functions and the MoCC to support the feedback of the data as well as the callback of other expected execution functions.
- BCOoL, a tool-supported meta-language dedicated to the specification of language coordination patterns to automatically coordinates the execution of, possibly heterogeneous, models.
- Sirius Animator, an extension to the model editor designer Sirius to create graphical animators for executable modeling languages.

FUNCTIONAL DESCRIPTION

The GEMOC Studio is an eclipse package that contains components supporting the GEMOC methodology for building and composing executable Domain-Specific Modeling Languages (DSMLs). It includes the two workbenches: The GEMOC Language Workbench: intended to be used by language designers (aka domain experts), it allows to build and compose new executable DSMLs. The GEMOC Modeling Workbench: intended to be used by domain designersto create, execute and coordinate models conforming to executable DSMLs. The different concerns of a DSML, as defined with the tools of the language workbench, are automatically deployed into the modeling workbench. They parametrize a generic execution framework that provide various generic services such as graphical animation, debugging tools, trace and event managers, timeline, etc.

- Participants: Benoit Combemale, Dorian Leroy, Thomas Degueule, Erwan Bousse, Fabien Coulon and Didier Vojtisek
- Contact: Benoit Combemale
- URL: http://gemoc.org

6.3. Kevoree

Kevoree Core

KEYWORDS: Cloud - Deployment - Embedded - Domotique - Heterogeneity - Software Components - Architecture - Software component - Dynamic adaptation - M2M - Dynamic deployment

SCIENTIFIC DESCRIPTION

Kevoree is an open-source models@runtime platform (http://www.kevoree.org) to properly support the dynamic adaptation of distributed systems. Models@runtime basically pushes the idea of reflection [132] one step further by considering the reflection layer as a real model that can be uncoupled from the running architecture (e.g. for reasoning, validation, and simulation purposes) and later automatically resynchronized with its running instance.

Kevoree has been influenced by previous work that we carried out in the DiVA project [132] and the Entimid project [135] . With Kevoree we push our vision of models@runtime [131] farther. In particular, Kevoree provides a proper support for distributed models@runtime. To this aim we introduced the Node concept to model the infrastructure topology and the Group concept to model semantics of inter node communication during synchronization of the reflection model among nodes. Kevoree includes a Channel concept to allow for multiple communication semantics between remoteComponents deployed on heterogeneous nodes. All Kevoree concepts (Component, Channel, Node, Group) obey the object type design pattern to separate deployment artifacts from running artifacts. Kevoree supports multiple kinds of very different execution node technology (e.g. Java, Android, MiniCloud, FreeBSD, Arduino, ...).

Kevoree is distributed under the terms of the LGPL open source license.

Main competitors:

the Fractal/Frascati eco-system (http://frascati.ow2.org).

SpringSource Dynamic Module (http://spring.io/)

GCM-Proactive (http://proactive.inria.fr/)

OSGi (http://www.osgi.org)

Chef

Vagran (http://vagrantup.com/)

Main innovative features:

distributed models@runtime platform (with a distributed reflection model and an extensible models@runtime dissemination set of strategies).

Support for heterogeneous node type (from Cyber Physical System with few resources until cloud computing infrastructure).

Fully automated provisioning model to correctly deploy software modules and their dependencies.

Communication and concurrency access between software modules expressed at the model level (not in the module implementation).

Impact:

Several tutorials and courses have been performed this year at EJCP for French PhD student, at ECNU summer school for 82 chineese PhD students. See also the web page http://www.kevoree.org .

In 2015, we mainly created a new implementation in C# and we created an implementation for system containers for driving resources using Kevoree. We also use Kevoree in the context of Mohammed's PhD to create testing infrastructure on-demand.

FUNCTIONAL DESCRIPTION

Kevoree is an open-source models@runtime platform to properly support the dynamic adaptation of distributed systems. Models@runtime basically pushes the idea of reflection one step further by considering the reflection layer as a real model that can be uncoupled from the running architecture (e.g. for reasoning, validation, and simulation purposes) and later automatically resynchronized with its running instance.

- Participants: Jean Emile Dartois, Olivier Barais, Aymeric Hervieu, Johann Bourcier, Noel Plouzeau, Benoit Baudry, Maxime Tricoire, Jacky Bourgeois, Inti Gonzalez Herrera, Ivan Paez Anaya, Francisco Javier Acosta Padilla, Mohamed Boussaa and Manuel Leduc
- Partner: Université de Rennes 1
- Contact: Olivier Barais
- URL: http://kevoree.org/

6.4. Melange

KEYWORDS: Model-driven engineering - Meta model - MDE - DSL - Model-driven software engineering - Dedicated langage - Language workbench - Meta-modelisation - Modeling language - Meta-modeling
SCIENTIFIC DESCRIPTION

Melange is a follow-up of the executable metamodeling language Kermeta, which provides a tool-supported dedicated meta-language to safely assemble language modules, customize them and produce new DSMLs. Melange provides specific constructs to assemble together various abstract syntax and operational semantics artifacts into a DSML. DSMLs can then be used as first class entities to be reused, extended, restricted or adapted into other DSMLs. Melange relies on a particular model-oriented type system that provides model polymorphism and language substitutability, i.e. the possibility to manipulate a model through different interfaces and to define generic transformations that can be invoked on models written using different DSLs. Newly produced DSMLs are correct by construction, ready for production (i.e., the result can be deployed and used as-is), and reusable in a new assembly.

Melange is tightly integrated with the Eclipse Modeling Framework ecosystem and relies on the meta-language Ecore for the definition of the abstract syntax of DSLs. Executable meta-modeling is supported by weaving operational semantics defined with Xtend. Designers can thus easily design an interpreter for their DSL in a non-intrusive way. Melange is bundled as a set of Eclipse plug-ins.
FUNCTIONAL DESCRIPTION

Melange is a language workbench which helps language engineers to mashup their various language concerns as language design choices, to manage their variability, and support their reuse. It provides a modular and reusable approach for customizing, assembling and integrating DSMLs specifications and implementations.

- Participants: Thomas Degueule, Benoit Combemale, Dorian Leroy, Erwan Bousse, Didier Vojtisek, Fabien Coulon, Jean-Marc Jezequel, Arnaud Blouin, Olivier Barais and David Mendez Acuna
- Contact: Benoit Combemale
- URL: http://melange-lang.org

6.5. Opencompare

KEYWORD: Software Product Line, Variability, MDE, Meta model, Configuration
FUNCTIONAL DESCRIPTION

Product comparison matrices (PCMs) are tabular data: supported and unsupported features are documented for both describing the product itself and for discriminating one product compared to another. PCMs abound – we are all using PCMs – and constitute a rich source of knowledge for easily comparing and choosing product. Yet the current practice is suboptimal both for humans and computers, mainly due to unclear semantics, heterogeneous forms of data, and lack of dedicated support.

OpenCompare.org is an ambitious project for the collaborative edition, the sharing, the standardisation, and the open exploitation of PCMs. The goal of OpenCompare.org is to provide an integrated set of tools (e.g., APIs, visualizations, configurators, editors) for democratizing their creation, import, maintenance, and exploitation.

- Participants: Mathieu Acher, Guillaume Becan and Sana Ben Nasr
- Contact: Mathieu Acher
- URL: http://opencompare.org

6.6. amiunique

KEYWORDS: Privacy - Browser fingerprinting
SCIENTIFIC DESCRIPTION

The amiunique web site has been deployed in the context of the DiverSE's research activities on browser fingerprinting and how software diversity can be leveraged in order to mitigate the impact of fingerprinting on the privacy of users. The construction of a dataset of genuine fingerprints is essential to understand in details how browser fingerprints can serve as unique identifiers and hence what should be modified in order to mitigate its impact privacy. This dataset also supports the large-scale investigation of the impact of web technology advances on fingerprinting. For example, we can analyze in details the impact of the HTML5 canvas element or the behavior of fingerprinting on mobile devices.

The whole source code of amiunique is open source and is distributed under the terms of the MIT license.

Similar sites: Panopticlick https://panopticlick.eff.org/ BrowserSpy http://browserspy.dk/ http://noc.to/ Main innovative features: canvas fingerprinting WebGL fingerprinting advanced JS features (platform, DNT, etc.)

Impact: The website has been showcased in several professional forums in 2014 and 2015 (Open World Forum 2014, FOSSA'14, FIC'15, ICT'15) and it has been visited by more than 100000 unique visitors in one year.
FUNCTIONAL DESCRIPTION

This web site aims at informing visitors about browser fingerprinting and possible tools to mitigate its effect, as well as at collecting data about the fingerprints that can be found on the web. It collects browser fingerprints with the explicit agreement of the users (they have to click on a button on the home page). Fingerprints are composed of 17 attributes, which include regular HTTP headers as well as the most recent state of the art techniques (canvas fingerprinting, WebGL information).

- Participants: Benoit Baudry and Pierre Laperdrix
- Partner: INSA Rennes
- Contact: Benoit Baudry
- URL: https://amiunique.org/

7. New Results

7.1. Results on Variability modeling and management

7.1.1. Feature Model Synthesis: Algorithms and Empirical Studies

We attack the problem of synthesising feature models by considering both configuration semantics and ontological semantics of a feature model. We define a generic synthesis procedure that computes the likely siblings or parent candidates for a given feature. We develop six heuristics for clustering and weighting the logical, syntactical and semantical relationships between feature names. We then perform an empirical evaluation on hundreds of feature models, coming from the SPLOT repository and Wikipedia. We provide evidence that a fully automated synthesis (i.e., without any user intervention) is likely to produce models far from the ground truths. As the role of the user is crucial, we empirically analyze the strengths and weaknesses of heuristics for computing ranking lists and different kinds of clusters. We show that a hybrid approach mixing logical and ontological techniques outperforms state-of-the-art solutions.

Numerous synthesis techniques and tools have been proposed, but only a few consider both configuration and ontological semantics of a feature model. We also boil down several feature model management operations to a synthesis problem. Our approach, the FAMILIAR environment, and empirical results support researchers and practitioners working on feature models. The synthesis problem is a core issue when reverse engineering, merging, slicing, or refactoring feature models. An article has been published in 2016 at Empirical Software Engineering journal, a major avenue for software engineering research [19].

7.1.2. Product Comparison Matrix

Product Comparison Matrices (PCMs) are widely used for documenting or comparing a set of products. PCMs are simple tabular data in which products are usually organized as rows, features as columns, while each cell define how a product implements the corresponding feature. We develop metamodeling and feature modeling techniques for formalizing PCMs. We perform numerous empirical experiments with users, tools, and data for validating our proposal. We also develop automated techniques to extract PCMs out of informal product descriptions, written in natural language. We establish a connection between PCMs and variability modeling formalism, which is of interest for the product line community. OpenCompare is a direct output of this research and is an important step towards the creation of a community around PCMs. We mined millions of Wikipedia tabular data together with end-users and developers to cross-validate our model-based approach [19]. We also mined data from BestBuy [17].

7.1.3. Machine Learning and Variability Testing

We propose the use of a machine learning approach to infer variability constraints from an oracle that is able to assess whether a given configuration is correct. We propose an automated procedure to randomly generate configurations, classify them according to the oracle, and synthesize cross-tree constraints. We validate our approach on a product-line video generator, using a simple computer vision algorithm as an oracle. We show that an interesting set of cross-tree constraint can be generated, with reasonable precision and recall. Our learning-based testing technique complements our initial effort in engineering an industrial video generator. The use of learning allows to significantly narrow the configuration space and discover complex constraints, hard to discover even for experts. We conduct a series of work in the computer vision domain to generate variants of videos, investigating the usefulness and effectiveness of variability techniques in novel areas. Our approach is novel and general: the same principles can be applied to other configurable systems [55].

7.1.4. Enumeration of All Feature Model Configurations

Feature models are widely used to encode the configurations of a software product line in terms of mandatory, optional and exclusive features as well as propositional constraints over the features. Numerous computationally expensive procedures have been developed to model check, test, configure, debug, or compute relevant information of feature models. We explore the possible improvement of relying on the enumeration of all configurations when performing automated analysis operations. We tackle the challenge of how to scale the existing enumeration techniques by relying on distributed computing. We show that the use of distributed computing techniques might offer practical solutions to previously unsolvable problems and opens new perspectives for the automated analysis of software product lines [40].

7.1.5. Software Unbundling

Unbundling is a phenomenon that consists of dividing an existing software artifact into smaller ones. It can happen for different reasons, one of them is the fact that applications tend to grow in functionalities and sometimes this can negatively influence the user experience. It can be seen as a way to produce different variants of an application. For example, mobile applications from well-known companies are being divided into simpler and more focused new ones. Despite its current importance, little is known or studied about unbundling or about how it relates to existing software engineering approaches, such as modularization. Consequently, recent cases point out that it has been performed unsystematically and arbitrarily. Our main goal is to present this novel and relevant concept and its underlying challenges in the light of software engineering, also exemplifying it with recent cases. We relate unbundling to standard software modularization, presenting the new motivations behind it, the resulting problems, and drawing perspectives for future support in the area [23].

7.1.6. Featured Model Types

By analogy with software product reuse, the ability to reuse (meta)models and model transformations is key to achieve better quality and productivity. To this end, various opportunistic reuse techniques have been developed, such as higher-order transformations, metamodel adaptation, and model types. However, in contrast to software product development that has moved to systematic reuse by adopting (model-driven) software product lines, we are not quite there yet for modelling languages, missing economies of scope and automation opportunities. Our vision is to transpose the product line paradigm at the metamodel level, where reusable assets are formed by metamodel and transformation fragments and "products" are reusable language building blocks (model types). We introduce featured model types to concisely model variability amongst metamodelling elements, enabling configuration, automated analysis, and derivation of tailored model types [53].

7.1.7. A Formal Modeling and Analysis Framework for SPL of Pre-emptive Real-time Systems

We present a formal analysis framework to analyze a family of platform products w.r.t. real-time properties. First, we propose an extension of the widely-used feature model, called Property Feature Model (PFM), that distinguishes features and properties explicitly Second, we present formal behavioral models of components of a real-time scheduling unit such that all real-time scheduling units implied by a PFM are automatically composed to be analyzed against the properties given by the PFM. We apply our approach to the verification of the schedulability of a family of scheduling units using the symbolic and statistical model checkers of Uppaal [44].

7.1.8. Exploration of Architectural Variants

In systems engineering, practitioners shall explore numerous architectural alternatives until choosing the most adequate variant. The decision-making process is most of the time a manual, time-consuming, and error-prone activity. The exploration and justification of architectural solutions is ad-hoc and mainly consists in a series of tries and errors on the modeling assets.

We report on an industrial case study in which we apply variability modeling techniques to automate the assessment and comparison of several candidate architectures (variants). We first describe how we can use a model-based approach such as the Common Variability Language (CVL) to specify the architectural variability. We show that the selection of an architectural variant is a multi-criteria decision problem in which there are numerous interactions (veto, favor, complementary) between criteria. We present a tooled process for exploring architectural variants integrating both CVL and the MYRIAD method for assessing and comparing variants based on an explicit preference model coming from the elicitation of stakeholders' concerns. This solution allows understanding differences among variants and their satisfactions with respect to criteria. Beyond variant selection automation improvement, this experiment results highlight that the approach improves rationality in the assessment and provides decision arguments when selecting the preferred variants. It is a joint work and collaboration with Thales [47].

7.1.9. A Complexity Tale: Web Configurators

Online configurators are basically everywhere. From physical goods (cars, clothes) to services (cloud solutions, insurances, etc.) such configurators have pervaded many areas of everyday life, in order to provide the customers products tailored to their needs. Being sometimes the only interfaces between product suppliers and consumers, much care has been devoted to the HCI aspects of configurators, aiming at offering an enjoyable buying experience. However, at the backend, the management of numerous and complex configuration options results from ad-hoc process rather than a systematic variability-aware engineering approach. We present our experience in analysing web configurators and formalising configuration options in terms of feature models or product configuration matrices. We also consider behavioural issues and perspectives on their architectural design [32].

7.2. Results on Software Language Engineering

7.2.1. Safe Model Polymorphism for Flexible Modeling

Domain-Specific Languages (DSLs) are increasingly used by domain experts to handle various concerns in systems and software development. To support this trend, the Model-Driven Engineering (MDE) community has developed advanced techniques for designing new DSLs. However, the widespread use of independently developed, and constantly evolving DSLs is hampered by the rigidity imposed to the language users by the DSLs and their tooling, e.g., for manipulating a model through various similar DSLs or successive versions of a given DSL. In [24] we propose a disciplined approach that leverages type groups' polymorphism to provide an advanced type system for manipulating models, in a polymorphic way, through different DSL interfaces. A DSL interface, aka. model type, specifies a set of features, or services, available on the model it types, and subtyping relations among these model types define the safe substitutions. This type system complements the Melange language workbench and is seamlessly integrated into the Eclipse Modeling Framework (EMF), hence providing structural interoperability and compatibility of models between EMF-based tools. We illustrate the validity and practicability of our approach by bridging safe interoperability between different semantic and syntactic variation points of a finite-state machine (FSM) language, as well as between successive versions of the Unified Modeling Language (UML).

7.2.2. Execution Framework for Model Debugging

The development and evolution of an advanced modeling environment for a Domain-Specific Modeling Language (DSML) is a tedious task, which becomes recurrent with the increasing number of DSMLs involved in the development and management of complex software-intensive systems. Recent efforts in language workbenches result in advanced frameworks that automatically provide syntactic tooling such as advanced editors. However, defining the execution semantics of languages and their tooling remains mostly hand crafted. Similarly to editors that share code completion or syntax highlighting, the development of advanced debuggers, animators, and others execution analysis tools shares common facilities, which should be reused among various DSMLs. In [37] we present the execution framework offered by the GEMOC studio, an Eclipse-based language and modeling workbench. The framework provides a generic interface to plug in different execution engines associated to their specific metalanguages used to define the discrete-event operational semantics of DSMLs. It also integrates generic runtime services that are shared among the approaches used to implement the execution semantics, such as graphical animation or omniscient debugging.

7.2.3. Variability Management in Language Families

The use of domain-specific languages (DSLs) has become a successful technique in the development of complex systems. Nevertheless, the construction of this type of languages is time-consuming and requires highly-specialized knowledge and skills. An emerging practice to facilitate this task is to enable reuse through the definition of language modules which can be later put together to build up new DSLs. In [29], we report on an effort for organizing the literature on language product line engineering. More precisely, we propose a definition for the life-cycle of language product lines, and we use it to analyze the capabilities of current approaches. In addition, we provide a mapping between each approach and the technological space it supports.

Still, the identification and definition of language modules are complex and error-prone activities, thus hindering the reuse exploitation when developing DSLs. In [50], [51], we propose a computer-aided approach to i) identify potential reuse in a set of legacy DSLs; and ii) capitalize such potential reuse by extracting a set of reusable language modules with well defined interfaces that facilitate their assembly. We validate our approach by using realistic DSLs coming out from industrial case studies and obtained from public GitHub repositories. We also developed a publicly available tool, namely Puzzle, that uses static analysis to facilitate the detection of specification clones in DSLs implemented under the executable metamodeling paradigm. Puzzle also enables the extraction specification clones as reusable language modules that can be later used to build up new DSLs.

7.2.4. A Tool-Supported Approach for Concurrent Execution of Heterogeneous Models

In the software and systems modeling community, research on domain-specific modeling languages (DSMLs) is focused on providing technologies for developing languages and tools that allow domain experts to develop system solutions efficiently. Unfortunately, the current lack of support for explicitly relating concepts expressed in different DSMLs makes it very difficult for software and system engineers to reason about information spread across models describing different system aspects. As a particular challenge, we investigate in [38] relationships between, possibly heterogeneous, behavioral models to support their concurrent execution. This is achieved by following a modular executable metamodeling approach for behavioral semantics understanding, reuse, variability and composability. This approach supports an explicit model of concurrency (MoCC) and domain-specific actions (DSA) with a well-defined protocol between them (incl., mapping, feedback and callback) reified through explicit domain-specific events (DSE). The protocol is then used to infer a relevant behavioral language interface for specifying coordination patterns to be applied on conforming executable models. All the tooling of the approach is gathered in the GEMOC studio, and outlined in the next section. Currently, the approach is experienced on a systems engineering language provided by Thales, named Capella.

7.2.5. Various Dimensions of Reuse

Reuse, enabled by modularity and interfaces, is one of the most important concepts in software engineering. This is evidenced by an increasingly large number of reusable artifacts, ranging from small units such as classes to larger, more sophisticated units such as components, services, frameworks, software product lines, and concerns. We give evidence in [43] that a canonical set of reuse interfaces has emerged over time: the variation, customization, and usage interfaces (VCU). A reusable artifact that provides all three interfaces reaches the highest potential of reuse, as it explicitly exposes how the artifact can be manipulated during the reuse process along these three dimensions. We demonstrate the wide applicability of the VCU interfaces along two axes: across abstraction layers of a system specification and across existing reuse techniques. The former is shown with the help of a comprehensive case study including reusable requirements, software, and hardware models for the authorization domain. The latter is shown with a discussion on how the VCU interfaces relate to existing reuse techniques.

7.2.6. Modeling for Sustainability

The complex problems that computational science addresses are more and more benefiting from the progress of computing facilities (e.g., simulators, librairies, accessible languages). Nevertheless, the actual solutions call for several improvements. Among those, we address the needs for leveraging on knowledge and expertise by focusing on Domain-Specific Modeling Languages application. In this work we explored, through concrete experiments, how the last DSML research help getting closer the problem and implementation spaces.

Various disciplines use models for different purposes. While engineering models, including software engineering models, are often developed to guide the construction of a non- existent system, scientific models, in contrast, are created to better understand a natural phenomenon (i.e., an already existing system). An engineering model may incorporate scientific models to build a system. Both engineering and scientific models have been used to support sustainability, but largely in a loosely-coupled fashion, independently developed and maintained from each other. Due to the inherent complex nature of sustainability that must balance trade-offs between social, environmental, and economic concerns, modeling challenges abound for both the scientific and engineering disciplines. In [39] we propose a vision that synergistically combines engineering and scientific models to enable broader engagement of society for addressing sustainability concerns, informed decision-making based on more accessible scientific models and data, and automated feed-back to the engineering models to support dynamic adaptation of sustainability systems. To support this vision, we identify a number of challenges to be addressed with particular emphasis on the socio-technical benefits of modeling.

As first experiments, we presented at the EclipseCon France, Europe and North America 2016, an approach to develop smart cyber physical systems in charge of managing the production, distribution and consumption of energies (e.g., water, electricity). The main objective is to enable a broader engagement of society,

while supporting a more informed decision-making, possibly automatically, on the development and run-time adaptation of sustainability systems (e.g., smart grid, home automation, smart cities). We illustrate this approach through a system that allows farmers to simulate and optimize their water consumption by combining the model of a farming system together with agronomical models (e.g., vegetable and animal lifecycle) and open data (e.g., climate series). To do so, we use Model Driven Engineering (MDE) and Domain Specific Languages (DSL) to develop such systems driven by scientific models that define the context (e.g., environment, social and economy), and model experiencing environments to engage general public and policy makers.

7.2.7. Formal Specification of a Packet Filtering Language Using the K Framework

Many project-specific languages, including in particular filtering languages, are defined using non-formal specifications written in natural languages. This leads to ambiguities and errors in the specification of those languages. In [46] we report on an industrial experiment on using a tool-supported language specification framework (K) for the formal specification of the syntax and semantics of a filtering language having a complexity similar to those of real-life projects. This experimentation aims at estimating, in a specific industrial setting, the difficulty and benefits of formally specifying a packet filtering language using a tool-supported formal approach.

7.2.8. Correct-by-construction model driven engineering composition operators

Model composition is a crucial activity in Model Driven Engineering both to reuse validated and verified model elements and to handle separately the various aspects in a complex system and then weave them while preserving their properties. Many research activities target this compositional validation and verification (V & V) strategy: allow the independent assessment of components and minimize the residual V & V activities at assembly time. However, there is a continuous and increasing need for the definition of new composition operators that allow the reconciliation of existing models to build new systems according to various requirements. These ones are usually built from scratch and must be systematically verified to assess that they preserve the properties of the assembled elements. This verification is usually tedious but is mandatory to avoid verifying the composite system for each use of the operators. Our work addresses these issues, we first target the use of proof assistants for specifying and verifying compositional verification frameworks relying on formal verification techniques instead of testing and proofreading. Then, using a divide and conquer approach, we focus on the development of elementary composition operators that are easy to verify and can be used to further define complex composition operators. In our approach [27], proofs for the complex operators are then obtained by assembling the proofs of the basic operators. To illustrate our proposal, we use the Coq proof assistant to formalize the language-independent elementary composition operators Union and Substitution and the proof that the conformance of models with respect to metamodels is preserved during composition. We show that more sophisticated composition operators that share parts of the implementation and have several properties in common (especially: aspect oriented modeling composition approach, invasive software composition, and package merge) can then be built from the basic ones, and that the proof of conformance preservation can also be built from the proofs of basic operators.

7.2.9. Engineering Modeling Languages

The DiverSE project-team is deeply involved in transferring research knowledge into education. In particular, one book in English have been published in 2016 as a textbook [59]. The book cover the broad scope of MDE, and are based on the experience of the project-team members.

7.3. Results on Heterogeneous and dynamic software architectures

We have selected three main contributions : two are in the field of runtime management, while the third one is in the field of non-functionnal software testing.

7.3.1. Precise and efficient resource management using models@runtime

Contribution. We have developed an efficient monitoring framework to quickly spot an abnormal resource consumption within a complex application. In these papers [25], we have proposed an optimistic adaptive monitoring system to determine the faulty components of an application. Suspected components are finely analyzed by the monitoring system, but only when required. Unsuspected components are left untouched and execute normally.

Originality. Current solutions that perform permanent and extensive monitoring to detect anomalies induce high overhead on the system, and can, by themselves, make the system unstable. Our system performs localized just-in-time monitoring that decreases the accumulated overhead of the monitoring system. Through our evaluation, we show that our technique correctly detects faulty components, while reducing overhead by 92.98 on average%.

Impact. Beyond the scientific originality of this work, the main impacts of this novel approach approach to monitor software component performance has been to (i) reinforce DIVERSE's visibility in the academic and industrial communities on software components and (ii) to create several research tracks that are currently explored in different projects of the team (HEADS and B-com PhD thesis). This work has been integrated within the Kevoree platform.

7.3.2. Dynamic web application using models@runtime

Contribution. We have developed a component-based platform supporting the development of dynamically adaptable single Web page applications. An important part of this contribution lies in the possibility to dynamically move code from the server to the client side allowing a great flexibility in the performance management. This contribution [56] is based on a models@runtime approach and has been implemented in our open source KevoreeJS platform.

Originality. Current solutions to create single Web page application are limited to a static code repartition between clients and server, thus limiting the flexibility at runtime.

Impact. Beyond the scientific originality of this work, the main impacts of this novel approach to monitor software component performance has been to (i) reinforce DIVERSE's visibility in the open-source community, (ii) to start several research tracks that are currently explored in different projects of the team (HEADS, STAMP, GRevis). This platforms is modular, one of the component has a monthly download count greater than 100k [3]).

7.3.3. Testing non-functional behavior of compiler and code generator

Contribution. We have developed NOTICE [36], [35], a component-based framework for non-functional testing of compilers through the monitoring of generated code in a controlled sand-boxing environment. In this work, we have proposed an automatic way of testing non-functional properties of compilers, while optimizing the generated application with respect to a set of specific non-functional properties (CPU, memory usage, energy consumption, *etc.*).

Originality. Compiler users generally apply different optimizations to generate efficient code with respect to specific non-functional properties such as energy consumption, execution time, etc. However, due to the huge number of optimizations provided by modern compilers, finding the best optimization sequence for a specific objective and a given program is more and more challenging.

Impact. Beyond the scientific originality of this work, the main impact of this novel approach is to enable the auto-tuning of compilers according to user requirements and to construct optimizations that yield to performance results that are better than standard optimization levels.

7.3.4. Automatic Microbenchmark Generation to Prevent Dead Code Elimination and Constant Folding

[3] https://www.npmjs.com/package/npmi

Contribution. Microbenchmarking consists of evaluating, in isolation, the performance of small code segments that play a critical role in large applications. The accuracy of a microbenchmark depends on two critical tasks: wrap the code segment into a payload that faithfully recreates the execution conditions that occur in the large application; build a scaffold that runs the payload a large number of times to get a statistical estimate of the execution time. While recent frameworks such as the Java Microbenchmark Harness (JMH) take care of the scaffold challenge, developers have very limited support to build a correct payload. This year, we focus on the automatic generation of pay-loads, starting from a code segment selected in a large application [54]. In particular, we aim at preventing two of the most common mistakes made in microbenchmarks: dead code elimination and constant folding. Since a microbench-mark is such a small program, if not designed carefully, it will be *over-optimized* by the JIT and result in distorted time measures. Our technique hence automatically extracts the segment into a compilable payload and generates additional code to prevent the risks of *over-optimization*. The whole approach is embedded in a tool called AutoJMH, which generates payloads for JMH scaffolds. We validate the capabilities AutoJMH, showing that the tool is able to process a large percentage of segments in real programs. We also show that AutoJMH can match the quality of payloads handwritten by performance experts and outperform those written by professional Java developers without experience un microbenchmarking.

7.3.5. Collaborations

This year, we had a close and fruitful collaboration with the industrial partners that are involved in the HEADS and Occiware projects, in particular an active interaction with the Tellu company in Norway in the Heads context [49]. Tellu relies on Kevoree and KevoreeJS to build their health management systems. They will be also a active member the new Stamp project led by DIVERSE. We can cite also an active collaboration with Orange Labs through Kevin Corre's joint PhD thesis. Another joint industrial (CIFRE) PhD started in September 2016, and we are also partner in a new starting FUI project. Finally, DIVERSE collaborates with the B-COM IRT (https://b-com.com/en), as one permanent member has a researcher position of one day per week at B-COM and a new joint PhD started in September [52].

At the academic level we collaborate actively with the Spiral team at Inria Lille (several joint projects), the Tacoma team (with two co-advised PhD students), the Myriad team (1 co-advised PhD student) and we have started two collaborations with the ASAP team.

7.4. Results on Diverse Implementations for Resilience

Diversity is acknowledged as a crucial element for resilience, sustainability and increased wealth in many domains such as sociology, economy and ecology. Yet, despite the large body of theoretical and experimental science that emphasizes the need to conserve high levels of diversity in complex systems, the limited amount of diversity in software-intensive systems is a major issue. This is particularly critical as these systems integrate multiple concerns, are connected to the physical world through multiple sensors, run eternally and are open to other services and to users. Here we present our latest observational and technical results about (i) new approaches to increase diversity in software systems, and (ii) software testing to assess the validity of software.

7.4.1. Software diversification

A main achievement in our investigations of software diversity, is a large scale analysis of browser fingerprints [45]. Browser fingerprinting consists in collecting information about a user's browser and its execution environment. A distinctive feature of these fingerprints is that they are unique and can be used to track users. We show that innovations in HTML5 provide access to highly discriminating attributes, notably with the use of the Canvas API which relies on multiple layers of the user's system. In addition, we show that browser fingerprinting is as effective on mobile devices as it is on desktops and laptops, albeit for radically different reasons due to their more constrained hardware and software environments. We also evaluate how browser fingerprinting could stop being a threat to user privacy if some technological evolutions continue (e.g., disappearance of plugins) or are embraced by browser vendors (e.g., standard HTTP headers).

As for automatic diversification of programs, we have had a strong focus on runtime transformations. Online Genetic Improvement embeds the ability to evolve and adapt inside a target software system enabling it to improve at runtime without any external dependencies or human intervention. We recently developed a general purpose tool enabling Online Genetic Improvement in software systems running on the java virtual machine. This tool, dubbed ECSELR, is embedded inside extant software systems at runtime, enabling such systems to autonomously generate diverse variants [31]. We have also worked on diversification against just-in-Time (JIT) Spraying: a technique that embeds return-oriented programming (ROP) gadgets in arithmetic or logical instructions as immediate offsets. We introduce libmask, a JIT compiler extension that transforms constants into global variables and marks the memory area for these global variables as read only. Hence, any constant is referred to by a memory address making exploitation of arithmetic and logical instructions more difficult. Then, these memory addresses are randomized to further harden the security [42].

7.4.2. Software testing

Our work in the area of software testing focuses on tailoring the testing tools (analysis, generation, oracle, etc.) to specific domains and purposes. This allows us to consider domain specific knowledge (e.g., architectural patterns for GUI implementation) in order to increase the relevance and the efficiency of testing. The main results of this year are about test case refactoring and testing code generators.

Software developers design test suites to verify that software meets its expected behaviors. Yet, many dynamic analysis techniques are performed on the exploitation of execution traces from test cases. In practice, one test case may imply various behaviors. However, the execution of a test case only yields one trace, which can hide the others. We have developed a new technique of test code refactoring, which splits a test case into small test fragments that cover a simpler part of the control flow to provide better support for dynamic analysis. This technique can effectively improve the execution traces of the test suite: exception contracts are better verified via applying this refactoring to original test suites [30].

Finding the smallest set of valid test configurations that ensure sufficient coverage of the system's feature interactions is essential, especially when the execution of test configurations is costly or time-consuming. However, this problem is NP-hard in general and approximation algorithms have often been used to address it in practice. We explore an approach based on constraint programming to increase the effectiveness of configuration testing while keeping the number of configurations as low as possible. For 79% of 224 feature models, our technique generated up to 60% fewer test configurations than the competitor tools [26].

The intensive use of generative programming techniques provides an elegant engineering solution to deal with the heterogeneity of platforms and technological stacks. Yet, producing correct and efficient code generator is complex and error-prone. We describe a practical approach based on a runtime monitoring infrastructure to automatically check the potential inefficient code generators. We evaluate our approach by analyzing the performance of Haxe, a popular high-level programming language that involves a set of cross-platform code generators. The results show that our approach is able to detect some performance inconsistencies that reveal real issues in Haxe code generators [36], [35]

Graphical User Interfaces (GUIs) intensively rely on event-driven programming: widgets send GUI events, which capture users' interactions, to dedicated objects called *controllers*. Controllers implement several *GUI listeners* that handle these events to produce GUI commands. We study to what extent the number of GUI commands that a GUI listener can produce has an impact on the code quality. We then identify a new type of design smell, called *Blob listener* that characterizes GUI listeners that can produce more than two GUI commands. We propose a systematic static code analysis procedure that searches for *Blob Listener* instances that we implement in *InspectorGuidget* [48].

8. Partnerships and Cooperations

8.1. National Initiatives

8.1.1. ANR

8.1.1.1. GEMOC

- Coordinator: Inria (DIVERSE)
- Other partners: ENSTA Bretagne, Inria, IRIT, I3S, Obeo, Thales
- Dates: 2012-2016
- Abstract: GEMOC focuses on a generic framework for heterogeneous software model execution and dynamic analysis. This work has the ambition to propose an innovative environment for the design of complex software-intensive systems by providing: a formal framework that integrates state-of-the-art in model-driven engineering (MDE) to build domain-specific modeling languages (DSMLs), and models of computation (MoC) to reason over the composition of heterogeneous concerns; an open-source design and modeling environment associated to a well-defined method for the definition of DSMLs, MoCs and rigorous composition of all concerns for execution and analysis purposes.

  This requires addressing two major scientific issues: the design and verification of a formal framework to combine several different DSMLs relying on distinct MoCs; the design and validation of a methodology for DSMLs and MoC development. GEMOC aims at participating in the development of next generation MDE environments through a rigorous, tool-supported process for the definition of executable DSMLs and the simulation of heterogeneous models.

8.1.1.2. SOPRANO

- Coordinator: CEA
- CEA, University of Paris-Sud, Inria Rennes, OcamlPro, Adacore
- Dates: 2014-2017
- Abstract: Today most major verification approaches rely on automatic external solvers. However these solvers do not fill the current and future needs for verification: lack of satisfying model generation, lack of reasoning on difficult theories (e.g. floating-point arithmetic), lack of extensibility for specific or new needs. The SOPRANO project aims at solving these problems and prepare the next generation of verification-oriented solvers by gathering experts from academia and industry. We will design a new framework for the cooperation of solvers, focused on model generation and borrowing principles from SMT (current standard) and CP (well-known in optimisation). These ideas will be implemented in an open-source platform, with regular evaluations from the industrial partners.

8.1.1.3. Gdiv MRSE

- Coordinator: B. Baudry
- Inria Rennes
- Dates: 2014-2016
- Abstract: The objective of GDiv is to setup a strong network of European partners around the core team composed of Inria and SINTEF. This network will gather another academic partner and between 3 and 5 industry partners in the areas of software development and deployment. The project proposal setup by the GDiv network will address the risks of large scale software reuse through integrated, multi-level software diversification techniques.

8.1.2. BGLE / LEOC

8.1.2.1. CONNEXION

- Coordinator: EDF
- Other partners: Atos WorldGrid, Rolls-Royce Civil Nuclear, Corys TESS, Esterel Technologies, All4Tec, Predict, CEA, Inria, CNRS / CRAN, ENS Cachan, LIG, Telecom ParisTech
- Dates: 2012-2016
- Abstract: The cluster CONNEXION (*digital command CONntrol for Nuclear EXport and renovatION*) aims to propose and validate an innovative architecture platforms suitable control systems for nuclear power plants in France and abroad. In this project the Triskell team investigates methods and tools to (i) automatically analyze and compare regulatory requirements evolutions and geographical differences; (ii) automatically generate test cases for critical interactive systems.

8.1.2.2. CLARITY

- Coordinator: Obéo
- Other partners: AIRBUS, Airbus Defence and Space, All4tec, ALTRAN Technologies, AREVA, Artal, C.E.S.A.M.E.S., Eclipse Foundation Europe, Inria Sophia Antipolis Méditerranée, PRFC, Scilab Enterprises, Thales Global Services, Thales Alenia Space, Thales Research & Technology, Thales Systèmes Aéroportés, Université de Rennes 1.
- Dates: 2014-2017
- Abstract: The CLARITY project aims to establish an international dimension ecosystem around Melody/Capella modeling workbench for systems engineering (MBSE) and engineering architectures (system, software, hardware).

8.1.2.3. Occiware

- Coordinator: Open Wide
- Open Wide, ActiveEon SA, CSRT - Cloud Systèmes Réseaux et Télécoms, Institut Mines-Télécom/Télécom SudParis, Inria, Linagora, Obeo, OW2 Consortium, Pôle Numérique, Université Joseph Fourier,
- Dates: 2014-2017
- Abstract: The Occiware project aims to establish a formal and equipped framework for the management of all cloud resource based on the OCCI standard.

8.1.3. DGA

8.1.3.1. MOTIV

- Coordinator: InPixal
- Other partners: Bertin, DGA, Inria
- Dates: 2012-2014
- Abstract: This project investigates innovative software test generation and management solutions to handle the very high degrees of variability in video processing algorithmic chains. The objective is to provide systematic criteria to qualify the testing activity when developing video processing software and to tailor these criteria to the variability dimensions that emerge in the context of visible images.

8.1.3.2. FPML (CYBERDEFENSE)

- Coordinator: DGA
- Partners: DGA MI, Inria
- Dates: 2014-2016
- Abstract: in the context of this project, DGA-MI and the Inria team DiverSE explore the existing approaches to ease the development of formal specifications of domain-Specific Languages (DSLs) dedicated to paquet filtering, while guaranteeing expressiveness, precision and safety. In the long term, this work is part of the trend to provide to DGA-MI and its partners a tooling to design and develop formal DSLs which ease the use while ensuring a high level of reasoning.

8.1.4. Cominlabs

8.1.4.1. PROFILE

- Coordinator: Université de Rennes 1
- Partners: Inria, Université de Rennes 2
- Dates: 2016-2019

- Abstract: The PROFILE project brings together experts from law, computer science and sociology to address the challenges raised by online profiling, following a multidisciplinary approach. More precisely, the project will pursue two complementary and mutually informed lines of research: (i) Investigate, design, and introduce a new right of opposition into the legal framework of data protection to better regulate profiling and to modify the behavior of commercial companies towards being more respectful of the privacy of their users; (ii)S Provide users with the technical means they need to detect stealthy profiling techniques as well as to control the extent of the digital traces they routinely produce. As a case study, we focus on browser fingerprinting, a new profiling technique for targeted advertisement. The project will develop a generic framework to reason on the data collected by profiling algorithms, to uncover their inner working, and make them more accountable to users. PROFILE will also propose an innovative protection to mitigate browser fingerprinting, based on the collaborative reconfiguration of browsers.

8.2. European Initiatives

8.2.1. FP7 & H2020 Projects

8.2.1.1. FP7 FET STREP DIVERSIFY

- Coordinator: Inria (DIVERSE)
- Partners: SINTEF, Université de Rennes 1, Trinity College Dublin, Inria (DiverSE, SPIRALS)
- Dates: 2013-2016
- Abstract: DIVERSIFY explores diversity as the foundation for a novel software design principle and increased adaptive capacities in CASs. Higher levels of diversity in the system provide a pool of software solutions that can eventually be used to adapt to unforeseen situations at design time. The scientific development of DIVERSIFY is based on a strong analogy with ecological systems, biodiversity, and evolutionary ecology. DIVERSIFY brings together researchers from the domains of software-intensive distributed systems and ecology in order to translate ecological concepts and processes into software design principles.

8.2.1.2. FP7 STREP HEADS

- Coordinator: SINTEF
- Other partners: Inria, Software AG, ATC, Tellu, eZmonitoring
- Dates: 2013-2016
- Abstract: The idea of the HEADS project is to leverage model-driven software engineering and generative programming techniques to provide a new integrated software engineering approach which allow advanced exploitation the full range of diversity and specificity of the future computing continuum. The goal is to empower the software and services industry to better take advantage of the opportunities of the future computing continuum and to effectively provide new innovative services that are seamlessly integrated to the physical world making them more pervasive, more robust, more reactive and closer (physically, socially, emotionally, etc.) to their users. We denote such services HD-services. HD-services (Heterogeneous and Distributed services) characterize the class of services or applications within the Future Internet whose logic and value emerges from a set of communicating software components distributed on a heterogeneous computing continuum from clouds to mobile devices, sensors and/or smart-objects.

8.2.1.3. H2020 ICT-10-2016 STAMP

- Coordinator: Inria (DIVERSE)
- Other partners: ATOS, ActiveEon, OW2, TellU, Engineering, XWiki, TU Delft, SINTEF
- Dates: 2016-2019

- Abstract: Leveraging advanced research in automatic test generation, STAMP aims at pushing automation in DevOps one step further through innovative methods of test amplification. It will reuse existing assets (test cases, API descriptions, dependency models), in order to generate more test cases and test configurations each time the application is updated. Acting at all steps of development cycle, STAMP techniques aim at reducing the number and cost of regression bugs at unit level, configuration level and production stage.

  STAMP will raise confidence and foster adoption of DevOps by the European IT industry. The project gathers 3 academic partners with strong software testing expertise, 5 software companies (in: e-Health, Content Management, Smart Cities and Public Administration), and an open source consortium. This industry-near research addresses concrete, business-oriented objectives. All solutions are open source and developed as microservices to facilitate exploitation, with a target at TRL 6.

8.2.2. Collaborations in European Programs, Except FP7 & H2020

8.2.2.1. ICT COST Action MPM4CPS (IC1404)

- Chair of the Action: Prof Hans Vangheluwe (BE)
- Dates: 2014-2018
- Abstract: Truly complex, designed systems, known as Cyber Physical Systems (CPS), are emerging that integrate physical, software, and network aspects. To date, no unifying theory nor systematic design methods, techniques and tools exist for such systems. Individual (mechanical, electrical, network or software) engineering disciplines only offer partial solutions. Multi-paradigm Modelling (MPM) proposes to model every part and aspect of a system explicitly, at the most appropriate level(s) of abstraction, using the most appropriate modelling formalism(s). Modelling languages' engineering, including model transformation, and the study of their semantics, are used to realize MPM. MPM is seen as an effective answer to the challenges of designing CPS. This COST Action promotes the sharing of foundations, techniques and tools, and provide educational resources, to both academia and industry. This is achieved by bringing together and disseminating knowledge and experiments on CPS problems and MPM solutions. Benoit Combemale is a member of the management committee.

8.2.3. Collaborations with Major European Organizations

SINTEF, ICT (Norway): Model-driven systems development for the construction of distributed, heterogeneous applications. We collaborate since 2008 and are currently in two FP7 projects together.

Université du Luxembourg, (Luxembourg): Models runtime for dynamic adaptation and multi-objective elasticity in cloud management; model-driven development.

Open University (UK): models runtime for the Internet of Things.

8.3. International Initiatives

- Université de Montréal (Canada)
- McGill University (Canada)
- University of Alabama (USA)
- TU Wien (Austria)
- Michigan State University (USA)
- Aachen University (Germany)

8.3.1. Participation in Other International Programs

The GEMOC studio has been sustained through the creation of a Research Consortium at the Eclipse Foundation.

8.3.2. International initiative GEMOC

The GEMOC initiative (cf. http://www.gemoc.org) is an open and international initiative launched in 2013 that coordinate research partners worldwide to develop breakthrough software language engineering (SLE) approaches that support global software engineering through the use of multiple domain-specific languages. GEMOC members aim to provide effective SLE solutions to problems associated with the design and implementation of collaborative, interoperable and composable modeling languages.

The GEMOC initiative aims to provide a framework that facilitates collaborative work on the challenges of using of multiple domain-specific languages in software development projects. The framework consists of mechanisms for coordinating the work of members, and for disseminating research results and other related information on GEMOC activities. The framework also provides the required infrastructure for sharing artifacts produced by members, including publications, case studies, and tools.

The governance of the GEMOC initiative is ensured by the Advisory Board. The role of the Advisory Board is to coordinate the GEMOC work and to ensure proper dissemination of work products and information about GEMOC events (e.g., meetings, workshops).

Benoit Combemale is the co-founder and currently acts as principal coordinator of the GEMOC initiative. Benoit Combemale and Jean-Marc Jézéquel are part of the Advisory Board, and 9 DIVERSE members are part of the GEMOC initiative.

8.4. International Research Visitors

8.4.1. Visits of International Scientists

Yves Le Traon, Professor at the University of Luxembourg, visited the team in June and July 2016.

Tanja Mayerhofer, Junior Researcher at the TU Wien, visited the team in September 2016.

Bernhard Rumpe, Professor at Aachen University, visited the team in May 2016.

8.4.1.1. Internships

Vikas Mishra, Master intership at the Birla Institute of Technology & Science, visited the team from June to August 2016.

Alexandre Nuttinck, Axel Halin, Master interships at the University of Namur, visited the team from September 2016 to January 2017.

8.4.2. Visits to International Teams

Manuel Leduc visited CWI for 3 weeks in December 2016

Benoit Baudry visited Professor Stephanie Forrest at the University of New Mexico for one month in April 2016.

Benoit Combemale visited Professor Jorg Kienzle at McGill University for 3 weeks in June 2016; and visited Professor Bernhard Rumpe at Aachen University in April 2016.

8.4.2.1. Research Stays Abroad

Marcelino Rodriguez-Cancio visited Vanderbildt University from November 2016 to May 2017.

9. Dissemination

9.1. Promoting Scientific Activities

9.1.1. Scientific Events Organisation

9.1.1.1. General Chair, Scientific Chair

Benoit Baudry and Benoit Combemale have been general co-chairs for MODELS 2016, the major international conference in the area of model-driven software and systems engineering.

9.1.1.2. Member of the Organizing Committees

Mathieu Acher:

- Social media chair, MODELS 2016

Arnaud Blouin:

- Student volunteers co-chair, MODELS 2016
- Social Media Chair, EICS 2016

9.1.2. Scientific Events Selection

9.1.2.1. Chair of Conference Program Committees

Noël Plouzeau was the program committee chair of the Component Based Software Engineering (CBSE) in 2016. Jean-Marc Jézéquel was co-chair of Digital Intelligence 2016, April 2016, Quebec City, Canada.

9.1.2.2. Member of the Conference Program Committees

Mathieu Acher:

- PC member SPLC 2016
- PC member SAC 2016
- PC member VaMoS 2016
- PC member SEAA 2016
- PC member REVE 2016

Olivier Barais:

- PC member SEAA 2016
- PC member 10th IEEE ICOSST-2016
- PC member compas2016
- PC member ASE 2016
- PC member TTC 2016
- PC member SAC 2016

Benoit Baudry:

- PC member ICSE 2016
- PC member ISSTA 2016
- PC member ICST 2016
- PC member ASE 2016

Arnaud Blouin:

- PC member of the foundation track for MODELS 2016
- PC member for IHM 2016
- PC member of the NIER/demo track for VISSOFT 2016
- PC member for the UCDAS workshop at COMPSAC 2016

Benoit Combemale:

- PC member for ECMFA'16
- PC member for ICMT'16
- PC member for the GEMOC'16 workshop at MODELS'16
- PC member for the EXE'16 workshop at MODELS'16
- PC member for the MiSE'16 workshop at ICSE'16
- PC member for the LaMOD'16 workshop at Modularity'16
- PC member for the MoMo'16 workshop at Modularity'16

Johann Bourcier:

- PC member for the SEsCPS'16 workshop at ICSE'16

Jean-Marc Jézéquel:

- Program Board of MODELS 2016
- PC member for SPLC 2016 Industry Track
- PC member for SPLC 2016 Vision Track
- PC member for SEAMS 2016
- PC member for GEMOC 2016

9.1.2.3. Reviewer

Arnaud Blouin was a reviewer for ICST 2016, ASE 2016, ECMFA 2016. Noël Plouzeau was a reviewer for ICSE 2016. Olivier Barais was a reviewer for ICSE 2016, Sosym 2016. Johann Bourcier was a reviewer for ICSE 2016, ASE 2016 and Middleware 2016.

9.1.3. Journal

9.1.3.1. Member of the Editorial Boards

Benoit Baudry:

- SOSYM
- STVR

Benoit Combemale:

- Springer Journal on Software and System Modeling (SoSYM),
- Elsevier Journal on Computer Languages, Systems and Structures (COMLAN),
- Elsevier Journal on Science of Computer Programming (SCP), Advisory Board member of the Software Section.

9.1.3.2. Reviewer - Reviewing Activities

Mathieu Acher was a reviewer for TSE, JSS, IST.

Johann Bourcier was a reviewer for JSS, IEEE Communications Magazine,

Arnaud Blouin: SoSyM

Jean-Marc Jézéquel:

- Associate Editor in Chief of SOSYM
- IEEE Computer
- JSS
- JOT

9.1.4. Invited Talks

Benoit Baudry:

- Running software in uncertain environments. Invited talk at the ACCESS Distinguished Lecture Series at KTH, Stockholm.
- ECSLER: tool support for runtime evolution inside the JVM at the CREST workshop on genetic improvement, University College London.
- Improving software quality and DevOps automation with STAMP at OW2'con 2016.

Benoit Combemale:

- Towards Language-Oriented Modeling. Invited talk at RWTH Aachen University, Germany.
- Dynamic Validation & Verification in Language-Oriented Modeling. Keynote at MoDeVVA 2016.
- On the Globalization of Modeling Languages. Invited talk at CNES.
- A Bit of Software Engineering! Invited talk at Airbus.
- Modeling for Smart Cyber-Physical Systems. Invited talk at the French CPS days.
- Modeling for Sustainability. Invited talk at Lancaster University, United Kingdom.

9.1.5. Leadership within the Scientific Community

Benoit Baudry:

- Steering committee member for the ACM/IEEE MODELS conference

Benoit Combemale:

- Steering committee member for the ACM SLE conference
- Founding member and member of the advisory board of the GEMOC initiative.

9.1.6. Research Administration

Benoit Baudry is the leader of the DiverSE research team; he is in the scientific advisory board of the SVV lab, University of Luxembourg.

Jean-Marc Jézéquel is Director of IRISA (UMR 6074). He is Coordinator of the academic club of the French Cyber-defense Excellence Cluster, and Director of the Rennes Node of EIT Digital.

9.2. Teaching - Supervision - Juries

9.2.1. Teaching

The DIVERSE team bears the bulk of the teaching on Software Engineering at the University of Rennes 1 and at INSA Rennes, for the first year of the Master of Computer Science (Project Management, Object-Oriented Analysis and Design with UML, Design Patterns, Component Architectures and Frameworks, Validation & Verification, Human-Computer Interaction) and for the second year of the MSc in software engineering (Model driven Engineering, Aspect-Oriented Software Development, Software Product Lines, Component Based Software Development, Validation & Verification, *etc.*).

Each of Jean-Marc Jézéquel, Noël Plouzeau, Olivier Barais, Johann Bourcier, Arnaud Blouin, and Mathieu Acher teaches about 200h in these domains, with Benoit Baudry and Benoit Combemale teaching about 50h, for a grand total of about 1300 hours, including several courses at ENSTB, Supelec, and ENSAI Rennes engineering school.

Olivier Barais is deputy director of the electronics and computer science teaching department of the University of Rennes 1. Mathieu Acher is in charge of teaching duties management of this department. Noël Plouzeau is the head of the final year of the Master in Computer Science at the University of Rennes 1. Johann Bourcier is co-manager of the Home-Automation option at the ESIR engineering school in Rennes.

The DIVERSE team also hosts several MSc and summer trainees every year.

9.2.2. Supervision

- PhD: Sana Ben Nasr [17], *Mining and Modeling Variability from Natural Language Documents: Two Case Studies*, Inria, April 2016, B. Baudry and M. Acher
- PhD: Guillaume Becan [19] *Metamodels and Feature Models: Complementary Approaches to Formalize Product Comparison Matrices*, MESR, September 2016, B. Baudry and M. Acher
- PhD: Thomas Degueule , *Composition and interoperability of domain-specific language engineering*, Univ. Rennes I, 02/12/16
- PhD: Francisco Acosta , *Automatic Deployment and Reconfigurations in Internet of Things*, Univ. Rennes I, 12/12/2016
- PhD: David Mendez Acuna , *Variability in Modeling Languages*, Inria, 16/12/2016
- PhD: Jacky Bourgeois , *Automatic Synchronization of Energy Production and Consumption*, Open University, DATE
- PhD in progress: Mohamed Boussaa, *An Architecture for Testing Large-Scale Dynamic Distributed Systems*, 2013, B. Baudry, O. Barais
- PhD in progress: Alejandro Gomez Boix, *Distributed counter-measure against browser fingerprinting*, 2016, B. Baudry, D. Bromberg
- PhD in progress: Nicolas Harrand, *Automatic diversity for code obfuscation*, 2016, B. Baudry
- PhD in progress: Pierre Laperdrix, *Secretless moving target against browser fingerprinting*, B. Baudry, G. Avoine
- PhD in progress: Johan Pelay, *Langage pour une programmation incrémentale de réseau*, 2016, O. Barais, F. Guillemin
- PhD in progress: Quentin Plazar, *Combining Decision Procedures for Constraint Programming and SMT*, 2015, M. Acher, A. Goetib
- PhD in progress: Marcelino Rodriguez Cancio, *Automatic computation diversification*, 2015, B. Baudry and B. Combemale
- PhD in progress: Paul Temple, *Variability Testing of Computer Vision Algorithms*, 2015, M. Acher, J.-M. Jézéquel
- PhD in progress: Kwaku Yeboah-Antwi, *Runtime emergence of software diversity*, 2013, B. Baudry
- PhD in progress: Manuel Leduc, *TITRE*, 2016, O. Barais, B. Combemale
- PhD in progress: Youssou NDiaye, *Modelling and evaluating security in user interfaces*, 2016, N. Aillery, O. Barais, A. Blouin, A. Bouabdallah
- PhD in progress: Jean-Émile Dartois, *TITRE*, 2016, O. Barais
- PhD in progress: Oscar Luis, *Automatic test amplification*, 2016, B. Baudry
- PhD in progress: Alexandre Rio, *TITRE*, 2016, O. Barais, Y. Morel
- PhD: in progress, Kevin Corre, *Modélisation de la confiance dans les services sociaux et conversationnels*, 2014, O. Barais, G. Sunye
- PhD: in progress, Gwendal Le Moulec, *Towards the automatic synthesis of virtual reality applications*, 2015, B. Arnaldi, A. Blouin, V. Gouranton

9.2.3. Juries

9.2.3.1. Mathieu Acher

was in the examination committee of the following PhD thesis:

- Guillaume Becan, September 2016, Univ Rennes I, Supervisor
- Sana Ben Nasr, April 2016, Univ Rennes I, Supervisor
- Jaime Chavarriaga, December 2016, Vrije Universiteit Brussel, Reviewer

9.2.3.2. Arnaud Blouin

was in the examination committee of the following PhD thesis:

- Thomas Degueule, December 2016, Univ Rennes I, Supervisor

9.2.3.3. Olivier Barais

was in the examination committee of the following PhD thesis:

- Thomas Degueule, December 2016, Univ Rennes I, Supervisor
- Maxime Colman, December 2016, Univ Lille I, Reviewer
- Alexandre Garnier, December 2016, EMN - Univ Nantes, Reviewer
- Matias Ezequiel Vara Larsen, April 2016, Univ Nice, Reviewer
- Djamel Khelladi, September 2016, UPMC, Reviewer
- Simon Dupont, April 2016, EMN - Univ Nantes, Reviewer

9.2.3.4. Benoit Baudry

was in the examination committee of the following PhD thesis:

- Guillaume Becan, September 2016, Univ Rennes I, Supervisor
- Sana Ben Nasr, April 2016, Univ Rennes I, Supervisor
- David Mendez, December 2016, Univ Rennes I, Supervisor

9.2.3.5. Olivier Barais

was in the examination committee of the following HDR:

- Sébastien Leriche, November 2016, ENAC, Reviewer

9.2.3.6. Jean-Marc Jézéquel

was in the examination committee of the following HDR:

- Tewfik Ziadi, December 2016, UPMC, Reviewer

9.2.3.7. Benoit Combemale

was in the examination committee of the following PhD thesis:

- Thomas Degueule, December 2016, Univ Rennes I, Supervisor
- Blazo Nastov, november 2016, Univ. Montpellier, Reviewer
- Florent Latombe, June 2016, Univ Toulouse, Examiner
- Andreas Wortmann, June 2016, Aachen University, Germany, Reviewer
- Matias Ezequiel Vara Larsen, April 2016, Univ Nice, Examiner

9.2.3.8. Johann Bourcier

was in the examination committee of the following PhD thesis:

- Francisco Javier Acosta Padilla, December 2016, Univ Rennes I, Supervisor

9.2.3.9. Jean-Marc Jézéquel

was in the examination committee of the following PhD thesis:

- Jabier Martinez, October 2016, Univ. Luxemburg, Examiner

9.3. Popularization

- IT industry forums. Our recent work on diversity against browser fingerprinting and the associated web site [4] have been demonstrated in the a major cybersecurity forum (FIC'15 [5]), the largest IT event organized by the European commission (ICT'15 [6]) and two open source forums (OWF'14 [7] and FOSSA'14 [8]).
- General-audience press. Our work on browser fingerprinting has been featured Le Monde [9], and two full articles appeared in the MISC [10] and the Clubic magazines [11].

[4] https://amiunique.org/
[5] https://fic2015.com/home/
[6] https://ec.europa.eu/digital-single-market/en/ict2015
[7] http://www.openworldforum.paris/
[8] https://fossa.inria.fr/
[9] http://abonnes.lemonde.fr/sciences/article/2014/11/24/le-fossa-bazar-techno-participatif_4528568_1650684.html
[10] http://connect.ed-diamond.com/MISC/MISC-081/Le-fingerprinting-une-nouvelle-technique-de-tracage

10. Bibliography

Major publications by the team in recent years

[1] B. BAUDRY, M. MONPERRUS. *The Multiple Facets of Software Diversity: Recent Developments in Year 2000 and Beyond*, in "ACM Computing Surveys", 2015, vol. 48, n° 1, pp. 16:1–16:26, https://hal.inria.fr/hal-01182103

[2] A. BLOUIN, N. MOHA, B. BAUDRY, H. SAHRAOUI, J.-M. JÉZÉQUEL. *Assessing the Use of Slicing-based Visualizing Techniques on the Understanding of Large Metamodels*, in "Information and Software Technology", 2015, vol. 62, pp. 124 - 142 [*DOI :* 10.1016/J.INFSOF.2015.02.007], https://hal.inria.fr/hal-01120558

[3] M. BOUSSAA, O. BARAIS, B. BAUDRY, G. SUNYÉ. *NOTICE: A Framework for Non-functional Testing of Compilers*, in "Proc. of the Int. Conf. on Software Quality, Reliability & Security (QRS)", August 2016, https://hal.archives-ouvertes.fr/hal-01344835

[4] G. BÉCAN, M. ACHER, B. BAUDRY, S. BEN NASR. *Breathing Ontological Knowledge Into Feature Model Synthesis: An Empirical Study*, in "Empirical Software Engineering", 2015, vol. 21, n° 4, pp. 1794–1841 [*DOI :* 10.1007/S10664-014-9357-1], https://hal.inria.fr/hal-01096969

[5] G. BÉCAN, N. SANNIER, M. ACHER, O. BARAIS, A. BLOUIN, B. BAUDRY. *Automating the Formalization of Product Comparison Matrices*, in "Proc. of the Int. Conf. on Automated Software Engineering (ASE)", September 2014 [*DOI :* 10.1145/2642937.2643000], https://hal.inria.fr/hal-01058440

[6] B. COMBEMALE, J. DEANTONI, B. BAUDRY, R. B. FRANCE, J.-M. JÉZÉQUEL, J. GRAY. *Globalizing Modeling Languages*, in "IEEE Computer", June 2014, pp. 10-13, https://hal.inria.fr/hal-00994551

[7] B. COMBEMALE, J. DEANTONI, M. E. VARA LARSEN, F. MALLET, O. BARAIS, B. BAUDRY, R. FRANCE. *Reifying Concurrency for Executable Metamodeling*, in "Proc. of the Int. Conf. on Software Language Engineering", October 2013, pp. 365-384 [*DOI :* 10.1007/978-3-319-02654-1_20], https://hal.inria.fr/hal-00850770

[8] J.-M. DAVRIL, E. DELFOSSE, N. HARIRI, M. ACHER, J. CLELANG-HUANG, P. HEYMANS. *Feature Model Extraction from Large Collections of Informal Product Descriptions*, in "Proc. of the Europ. Software Engineering Conf. and the ACM SIGSOFT Symp. on the Foundations of Software Engineering (ESEC/FSE)", September 2013, pp. 290-300 [*DOI :* 10.1145/2491411.2491455], https://hal.inria.fr/hal-00859475

[9] T. DEGUEULE, B. COMBEMALE, A. BLOUIN, O. BARAIS, J.-M. JÉZÉQUEL. *Melange: A Meta-language for Modular and Reusable Development of DSLs*, in "Proc. of the Int. Conf. on Software Language Engineering (SLE)", October 2015, https://hal.inria.fr/hal-01197038

[10] J. A. GALINDO DUARTE, M. ALFEREZ, M. ACHER, B. BAUDRY, D. BENAVIDES. *A Variability-Based Testing Approach for Synthesizing Video Sequences*, in "Proc. of the Int. Symp. on Software Testing and Analysis (ISSTA)", July 2014, https://hal.inria.fr/hal-01003148

[11] http://pro.clubic.com/webmarketing/publicite-en-ligne/actualite-742853-fingerprinting-cookies.html

[11] I. GONZALEZ-HERRERA, J. BOURCIER, E. DAUBERT, W. RUDAMETKIN, O. BARAIS, F. FOUQUET, J.-M. JÉZÉQUEL, B. BAUDRY. *ScapeGoat: Spotting abnormal resource usage in component-based reconfigurable software systems*, in "Journal of Systems and Software", 2016 [*DOI :* 10.1016/J.JSS.2016.02.027], https://hal.inria.fr/hal-01354999

[12] J.-M. JÉZÉQUEL, B. COMBEMALE, O. BARAIS, M. MONPERRUS, F. FOUQUET. *Mashup of Meta-Languages and its Implementation in the Kermeta Language Workbench*, in "Software and Systems Modeling", 2015, vol. 14, n^{o} 2, pp. 905-920, https://hal.inria.fr/hal-00829839

[13] P. LAPERDRIX, W. RUDAMETKIN, B. BAUDRY. *Beauty and the Beast: Diverting modern web browsers to build unique browser fingerprints*, in "Proc. of the Symp. on Security and Privacy (S&P)", May 2016, https://hal.inria.fr/hal-01285470

[14] M. RODRIGUEZ-CANCIO, B. COMBEMALE, B. BAUDRY. *Automatic Microbenchmark Generation to Prevent Dead Code Elimination and Constant Folding*, in "Proc. of the Int. Conf. on Automated Software Engineering (ASE)", September 2016, https://hal.inria.fr/hal-01343818

[15] M. TRICOIRE, O. BARAIS, M. LEDUC, J. BOURCIER, F. FOUQUET, G. NAIN, M. LUDOVIC, G. SUNYÉ, B. MORIN. *KevoreeJS: Enabling Dynamic Software Reconfigurations in the Browser*, in "Proc. of WICSA and CompArch", April 2016 [*DOI :* 10.1109/CBSE.2016.20], https://hal.inria.fr/hal-01354997

Publications of the year

Doctoral Dissertations and Habilitation Theses

[16] F. J. ACOSTA PADILLA. *Self-adaptation for Internet of Things applications*, Université de Rennes 1, France, December 2016, https://hal.inria.fr/tel-01426219

[17] S. BEN NASR. *Mining and Modeling Variability from Natural Language Documents: Two Case Studies*, Université Rennes 1, April 2016, https://hal.inria.fr/tel-01388392

[18] J. BOURGEOIS. *Interactive Demand-Shifting in the Context of Domestic Micro-Generation*, The Open University ; Université de Rennes 1 [UR1], June 2016, https://hal.inria.fr/tel-01385022

[19] G. BÉCAN. *Metamodels and Feature Models: Complementary Approaches to Formalize Product Comparison Matrices*, Université Rennes 1, September 2016, https://hal.inria.fr/tel-01416129

[20] T. DEGUEULE. *Composition and Interoperability for External Domain-Specific Language Engineering*, Université de Rennes 1 [UR1], December 2016, https://hal.inria.fr/tel-01427009

[21] D. MÉNDEZ-ACUÑA. *Leveraging Software Product Lines Engineering in the Construction of Domain Specific Languages*, Université de Rennes 1, France, December 2016, https://hal.archives-ouvertes.fr/tel-01427187

Articles in International Peer-Reviewed Journals

[22] S. BEN NASR, G. BÉCAN, M. ACHER, J. F. F. BOSCO, N. SANNIER, B. BAUDRY, J.-M. DAVRIL. *Automated Extraction of Product Comparison Matrices From Informal Product Descriptions*, in "Journal of Systems and Software", 2017, vol. 124, pp. 82 - 103 [*DOI :* 10.1016/J.JSS.2016.11.018], https://hal.inria.fr/hal-01427218

[23] J. F. F. BOSCO, M. ACHER, O. BARAIS. *Software Unbundling: Challenges and Perspectives*, in "Transactions on Modularity and Composition", May 2016, https://hal.inria.fr/hal-01427560

[24] T. DEGUEULE, B. COMBEMALE, A. BLOUIN, O. BARAIS, J.-M. JÉZÉQUEL. *Safe Model Polymorphism for Flexible Modeling*, in "Computer Languages, Systems and Structures", September 2016, https://hal.inria.fr/hal-01367305

[25] I. GONZALEZ-HERRERA, J. BOURCIER, E. DAUBERT, W. RUDAMETKIN, O. BARAIS, F. FOUQUET, J.-M. JÉZÉQUEL, B. BAUDRY. *ScapeGoat: Spotting abnormal resource usage in component-based reconfigurable software systems*, in "Journal of Systems and Software", 2016 [*DOI :* 10.1016/J.JSS.2016.02.027], https://hal.inria.fr/hal-01354999

[26] A. HERVIEU, D. MARIJAN, A. GOTLIEB, B. BAUDRY. *Optimal Minimisation of Pairwise-covering Test Configurations Using Constraint Programming*, in "Information and Software Technology", March 2016, vol. 71, pp. 129-146, https://hal.inria.fr/hal-01352831

[27] M. KEZADRI, M. PANTEL, X. THIRIOUX, B. COMBEMALE. *Correct-by-construction model driven engineering composition operators*, in "Formal Aspects of Computing", January 2016, vol. 28, n° 3 [*DOI :* 10.1007/S00165-016-0354-6], https://hal.inria.fr/hal-01319576

[28] G. KHANDU NARWANE, J. A. GALINDO DUARTE, S. NARAYANAN KRISHNA, D. BENAVIDES, J.-V. MILLO, S. RAMESH. *Traceability analyses between features and assets in software product lines*, in "Entropy", July 2016, https://hal.inria.fr/hal-01342351

[29] D. MÉNDEZ-ACUÑA, J. A. GALINDO DUARTE, T. DEGUEULE, B. COMBEMALE, B. BAUDRY. *Leveraging Software Product Lines Engineering in the Development of External DSLs: A Systematic Literature Review*, in "Computer Languages, Systems and Structures", September 2016 [*DOI :* 10.1016/J.CL.2016.09.004], https://hal.archives-ouvertes.fr/hal-01372702

[30] J. XUAN, B. CORNU, M. MARTINEZ, B. BAUDRY, L. SEINTURIER, M. MONPERRUS. *B-Refactoring: Automatic Test Code Refactoring to Improve Dynamic Analysis*, in "Information and Software Technology", 2016, vol. 76, pp. 65-80 [*DOI :* 10.1016/J.INFSOF.2016.04.016], https://hal.archives-ouvertes.fr/hal-01309004

[31] K. YEBOAH-ANTWI, B. BAUDRY. *Online Genetic Improvement on the java virtual machine with ECSELR*, in "Genetic Programming and Evolvable Machines", October 2016, pp. 1-27 [*DOI :* 10.1007/S10710-016-9278-4], https://hal.inria.fr/hal-01382964

Invited Conferences

[32] G. PERROUIN, M. ACHER, J.-M. DAVRIL, A. LEGAY, P. HEYMANS. *A Complexity Tale: Web Configurators*, in "VACE 2016 - 1st International Workshop on Variability and Complexity in Software Design Pages (co-located with ICSE'16)", Austin, United States, May 2016, pp. 28 - 31 [*DOI :* 10.1145/2897045.2897051], https://hal.inria.fr/hal-01427165

International Conferences with Proceedings

[33] A. ALLEN, C. ARAGON, C. BECKER, J. C. CARVER, A. CHIS, B. COMBEMALE, M. CROUCHER, K. CROWSTON, D. GARIJO, A. GEHANI, C. GOBLE, R. HAINES, R. HIRSCHFELD, J. HOWISON, K. HUFF, C. JAY, D. KATZ, C. KIRCHNER, K. KUKSENOK, R. LÄMMEL, O. NIERSTRASZ, M. TURK, R. V. VAN NIEUWPOORT, M. VAUGHN, J. VINJU. *Lightning Talk:"I solemnly pledge" A Manifesto for Personal*

Responsibility in the Engineering of Academic Software, in "Fourth Workshop on Sustainable Software for Science: Practice and Experiences (WSSSPE4)", Manchester, United Kingdom, G. ALLEN, J. CARVER, M. HEROUX, L. J. HWANG, D. S. KATZ, K. E. NIEMEYER, M. PARASHAR, C. C. VENTERS, S.-C. T. CHOI, T. CRICK, M. R. CRUSOE, S. GESING, R. HAINES (editors), Proceedings of the Fourth Workshop on Sustainable Software for Science: Practice and Experiences (WSSSPE4), CEUR, 2016, vol. 1686, https://hal.inria.fr/hal-01367344

[34] O. BARAIS, J. BOURCIER, Y.-D. BROMBERG, C. DION. *Towards microservices architecture to transcode videos in the large at low costs*, in "TEMU 2016 - International Conference on Telecommunications and Multimedia", Heraklion, Greece, July 2016, pp. 1 - 6 [*DOI :* 10.1109/TEMU.2016.7551918], https://hal.inria.fr/hal-01427277

[35] M. BOUSSAA, O. BARAIS, B. BAUDRY, G. SUNYÉ. *Automatic Non-functional Testing of Code Generators Families*, in "15th International Conference on Generative Programming: Concepts & Experiences (GPCE 2016)", Amsterdam, Netherlands, Proceedings of the 2016 ACM SIGPLAN International Conference on Generative Programming: Concepts and Experiences, October 2016, 11 p. , https://hal.inria.fr/hal-01356849

[36] M. BOUSSAA, O. BARAIS, B. BAUDRY, G. SUNYÉ. *NOTICE: A Framework for Non-functional Testing of Compilers*, in "2016 IEEE International Conference on Software Quality, Reliability & Security (QRS 2016)", Vienna, Austria, IEEE Computer Society Conference Publishing Services (CPS), August 2016, https://hal.archives-ouvertes.fr/hal-01344835

[37] E. BOUSSE, T. DEGUEULE, D. VOJTISEK, T. MAYERHOFER, J. DEANTONI, B. COMBEMALE. *Execution Framework of the GEMOC Studio (Tool Demo)*, in "Proceedings of the 2016 ACM SIGPLAN International Conference on Software Language Engineering", Amsterdam, Netherlands, SLE 2016, October 2016, 8 p. , https://hal.inria.fr/hal-01355391

[38] B. COMBEMALE, C. BRUN, J. CHAMPEAU, X. CRÉGUT, J. DEANTONI, J. LE NOIR. *A Tool-Supported Approach for Concurrent Execution of Heterogeneous Models*, in "8th European Congress on Embedded Real Time Software and Systems (ERTS 2016)", Toulouse, France, 2016, https://hal.inria.fr/hal-01258358

[39] B. COMBEMALE, B. H. CHENG, A. MOREIRA, J.-M. BRUEL, J. GRAY. *Modeling for Sustainability*, in "Modeling in Software Engineering 2016 (MiSE'16)", Austin, United States, Modeling in Software Engineering 2016 (MiSE'16), ACM, 2016, https://hal.inria.fr/hal-01185800

[40] J. A. GALINDO DUARTE, M. ACHER, J. M. TIRADO, C. VIDAL, B. BAUDRY, D. BENAVIDES. *Exploiting the Enumeration of All Feature Model Configurations: A New Perspective with Distributed Computing*, in "Software Product Line Conference", Beijing, China, September 2016, https://hal.archives-ouvertes.fr/hal-01334851

[41] I. GONZALEZ-HERRERA, J. BOURCIER, W. RUDAMETKIN, O. BARAIS, F. FOUQUET. *Squirrel: Architecture Driven Resource Management*, in "SAC - 31st Annual ACM Symposium on Applied Computing", Pisa, Italy, April 2016 [*DOI :* 10.1145/0000000.0000000], https://hal.inria.fr/hal-01355000

[42] A. JANGDA, M. MISHRA, B. BAUDRY. *libmask: Protecting Browser JIT Engines from the Devil in the Constants*, in "Annual Conference on Privacy, Security and Trust", Auckland, New Zealand, December 2016, https://hal.inria.fr/hal-01382971

[43] J. KIENZLE, G. MUSSBACHER, O. ALAM, M. SCHÖTTLE, N. BELLOIR, P. COLLET, B. COMBEMALE, J. DEANTONI, J. KLEIN, B. RUMPE. *VCU: The Three Dimensions of Reuse*, in "The 15th International Conference on Software Reuse (ICSR-15)", Limassol, Cyprus, The 15th International Conference on Software Reuse, 2016, https://hal.inria.fr/hal-01287720

[44] J. H. KIM, A. LEGAY, L.-M. TRAONOUEZ, M. ACHER, S. KANG. *A Formal Modeling and Analysis Framework for Software Product Line of Preemptive Real-Time Systems*, in "Symposium on Applied Computing", Pise, Italy, Proceedings of the 31st Annual ACM Symposium on Applied Computing, ACM, April 2016, pp. 1562 - 1565 [*DOI :* 10.1145/2851613.2851977], https://hal.archives-ouvertes.fr/hal-01241673

[45] P. LAPERDRIX, W. RUDAMETKIN, B. BAUDRY. *Beauty and the Beast: Diverting modern web browsers to build unique browser fingerprints*, in "37th IEEE Symposium on Security and Privacy (S&P 2016)", San Jose, United States, May 2016, https://hal.inria.fr/hal-01285470

[46] G. LE GUERNIC, B. COMBEMALE, J. A. GALINDO DUARTE. *Industrial Experience Report on the Formal Specification of a Packet Filtering Language Using the K Framework*, in "3rd Workshop on Formal Integrated Development Environment", Limassol, Cyprus, Catherine Dubois and Dominique Mery and Paolo Masci, November 2016, https://hal.inria.fr/hal-01401849

[47] J. LE NOIR, S. MADELÉNAT, C. LABREUCHE, O. CONSTANT, G. GAILLIARD, M. ACHER, O. BARAIS. *A Decision-making Process for Exploring Architectural Variants in Systems Engineering*, in "Software Product Lines Conference (SPLC)", Beijing, China, September 2016 [*DOI :* 10.1145/1235], https://hal.inria.fr/hal-01374140

[48] V. LELLI, A. BLOUIN, B. BAUDRY, F. COULON, O. BEAUDOUX. *Automatic Detection of GUI Design Smells: The Case of Blob Listener*, in "Proceedings of the 8th ACM SIGCHI Symposium on Engineering Interactive Computing Systems (EICS'16)", Brussels, Belgium, June 2016, pp. 263-274 [*DOI :* 10.1145/2933242.2933260], https://hal.inria.fr/hal-01308625

[49] B. MORIN, F. FLEUREY, K. E. HUSA, O. BARAIS. *A Generative Middleware for Heterogeneous and Distributed Services*, in "19th International ACM Sigsoft Symposium on Component-Based Software Engineering (CBSE 2016)", Venise, Italy, 19th International ACM Sigsoft Symposium on Component-Based Software Engineering, April 2016 [*DOI :* 10.1109/CBSE.2016.12], https://hal.archives-ouvertes.fr/hal-01356104

[50] D. MÉNDEZ-ACUÑA, J. A. GALINDO DUARTE, B. COMBEMALE, A. BLOUIN, B. BAUDRY, G. LE GUERNIC. *Reverse-engineering reusable language modules from legacy domain-specific languages*, in "International Conference on Software Reuse", Limassol, Cyprus, Proceedings of the International Conference on Software Reuse, June 2016, https://hal.archives-ouvertes.fr/hal-01284816

[51] D. MÉNDEZ-ACUÑA, J. A. GALINDO DUARTE, B. COMBEMALE, A. BLOUIN, B. BAUDRY. *Puzzle: A tool for analyzing and extracting specification clones in DSLs*, in "ICSR 2016 the 15th International Conference on Software Reuse", Limassol, Cyprus, Proceedings of the 15th International Conference on Software Reuse, June 2016, https://hal.archives-ouvertes.fr/hal-01284822

[52] E. OUTIN, J.-E. DARTOIS, O. BARAIS, J.-L. PAZAT. *Seeking for the Optimal Energy Modelisation Accuracy to Allow Efficient Datacenter Optimizations*, in "16th International Symposium on Cluster, Cloud and Grid Computing", Cartagena, Italy, IEEE/ACM 16th International Symposium on Cluster, Cloud and Grid Computing, May 2016 [*DOI :* 10.1109/CCGRID.2016.67], https://hal.archives-ouvertes.fr/hal-01356099

[53] G. PERROUIN, M. AMRANI, M. ACHER, B. COMBEMALE, A. LEGAY, P.-Y. SCHOBBENS. *Featured model types: Towards Systematic Reuse in Modelling Language Engineering*, in "MiSE '16 - 8th International Workshop on Modeling in Software Engineering", New york, United States, ACM, May 2016, pp. 1 - 7 [*DOI :* 10.1145/2896982.2896987], https://hal.inria.fr/hal-01406507

[54] M. RODRIGUEZ-CANCIO, B. COMBEMALE, B. BAUDRY. *Automatic Microbenchmark Generation to Prevent Dead Code Elimination and Constant Folding*, in "31st IEEE/ACM International Conference on Automated Software Engineering (ASE 2016)", Singapore, Singapore, ASE 2016:Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering, September 2016, https://hal.inria.fr/hal-01343818

[55] P. TEMPLE, J. A. GALINDO DUARTE, M. ACHER, J.-M. JÉZÉQUEL. *Using Machine Learning to Infer Constraints for Product Lines*, in "Software Product Line Conference (SPLC)", Beijing, China, September 2016 [*DOI :* 10.1145/2934466.2934472], https://hal.inria.fr/hal-01323446

[56] M. TRICOIRE, O. BARAIS, M. LEDUC, J. BOURCIER, F. FOUQUET, G. NAIN, M. LUDOVIC, G. SUNYÉ, B. MORIN. *KevoreeJS: Enabling Dynamic Software Reconfigurations in the Browser*, in "WICSA and CompArch 2016", Venice, Italy, ACM Sigsoft, April 2016 [*DOI :* 10.1109/CBSE.2016.20], https://hal.inria.fr/hal-01354997

[57] M. VINUEZA, J. L. RODAS, J. A. GALINDO DUARTE, D. BENAVIDES. *El uso de modelos de características con atributos para pruebas en sistemas de alta variabilidad: primeros pasos*, in "CEDI 2016", Salamanca, Spain, September 2016, https://hal.inria.fr/hal-01342354

National Conferences with Proceedings

[58] J. L. RODAS, J. OLIVARES, J. A. GALINDO DUARTE, D. BENAVIDES. *Hacia el uso de sistemas de recomendación en sistemas de alta variabilidad*, in "CEDI 2016", Salamanca, Spain, September 2016, https://hal.inria.fr/hal-01342353

Scientific Books (or Scientific Book chapters)

[59] B. COMBEMALE, R. FRANCE, J.-M. JÉZÉQUEL, B. RUMPE, J. R. STEEL, D. VOJTISEK. *Engineering Modeling Languages: Turning Domain Knowledge into Tools*, Chapman and Hall/CRC, November 2016, 398 p. , https://hal.inria.fr/hal-01355374

[60] T. DEGUEULE, B. COMBEMALE, J.-M. JÉZÉQUEL. *On Language Interfaces*, in "PAUSE: Present And Ulterior Software Engineering", B. MEYER, M. MAZZARA (editors), Springer, February 2017, https://hal.inria.fr/hal-01424909

Research Reports

[61] M. ACHER, F. ESNAULT. *Large-scale Analysis of Chess Games with Chess Engines: A Preliminary Report*, Inria Rennes Bretagne Atlantique, April 2016, n^{o} RT-0479, https://hal.inria.fr/hal-01307091

[62] B. DANGLOT, P. PREUX, B. BAUDRY, M. MONPERRUS. *Correctness Attraction: A Study of Stability of Software Behavior Under Runtime Perturbation*, HAL, 2016, n^{o} hal-01378523, https://hal.archives-ouvertes.fr/hal-01378523

[63] G. LE GUERNIC, J. A. GALINDO DUARTE. *Experience Report on the Formal Specification of a Packet Filtering Language Using the K Framework*, Inria Rennes Bretagne Atlantique, October 2016, n° RR-8967, 41 p. , https://hal.inria.fr/hal-01385541

Other Publications

[64] T. DEGUEULE. *Interoperability and Composition of DSLs with Melange*, June 2016, working paper or preprint, https://hal.inria.fr/hal-01336940

[65] M. RODRIGUEZ-CANCIO, B. COMBEMALE, B. BAUDRY. *Approximate Loop Unrolling*, November 2016, working paper or preprint, https://hal.inria.fr/hal-01401828

References in notes

[66] A. ARCURI, L. C. BRIAND. *A practical guide for using statistical tests to assess randomized algorithms in software engineering*, in "ICSE", 2011, pp. 1-10

[67] A. AVIZIENIS. *The N-version approach to fault-tolerant software*, in "Software Engineering, IEEE Transactions on", 1985, n° 12, pp. 1491–1501

[68] F. BACHMANN, L. BASS. *Managing variability in software architectures*, in "SIGSOFT Softw. Eng. Notes", 2001, vol. 26, n° 3, pp. 126–132

[69] F. BALARIN, Y. WATANABE, H. HSIEH, L. LAVAGNO, C. PASSERONE, A. SANGIOVANNI-VINCENTELLI. *Metropolis: An integrated electronic system design environment*, in "Computer", 2003, vol. 36, n° 4, pp. 45–52

[70] E. BANIASSAD, S. CLARKE. *Theme: an approach for aspect-oriented analysis and design*, in "26th International Conference on Software Engineering (ICSE)", 2004, pp. 158-167

[71] E. G. BARRANTES, D. H. ACKLEY, S. FORREST, D. STEFANOVIĆ. *Randomized instruction set emulation*, in "ACM Transactions on Information and System Security (TISSEC)", 2005, vol. 8, n° 1, pp. 3–40

[72] D. BATORY, R. E. LOPEZ-HERREJON, J.-P. MARTIN. *Generating Product-Lines of Product-Families*, in "ASE '02: Automated software engineering", IEEE, 2002, pp. 81–92

[73] S. BECKER, H. KOZIOLEK, R. REUSSNER. *The Palladio component model for model-driven performance prediction*, in "Journal of Systems and Software", January 2009, vol. 82, n° 1, pp. 3–22

[74] N. BENCOMO. *On the use of software models during software execution*, in "MISE '09: Proceedings of the 2009 ICSE Workshop on Modeling in Software Engineering", IEEE Computer Society, May 2009

[75] A. BEUGNARD, J.-M. JÉZÉQUEL, N. PLOUZEAU. *Contract Aware Components, 10 years after*, in "WCSI", 2010, pp. 1-11

[76] J. BOSCH. *Design and use of software architectures: adopting and evolving a product-line approach*, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 2000

[77] J. BOSCH, G. FLORIJN, D. GREEFHORST, J. KUUSELA, J. H. OBBINK, K. POHL. *Variability Issues in Software Product Lines*, in "PFE '01: Revised Papers from the 4th International Workshop on Software Product-Family Engineering", London, UK, Springer-Verlag, 2002, pp. 13–21

[78] L. C. BRIAND, E. ARISHOLM, S. COUNSELL, F. HOUDEK, P. THÉVENOD–FOSSE. *Empirical studies of object-oriented artifacts, methods, and processes: state of the art and future directions*, in "Empirical Software Engineering", 1999, vol. 4, n^{o} 4, pp. 387–404

[79] J. T. BUCK, S. HA, E. A. LEE, D. G. MESSERSCHMITT. *Ptolemy: A framework for simulating and prototyping heterogeneous systems*, in "Int. Journal of Computer Simulation", 1994

[80] T. BURES, P. HNETYNKA, F. PLASIL. *Sofa 2.0: Balancing advanced features in a hierarchical component model*, in "Software Engineering Research, Management and Applications, 2006. Fourth International Conference on", IEEE, 2006, pp. 40–48

[81] B. H. C. CHENG, R. LEMOS, H. GIESE, P. INVERARDI, J. MAGEE, J. ANDERSSON, B. BECKER, N. BENCOMO, Y. BRUN, B. CUKIC, G. MARZO SERUGENDO, S. DUSTDAR, A. FINKELSTEIN, C. GACEK, K. GEIHS, V. GRASSI, G. KARSAI, H. M. KIENLE, J. KRAMER, M. LITOIU, S. MALEK, R. MIRANDOLA, H. A. MÜLLER, S. PARK, M. SHAW, M. TICHY, M. TIVOLI, D. WEYNS, J. WHITTLE. , D. HUTCHISON, T. KANADE, J. KITTLER, J. M. KLEINBERG, F. MATTERN, J. C. MITCHELL, M. NAOR, O. NIERSTRASZ, C. PANDU RANGAN, B. STEFFEN, M. SUDAN, D. TERZOPOULOS, D. TYGAR, M. Y. VARDI, G. WEIKUM, B. H. C. CHENG, R. LEMOS, H. GIESE, P. INVERARDI, J. MAGEE (editors) *Software Engineering for Self-Adaptive Systems: A Research Roadmap* , Betty H. C. Cheng, Rogério de Lemos, Holger Giese, Paola Inverardi, and Jeff Magee, Springer Berlin Heidelberg, Berlin, Heidelberg, 2009, vol. 5525

[82] J. COPLIEN, D. HOFFMAN, D. WEISS. *Commonality and Variability in Software Engineering*, in "IEEE Software", 1998, vol. 15, n^{o} 6, pp. 37–45

[83] I. CRNKOVIC, S. SENTILLES, A. VULGARAKIS, M. R. CHAUDRON. *A classification framework for software component models*, in "Software Engineering, IEEE Transactions on", 2011, vol. 37, n^{o} 5, pp. 593–615

[84] K. CZARNECKI, U. W. EISENECKER. *Generative programming: methods, tools, and applications*, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 2000

[85] R. DEMILLI, A. J. OFFUTT. *Constraint-based automatic test data generation*, in "Software Engineering, IEEE Transactions on", 1991, vol. 17, n^{o} 9, pp. 900–910

[86] K. DEB, A. PRATAP, S. AGARWAL, T. MEYARIVAN. *A fast and elitist multiobjective genetic algorithm: NSGA-II*, in "Evolutionary Computation, IEEE Transactions on", 2002, vol. 6, n^{o} 2, pp. 182–197

[87] S. FORREST, A. SOMAYAJI, D. H. ACKLEY. *Building diverse computer systems*, in "Operating Systems, 1997., The Sixth Workshop on Hot Topics in", IEEE, 1997, pp. 67–72

[88] R. B. FRANCE, B. RUMPE. *Model-driven Development of Complex Software: A Research Roadmap*, in "Proceedings of the Future of Software Engineering Symposium (FOSE '07)", L. C. BRIAND, A. L. WOLF (editors), IEEE, 2007, pp. 37–54

[89] S. FREY, F. FITTKAU, W. HASSELBRING. *Search-based genetic optimization for deployment and reconfiguration of software in the cloud*, in "Proceedings of the 2013 International Conference on Software Engineering", IEEE Press, 2013, pp. 512–521

[90] G. HALMANS, K. POHL. *Communicating the Variability of a Software-Product Family to Customers*, in "Software and System Modeling", 2003, vol. 2, n° 1, pp. 15-36

[91] C. HARDEBOLLE, F. BOULANGER. *ModHel'X: A component-oriented approach to multi-formalism modeling*, in "Models in Software Engineering", Springer, 2008, pp. 247–258

[92] M. HARMAN, B. F. JONES. *Search-based software engineering*, in "Information and Software Technology", 2001, vol. 43, n° 14, pp. 833–839

[93] H. HEMMATI, L. C. BRIAND, A. ARCURI, S. ALI. *An enhanced test case selection approach for model-based testing: an industrial case study*, in "SIGSOFT FSE", 2010, pp. 267-276

[94] J. HUTCHINSON, J. WHITTLE, M. ROUNCEFIELD, S. KRISTOFFERSEN. *Empirical assessment of MDE in industry*, in "Proceedings of the 33rd International Conference on Software Engineering (ICSE '11)", R. N. TAYLOR, H. GALL, N. MEDVIDOVIC (editors), ACM, 2011, pp. 471–480

[95] J.-M. JÉZÉQUEL. *Model Driven Design and Aspect Weaving*, in "Journal of Software and Systems Modeling (SoSyM)", may 2008, vol. 7, n° 2, pp. 209–218, http://www.irisa.fr/triskell/publis/2008/Jezequel08a.pdf

[96] K. C. KANG, S. G. COHEN, J. A. HESS, W. E. NOVAK, A. S. PETERSON. *Feature-Oriented Domain Analysis (FODA) Feasibility Study*, Carnegie-Mellon University Software Engineering Institute, November 1990

[97] J. KRAMER, J. MAGEE. *Self-Managed Systems: an Architectural Challenge*, in "Future of Software Engineering", IEEE, 2007, pp. 259–268

[98] K.-K. LAU, P. V. ELIZONDO, Z. WANG. *Exogenous connectors for software components*, in "Component-Based Software Engineering", Springer, 2005, pp. 90–106

[99] P. MCMINN. *Search-based software test data generation: a survey*, in "Software Testing, Verification and Reliability", 2004, vol. 14, n° 2, pp. 105–156

[100] J. MEEKEL, T. B. HORTON, C. MELLONE. *Architecting for Domain Variability*, in "ESPRIT ARES Workshop", 1998, pp. 205-213

[101] A. M. MEMON. *An event-flow model of GUI-based applications for testing*, in "Software Testing, Verification and Reliability", 2007, vol. 17, n° 3, pp. 137–157

[102] B. MORIN, O. BARAIS, J.-M. JÉZÉQUEL, F. FLEUREY, A. SOLBERG. *Models at Runtime to Support Dynamic Adaptation*, in "IEEE Computer", October 2009, pp. 46-53, http://www.irisa.fr/triskell/publis/2009/Morin09f.pdf

[103] P.-A. MULLER, F. FLEUREY, J.-M. JÉZÉQUEL. *Weaving Executability into Object-Oriented Meta-Languages*, in "Proc. of MODELS/UML'2005", Jamaica, LNCS, Springer, 2005

[104] R. MÉLISSON, P. MERLE, D. ROMERO, R. ROUVOY, L. SEINTURIER. *Reconfigurable run-time support for distributed service component architectures*, in "the IEEE/ACM international conference", New York, New York, USA, ACM Press, 2010, 171 p.

[105] C. NEBUT, Y. LE TRAON, J.-M. JÉZÉQUEL. *System Testing of Product Families: from Requirements to Test Cases*, Springer Verlag, 2006, pp. 447–478, http://www.irisa.fr/triskell/publis/2006/Nebut06b.pdf

[106] C. NEBUT, S. PICKIN, Y. LE TRAON, J.-M. JÉZÉQUEL. *Automated Requirements-based Generation of Test Cases for Product Families*, in "Proc. of the 18th IEEE International Conference on Automated Software Engineering (ASE'03)", 2003, http://www.irisa.fr/triskell/publis/2003/nebut03b.pdf

[107] L. M. NORTHROP. *SEI's Software Product Line Tenets*, in "IEEE Softw.", 2002, vol. 19, n^{o} 4, pp. 32–40

[108] L. M. NORTHROP. *A Framework for Software Product Line Practice*, in "Proceedings of the Workshop on Object-Oriented Technology", Springer-Verlag London, UK, 1999, pp. 365–376

[109] I. OBER, S. GRAF, I. OBER. *Validating timed UML models by simulation and verification*, in "International Journal on Software Tools for Technology Transfer", 2006, vol. 8, n^{o} 2, pp. 128–145

[110] D. L. PARNAS. *On the Design and Development of Program Families*, in "IEEE Trans. Softw. Eng.", 1976, vol. 2, n^{o} 1, pp. 1–9

[111] S. PICKIN, C. JARD, T. JÉRON, J.-M. JÉZÉQUEL, Y. LE TRAON. *Test Synthesis from UML Models of Distributed Software*, in "IEEE Transactions on Software Engineering", April 2007, vol. 33, n^{o} 4, pp. 252–268

[112] K. POHL, G. BÖCKLE, F. J. VAN DER LINDEN. *Software Product Line Engineering: Foundations, Principles and Techniques*, Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005

[113] B. RANDELL. *System structure for software fault tolerance*, in "Software Engineering, IEEE Transactions on", 1975, n^{o} 2, pp. 220–232

[114] M. RINARD. *Obtaining and reasoning about good enough software*, in "Proceedings of Annual Design Automation Conference (DAC)", 2012, pp. 930-935

[115] J. ROTHENBERG, L. E. WIDMAN, K. A. LOPARO, N. R. NIELSEN. *The Nature of Modeling*, in "in Artificial Intelligence, Simulation and Modeling", John Wiley & Sons, 1989, pp. 75–92

[116] P. RUNESON, M. HÖST. *Guidelines for conducting and reporting case study research in software engineering*, in "Empirical Software Engineering", 2009, vol. 14, n^{o} 2, pp. 131–164

[117] D. SCHMIDT. *Guest Editor's Introduction: Model-Driven Engineering*, in "IEEE Computer", 2006, vol. 39, n^{o} 2, pp. 25–31

[118] F. SHULL, J. SINGER, D. I. SJBERG. *Guide to advanced empirical software engineering*, Springer, 2008

[119] S. SIDIROGLOU-DOUSKOS, S. MISAILOVIC, H. HOFFMANN, M. RINARD. *Managing performance vs. accuracy trade-offs with loop perforation*, in "Proc. of the Symp. on Foundations of software engineering", New York, NY, USA, ESEC/FSE '11, ACM, 2011, pp. 124-134

[120] J. STEEL, J.-M. JÉZÉQUEL. *On Model Typing*, in "Journal of Software and Systems Modeling (SoSyM)", December 2007, vol. 6, n^{o} 4, pp. 401–414, http://www.irisa.fr/triskell/publis/2007/Steel07a.pdf

[121] C. SZYPERSKI, D. GRUNTZ, S. MURER. *Component software: beyond object-oriented programming*, Addison-Wesley, 2002

[122] J.-C. TRIGAUX, P. HEYMANS. *Modelling variability requirements in Software Product Lines: a comparative survey*, FUNDP Namur, 2003

[123] M. UTTING, B. LEGEARD. *Practical model-based testing: a tools approach*, Morgan Kaufmann, 2010

[124] P. VROMANT, D. WEYNS, S. MALEK, J. ANDERSSON. *On interacting control loops in self-adaptive systems*, in "SEAMS 2011", ACM, 2011, pp. 202–207

[125] C. YILMAZ, M. B. COHEN, A. A. PORTER. *Covering arrays for efficient fault characterization in complex configuration spaces*, in "Software Engineering, IEEE Transactions on", 2006, vol. 32, n^{o} 1, pp. 20–34

[126] Z. A. ZHU, S. MISAILOVIC, J. A. KELNER, M. RINARD. *Randomized accuracy-aware program transformations for efficient approximate computations*, in "Proc. of the Symp. on Principles of Programming Languages (POPL)", 2012, pp. 441-454

[127] T. ZIADI, J.-M. JÉZÉQUEL. *Product Line Engineering with the UML: Deriving Products*, Springer Verlag, 2006, pp. 557-586