Activity Report 2017

Team DIVERSE

Diversity-centric Software Engineering

Joint team with Inria Rennes – Bretagne Atlantique

D4 – Language 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:

A1.2.1. - Dynamic reconfiguration
A2.1.2. - Object-oriented programming
A2.1.10. - Domain-specific languages
A2.5. - Software engineering
A2.5.1. - Software Architecture & Design
A2.5.2. - Component-based Design
A2.5.3. - Empirical Software Engineering
A2.5.4. - Software Maintenance & Evolution
A2.6.2. - Middleware
A4.4. - Security of equipment and software
A4.8. - Privacy-enhancing technologies

Other Research Topics and Application Domains:

B3.1. - Sustainable development
B3.1.1. - Resource management
B6.1. - Software industry
B6.1.1. - Software engineering
B6.1.2. - Software evolution, maintenance
B6.4. - Internet of things
B6.5. - Information systems
B6.6. - Embedded systems
B8.1.2. - Sensor networks for smart buildings
B9.4.1. - Computer science
B9.8. - Privacy

1. Personnel

Research Scientist

Benoit Baudry [Team leader, Inria, Researcher, until Aug 2017, HDR]

Faculty Members

Mathieu Acher [Univ de Rennes I, Associate Professor]
Jean-Marc Jezequel [Univ de Rennes I, Professor, HDR]
Olivier Barais [Univ de Rennes I, Professor, HDR]
Arnaud Blouin [INSA Rennes, Associate Professor]
Johann Bourcier [Univ de Rennes I, Associate Professor]
Benoit Combemale [Univ de Rennes I, Associate Professor, until Aug 2017, HDR]
Noel Plouzeau [Univ de Rennes I, Associate Professor]

PhD Students

Mohamed Boussaa [Univ de Rennes I, Researcher, until Aug 2017]

Kevin Corre [Orange Labs]
Jean-Emile Dartois [Institut de recherche technologique B-com]
Alejandro Gomez Boix [Inria]
Nicolas Harrand [Inria]
Pierre Laperdrix [Univ de Rennes I, until Sep 2017]
Gwendal Le Moulec [INSA Rennes]
Manuel Leduc [Univ de Rennes I]
Dorian Leroy [Université étrangère]
Ludovic Mouline [Gouvernement étranger]
Youssou Ndiaye [Orange Labs]
Johan Pelay [Institut de recherche technologique B-com]
Quentin Plazar [Inria]
Marcelino Rodriguez Cancio [Univ de Rennes I]
Paul Temple [Univ de Rennes I]
Oscar Luis Vera Perez [Inria]

Technical staff

Amine Benelallam [Univ de Rennes I]
Fabien Coulon [Univ de Rennes I]
Caroline Landry [Inria]
Pierre Laperdrix [Inria, from Oct 2017]
Tanja Mayerhofer [Univ de Rennes I, from Mar 2017 until May 2017]
Maxime Tricoire [Inria]
Andreas Wortmann [Univ de Rennes I, from Feb 2017 until May 2017]

Interns

Antoine Cheron [Univ de Rennes I, from Mar 2017 until Sep 2017]
Axel Halin [Univ de Rennes I, until Jan 2017]
Hugo Martin [Univ de Rennes I, from May 2017 until Aug 2017]
Koko Armando Nguepi Kenfack [Univ de Rennes I, from Oct 2017]
Alexandre Nuttinck [Univ de Rennes I, until Jan 2017]

Administrative Assistant

Tifenn Donguy [CNRS]

Visiting Scientist

Erwan Bousse [Université étrangère, from Sep 2017]

External Collaborators

Gurvan Le Guernic [DGA]
Benoit Combemale [Univ Toulouse, from Sep 2017, HDR]

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) [101]. 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 [72]. 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 [54]. 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 [104].

[1] See http://www.sleconf.org/

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. [78] 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 [86].

3.1.2. Variability modeling

While the basic vision underlying *Software Product Lines* (SPL) can probably be traced back to David Parnas seminal article [94] 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 [92]. 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* [91]. Bosch provides a different definition [60]: *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 [66]. 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 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 [74]. 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 [61]. Variability management is thus growingly seen as the cornerstone of SPL development, covering the entire development life cycle, from requirements elicitation [106] to product derivation [111] to product testing [90], [89].

Halmans *et al.* [74] 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.* [96]: 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.* [84] (feature, hardware platform, performances and attributes variability) or Bass *et al.* [52] 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* [80]. 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 [61]. 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 [56], [68]. Model Driven Engineering (MDE) makes it possible to automate this weaving process [79]. 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 [87]) manipulating these models to produce for instance a detailed design that can ultimately be transformed to code, or to test suites [95], or other software artifacts.

3.1.3. Component-based software development

Component-based software development [105] 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 [67]: description of requirements on services required from the other components; indirect connections between components thanks to ports and connectors constructs [82]; hierarchical definition of components (assemblies of components can define new component types); connectors supporting various communication semantics [64]; quantitative properties on the services [59].

In recent years component-based architectures have evolved from static designs to dynamic, adaptive designs (e.g., SOFA [64], Palladio [57], Frascati [88]). 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 [81]. 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 [65], [108]. 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 [58], 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 [99]". 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 [76] 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 [73]. Multi Objectives Search based techniques [70] 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 [107] mainly relies on model analysis, constraint solving [69] and search-based reasoning [83]. DIVERSE leverages in particular the applications of model-based testing in the context of highly-configurable systems and [109] interactive systems [85] as well as recent advances based on diversity for test cases selection [77].

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 [93], 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 [63] 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 [53] provides modeling elements amenable to semantically equivalent mathematical models. Metropolis offers a precise semantics flexible enough to support different models of computation. ModHel'X [75] 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 [51], recovery blocks [97] and code randomization [55], 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 [102], [100]. Such methods have been used for example to understand the impact of new software development paradigms [62]. 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 [50].

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

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 [101], [72]. 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.

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

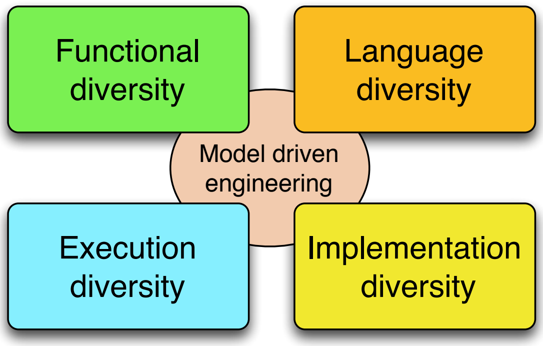

Figure 1. The four research axis of DIVERSE, which rely on a MDE scientific background

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 [103] (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 [97] and security [71] 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" [98], [110].

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

5.1.1. Awards

- Publications
  - Learning-Contextual Variability Models, IEEE Software
  - Correctness Attraction: A Study of Stability of Software Behavior Under Runtime Perturbation, Empirical Software Engineering

  item Great positions for members (KTH, Univ Toulouse, Mc Gill, . . .)
- Three new direct collaborations with Industrial partners: Orange, Nokia, Safran
- Great visibility for AmIUnique and several popularization actions
- Kermeta transfer to Obeo

6. New Software and Platforms

6.1. 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/

6.2. 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: Aymeric Hervieu, Benoit Baudry, Didier Vojtisek, Edward Mauricio Alferez Salinas, Guillaume Bécan, Joao Bosco Ferreira-Filho, Julien Richard-Foy, Mathieu Acher, Olivier Barais and Sana Ben Nasr
- Contact: Mathieu Acher
- URL: http://familiar-project.github.com

6.3. GEMOC Studio

KEYWORDS: DSL - Language workbench - Model debugging
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: Didier Vojtisek, Dorian Leroy, Erwan Bousse, Fabien Coulon and Julien Deantoni
- Partners: IRIT - ENSTA - I3S - OBEO - Thales TRT
- Contact: Benoît Combemale
- URL: http://gemoc.org/studio.html

6.4. Kevoree

Kevoree Core
KEYWORDS: M2M - Dynamic components - Iot - Heterogeneity - Smart home - Cloud - Software architecture - 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: Aymeric Hervieu, Benoit Baudry, Francisco-Javier Acosta Padilla, Inti Gonzalez Herrera, Ivan Paez Anaya, Jacky Bourgeois, Jean Emile Dartois, Johann Bourcier, Manuel Leduc, Maxime Tricoire, Mohamed Boussaa, Noël Plouzeau and Olivier Barais
- Contact: Olivier Barais
- URL: http://kevoree.org/

6.5. Melange

KEYWORDS: Modeling language - Meta-modelisation - Language workbench - Dedicated langage - Model-driven software engineering - DSL - MDE - Meta model - Model-driven engineering - 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: Arnaud Blouin, Benoît Combemale, David Mendez Acuna, Didier Vojtisek, Dorian Leroy, Erwan Bousse, Fabien Coulon, Jean-Marc Jézéquel, Olivier Barais and Thomas Degueule
- Contact: Benoît Combemale
- URL: http://melange-lang.org

6.6. Opencompare

KEYWORD: Software Product Line

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

7. New Results

7.1. Results on Variability modeling and management

7.1.1. Variability and testing.

Many approaches for testing configurable software systems start from the same assumption: it is impossible to test all configurations. This motivated the definition of variability-aware abstractions and sampling techniques to cope with large configuration spaces. Yet, there is no theoretical barrier that prevents the exhaustive testing of all configurations by simply enumerating them, if the effort required to do so remains acceptable. Not only this: we believe there is lots to be learned by systematically and exhaustively testing a configurable system. We report on the first ever endeavor to test all possible configurations of an industry-strength, open source configurable software system, JHipster, a popular code generator for web applications. We built a testing scaffold for the 26,000+ configurations of JHipster using a cluster of 80 machines during 4 nights for a total of 4,376 hours (182 days) CPU time. We find that 35.70% configurations fail and we identify the feature interactions that cause the errors. We show that sampling testing strategies (like dissimilarity and 2-wise) (1) are more effective to find faults than the 12 default configurations used in the JHipster continuous integration; (2) can be too costly and exceed the available testing budget. We cross this quantitative analysis with the qualitative assessment of JHipster's lead developers. Additional resources: preliminary effort on JHipster [32], https://arxiv.org/abs/1710.07980 https://github.com/axel-halin/Thesis-JHipster/

7.1.2. Variability and teaching.

Software Product Line (SPL) engineering has emerged to provide the means to efficiently model, produce, and maintain multiple similar software variants, exploiting their common properties, and managing their variabilities (differences). With over two decades of existence, the community of SPL researchers and practitioners is thriving as can be attested by the extensive research output and the numerous successful industrial projects. Education has a key role to support the next generation of practitioners to build highly complex, variability-intensive systems. Yet, it is unclear how the concepts of variability and SPLs are taught, what are the possible missing gaps and difficulties faced, what are the benefits, or what is the material available. Also, it remains unclear whether scholars teach what is actually needed by industry. We report on three initiatives we have conducted with scholars, educators, industry practitioners, and students to further understand the connection between SPLs and education, i.e., an online survey on teaching SPLs we performed with 35 scholars, another survey on learning SPLs we conducted with 25 students, as well as two workshops held at the International Software Product Line Conference in 2014 and 2015 with both researchers and industry practitioners participating. We build upon the two surveys and the workshops to derive recommendations for educators to continue improving the state of practice of teaching SPLs, aimed at both individual educators as well as the wider community. Finally, we are developing and maintaining a repository for teaching SPLs and variability. Additional resources: https://teaching.variability.io

7.1.3. *Variability and constraint solving.*

Array constraints are essential for handling data structures in automated reasoning and software verification. Unfortunately, the use of a typical finite domain (FD) solver based on local consistency-based filtering has strong limitations when constraints on indexes are combined with constraints on array elements and size. This work proposes an efficient and complete FD-solving technique for extended constraints over (possibly unbounded) arrays. We describe a simple but particularly powerful transformation for building an equisatisfiable formula that can be efficiently solved using standard FD reasoning over arrays, even in the unbounded case. Experiments show that the proposed solver significantly outperforms FD solvers, and successfully competes with the best SMT-solvers [38]. This work is not directly related to variability and SPL. But it contributes to DiverSE's attempts to connect artificial intelligence techniques to software variability engineering, in which constraint solving or machine learning are typically applied.

7.1.4. *Variability and machine learning (performance specialization of variability-intensive systems).*

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. Specifically, based on an oracle (e.g. a runtime test) that tells us whether a given configuration meets the requirements (e.g. speed or memory footprint), we leverage machine learning to retrofit the acquired knowledge into a variability model of the system that can be used to automatically specialize the configurable system. We validate our approach on a set of well-known configurable software systems (Apache server, x264, etc.) Our results show that, for many different kinds of objectives and performance qualities, the approach has interesting accuracy, precision and recall after a learning stage based on a relative small number of random samples [43]. Additional resources: https://learningconstraints.github.io and VaryVary ANR project

7.1.5. *Variability and machine learning (learning contextual variability models).*

Modeling how contextual factors relate to a software system's configuration space is usually a manual, error-prone task that depends highly on expert knowledge. Machine-learning techniques can automatically predict the acceptable software configurations for a given context. Such an approach executes and observes a sample of software configurations within a sample of contexts. It then learns what factors of each context will likely discard or activate some of the software's features. This lets developers and product managers automatically extract the rules that specialize highly configurable systems for specific contexts [27] Additional resources: https://learningconstraints.github.io and VaryVary ANR project

We are currently exploring the use of machine learning for variability-intensive systems in the context of VaryVary ANR project (see also VaryLaTeX [28]).

7.2. Results on Software Language Engineering

7.2.1. *On Language Interfaces*

Complex systems are developed by teams of experts from multiple domains , who can be liberated from becoming programming experts through domain-specific languages (DSLs). The implementation of the different concerns of DSLs (including syntaxes and semantics) is now well-established and supported by various languages workbenches. However, the various services associated to a DSL (e.g., editors, model checker, debugger or composition operators) are still directly based on its implementation. Moreover, while most of the services crosscut the different DSL concerns, they only require specific information on each. Consequently, this prevents the reuse of services among related DSLs, and increases the complexity of service implementation. Leveraging the time-honored concept of interface in software engineering, we discuss in [40] the benefits of language interfaces in the context of software language engineering. In particular, we elaborate on particular usages that address current challenges in language development.

7.2.2. Revisiting Visitors for Modular Extension of Executable DSMLs

Executable Domain-Specific Modeling Languages (xDSMLs) are typically defined by metamodels that specify their abstract syntax, and model interpreters or compilers that define their execution semantics. To face the proliferation of xDSMLs in many domains, it is important to provide language engineering facilities for opportunistic reuse, extension, and customization of existing xDSMLs to ease the definition of new ones. Current approaches to language reuse either require to anticipate reuse, make use of advanced features that are not widely available in programming languages, or are not directly applicable to metamodel-based xDSMLs. In [35], we propose a new language implementation pattern, named REVISITOR, that enables independent extensibility of the syntax and semantics of metamodel-based xDSMLs with incremental compilation and without anticipation. We seamlessly implement our approach alongside the compilation chain of the Eclipse Modeling Framework, thereby demonstrating that it is directly and broadly applicable in various modeling environments. We show how it can be employed to incrementally extend both the syntax and semantics of the fUML language without requiring anticipation or re-compilation of existing code, and with acceptable performance penalty compared to classical handmade visitors.

7.2.3. Advanced and efficient execution trace management for executable domain-specific modeling languages

Executable Domain-Specific Modeling Languages (xDSMLs) enable the application of early dynamic verification and validation (V&V) techniques for behavioral models. At the core of such techniques, execution traces are used to represent the evolution of models during their execution. In order to construct execution traces for any xDSML, generic trace metamodels can be used. Yet, regarding trace manipulations, generic trace metamodels lack efficiency in time because of their sequential structure, efficiency in memory because they capture superfluous data, and usability because of their conceptual gap with the considered xDSML. We contributed in [22] a novel generative approach that defines a multidimensional and domain-specific trace metamodel enabling the construction and manipulation of execution traces for models conforming to a given xDSML. Efficiency in time is improved by providing a variety of navigation paths within traces, while usability and memory are improved by narrowing the scope of trace metamodels to fit the considered xDSML. We evaluated our approach by generating a trace metamodel for fUML and using it for semantic differencing, which is an important V&V technique in the realm of model evolution. Results show a significant performance improvement and simplification of the semantic differencing rules as compared to the usage of a generic trace metamodel.

7.2.4. Omniscient Debugging for Executable DSLs

Omniscient debugging is a promising technique that relies on execution traces to enable free traversal of the states reached by a model (or program) during an execution. While a few General-Purpose Languages (GPLs) already have support for omniscient debugging, developing such a complex tool for any executable Domain Specific Language (DSL) remains a challenging and error prone task. A generic solution must: support a wide range of executable DSLs independently of the metaprogramming approaches used for implementing their semantics; be efficient for good responsiveness. Our contribution in [21] relies on a generic omniscient debugger supported by efficient generic trace management facilities. To support a wide range of executable DSLs, the debugger provides a common set of debugging facilities, and is based on a pattern to define runtime services independently of metaprogramming approaches. Results show that our debugger can be used with various executable DSLs implemented with different metaprogramming approaches. As compared to a solution that copies the model at each step, it is on average six times more efficient in memory, and at least 2.2 faster when exploring past execution states, while only slowing down the execution 1.6 times on average.

7.2.5. Reverse Engineering Language Product Lines from Existing DSL Variants

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 [26], we propose a reverse-engineering technique to ease-off such a development scenario. Our approach receives a set of DSL variants which are used to automatically recover a language modular design and to synthesize the corresponding variability models. The validation is performed in a project involving industrial partners that required three different variants of a DSL for finite state machines. This validation shows that our approach is able to correctly identify commonalities and variability.

7.2.6. Software Language Engineering for Virtual Reality Software Development

Due to the nature of Virtual Reality (VR) research, conducting experiments in order to validate the researcher's hypotheses is a must. However, the development of such experiments is a tedious and time-consuming task. In [48], we propose to make this task easier, more intuitive and faster with a method able to describe and generate the most tedious components of VR experiments. The main objective is to let experiment designers focus on their core tasks: designing , conducting, and reporting experiments. To that end, we applied well-established SLE concepts promoted in DIVERSE to the VR domain to ease the development of VR experiments. More precisely, we propose the use of DSLs to ease the description and generation of VR experiments. An analysis of published VR experiments is used to identify the main properties that characterize VR experiments. This allowed us to design AGENT (Automatic Generation of ExperimeNtal proTocol runtime), a DSL for specifying and generating experimental protocol runtimes. We demonstrated the feasibility of our approach by using AGENT on two experiments published in the VRST'16 proceedings.

7.2.7. Create and Play your Pac-Man Game with the GEMOC Studio

Executable Domain-Specific Languages (DSLs) are used for defining the behaviors of systems. In particular, the operational semantics of such DSLs may define how conforming models react to stimuli from their environment. This commonly requires adapting the semantics to define both the possible domain-level stimuli, and their handling during the execution. However, manually adapting the semantics for such cross-cutting concern is a complex and error-prone task. In , we demonstrate a tool addressing this problem by allowing the augmentation of operational semantics for handling stimuli, and by automatically generating a complete behavioral language interface from this augmentation. At runtime, this interface can receive stimuli sent to models, and can safely handle them by automatically interrupting the execution flow. This tool is an extension to the GEMOC Studio, a language and modeling workbench for executable DSLs We demonstrate how it can be used to implement a Pac-Man DSL enabling to create and play Pac-Man games.

7.3. Results on Heterogeneous and dynamic software architectures

We have selected three main contributions for DIVERSE's research axis #4: one is in the field of runtime management, while the two others one are in the field of Privacy and Security.

7.3.1. Verifying the configuration of Virtualized Network Functions in Software Defined Networks

In Kevoree, one of the goal is to work on the shipping pases in which we aim at making deployment, and the reconfiguration simple and accessible to the whole team. This year we work to include the capacity to manage network configuration when reconfiguring application stack. In this context, the deployment of modular virtual network functions (VNFs) in software defined infrastructures (SDI) enables cloud and network providers to deploy integrated network services across different resource domains. It leads to a large interleaving between network configuration through software defined network controllers and VNF deployment within this network. Most of the configuration management tools and network orchestrator used to deploy VNF lack of an abstraction to express Assume-Guarantee contracts between the VNF and the SDN configuration. Consequently, VNF deployment can be inconsistent with network configurations.

Contribution. To tackle this challenge, in this work [41], we develop an approach to check the consistency between the VNF description described from a set of structural models and flow-chart models and a proposed deployment on a real SDN infrastructure with its own configuration manager. We illustrate our approach on virtualized Evolved Packet Core function.

Originality. The originality of this work is to propose a model to capture VNF.

Impact. Beyond the scientific originality of this work, the main impacts of this novel approach to check SDN configuration 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 (B-com PhD thesis and Nokia common labs). This work is being integrated within the Kevoree platform.

7.3.2. Identity Negotiation at Runtime

Authentication delegation is a major function of the modern web. Identity Providers (IdP) acquired a central role by providing this function to other web services. By knowing which web services or web applications access its service, an IdP can violate the end-user privacy by discovering information that the user did not want to share with its IdP. For instance, WebRTC introduces a new field of usage as authentication delegation happens during the call session establishment, between two users. As a result, an IdP can easily discover that Bob has a meeting with Alice. A second issue that increases the privacy violation is the lack of choice for the end-user to select its own IdP. Indeed, on many web-applications, the end-user can only select between a subset of IdPs, in most cases Facebook or Google.

Contribution. This year, we analyze this phenomena [23], in particular why the end-user cannot easily select its preferred IdP, though there exists standards in this field such as OpenID Connect and OAuth 2? To lead this analysis, we conduct three investigations. The first one is a field survey on OAuth 2 and OpenID Connect scope usage by web sites to understand if scopes requested by web-sites could allow for user defined IdPs. The second one tries to understand whether the problem comes from the OAuth 2 protocol or its implementations by IdP. The last one tries to understand if trust relations between websites and IdP could prevent the end user to select its own IdP. Finally, we sketch possible architecture for web browser based identity management, and report on the implementation of a prototype. We also describe our implementation of the WebRTC identity architecture [24]. We adapt OpenID Connect servers to support WebRTC peer to peer authentication and detail the issues and solutions found in the process.

Originality. We observe that although WebRTC allows for the exchange of identity assertion between peers, users lack feedback and control over the other party authentication. To allow identity negotiation during a WebRTC communication setup, we propose an extension to the Session Description Protocol. Our implementation demonstrates current limitations with respect to the current WebRTC specification.

Impact. This work is done with Orange.

7.3.3. Raising Time Awareness in Model-Driven Engineering

The conviction that big data analytics is a key for the success of modern businesses is growing deeper, and the mobilisation of companies into adopting it becomes increasingly important. Big data integration projects enable companies to capture their relevant data, to efficiently store it, turn it into domain knowledge, and finally monetize it. In this context, historical data, also called temporal data, is becoming increasingly available and delivers means to analyse the history of applications, discover temporal patterns, and predict future trends. Despite the fact that most data that today's applications are dealing with is inherently temporal current approaches, methodologies, and environments for developing these applications don't provide sufficient support for handling time. We envision that Model-Driven Engineering (MDE) would be an appropriate ecosystem for a seamless and orthogonal integration of time into domain modeling and processing.

Contribution. This year, we investigate the state-of-the-art in MDE techniques and tools in order to identify the missing bricks for raising time-awareness in MDE and outline research directions in this emerging domain [30].

Originality. We propose an extended context representation for self-adaptive software that integrates the history of planned actions as well as their expected effects over time into the context representations. We demonstrate on a cloud elasticity manager case study that such *temporal action-aware context* leads to improved reasoners while still be highly scalable. This work is original with respect to the state of the art since it provides a way to represent and take into account the impact of reconfiguration actions on a system.

Impact. This work is done through a collaboration with the SnT in Luxembourg and a startup called DataThings, working on domain model representation for various industrial domains.

7.3.4. 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. Tellu relies on Kevoree and KevoreeJS to build their health management systems. They will be also an 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.

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, 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

Our work on software diversification explores various ways of adding randomness in program executions: state perturbations that preserve functional correctness [25]; randomizing of web APIs to mitigate browser fingerprinting [33].

Can the execution of software be perturbed without breaking the correctness of the output? In this work [25], we devise a protocol to answer this question from a novel perspective. In an experimental study, we observe that many perturbations do not break the correctness in ten subject programs. We call this phenomenon "correctness attraction". The uniqueness of this protocol is that it considers a systematic exploration of the perturbation space as well as perfect oracles to determine the correctness of the output. To this extent, our findings on the stability of software under execution perturbations have a level of validity that has never been reported before in the scarce related work. A qualitative manual analysis enables us to set up the first taxonomy ever of the reasons behind correctness attraction.

The rich programming interfaces (APIs) provided by web browsers can be diverted to collect a browser fingerprint. A small number of queries on these interfaces are sufficient to build a fingerprint that is statistically unique and very stable over time. Consequently, the fingerprint can be used to track users. Our work [33] aims at mitigating the risk of browser fingerprinting for users privacy by 'breaking' the stability of a fingerprint over time. We add randomness in the computation of selected browser functions, in order to have them deliver slightly different answers for each browsing session. Randomization is possible thanks to the following properties of browsers implementations: (i) some functions have a nondeterministic specification, but a deterministic implementation ; (ii) multimedia functions can be slightly altered without deteriorating user's perception. We present FPRandom, a modified version of Firefox that adds randomness to mitigate the most recent fingerprinting algorithms, namely canvas fingerprinting, AudioContext fingerprinting and the unmasking of browsers through the order of JavaScript properties. We evaluate the effectiveness of FPRandom by testing it against known fingerprinting tests. We also conduct a user study and evaluate the performance overhead of randomization to determine the impact on the user experience.

The other aspect in the area of software diversity is about the statistical analysis of browser fingerprinting on a large industrial dataset [17], [31].

7.4.2. Software testing

Generative software development has paved the way for the creation of multiple code generators and compilers that serve as a basis for automatically generating code to a broad range of software and hardware platforms. With full automatic code generation, the user is able to easily and rapidly synthetize software artifacts for various software platforms. In addition, modern generators (i.e., C compilers) become highly configurable, offering numerous configuration options that the user can use to easily customize the generated code for the target hardware platform. In this context, it is crucial to verify the correct behaviour of code generators. Numerous approaches have been proposed to verify the functional outcome of generated code but few of them evaluate the non-functional properties of automatically generated code, namely the performance and resource usage properties. The thesis of Mohamed Boussaa [16] has addressed this limitation.

8. Bilateral Contracts and Grants with Industry

8.1. Bilateral Contracts with Industry

8.1.1. GLOSE

- Partners: Inria/CNRS/Safran
- Dates: 2017-2021
- Abstract: The GLOSE project develops new techniques for heterogeneous modeling and simulation in the context of systems engineering. It aims to provide formal and operational tools and methods to formalize the behavioral semantics of the various modeling languages used at system-level. These semantics will be used to extract behavioral language interfaces supporting the definition of coordination patterns. These patterns, in turn, can systematically be used to drive the coordination of any model conforming to these languages. The project is structured according to the following tasks: concurrent xDSML engineering, coordination of discrete models, and coordination of discrete/continuous models. The project is funded in the context of the network DESIR, and supported by the GEMOC initiative.

8.1.2. One Shot Software (OSS)

- Partners: Inria/Orange
- Dates: 2017-2019
- Abstract: The OSS project investigates an extreme version of moving target defense where a slightly different version of the application is deployed each time it is used (e.g., for crypto functions or payment services). We investigate the analysis, synthesis and transformation techniques to support diversification at 5 points of a software construction pipeline, which, once combined yield up to billions of variants. We also evaluate the support of diversification as a first class property in DevOps.

9. Partnerships and Cooperations

9.1. National Initiatives

9.1.1. ANR

9.1.1.1. 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.

9.1.1.2. VaryVary ANR JCJC

- Coordinator: Mathieu Acher
- DiverSE, Inria/IRISA Rennes
- Dates: 2017-2021
- Abstract: Most modern software systems (operating systems like Linux, Web browsers like Firefox or Chrome, video encoders like x264 or ffmpeg, servers, mobile applications, etc.) are subject to variation or come in many variants. Hundreds of configuration options, features, or plugins can be combined, each potentially with distinct functionality and effects on execution time, memory footprint, etc. Among configurations, some of them are chosen and do not compile, crash at runtime, do not pass a test suite, or do not reach a certain performance quality (e.g., energy consumption, security). In this JCJC ANR project, we follow a thought-provocative and unexplored direction: We consider that the variability boundary of a software system can be specialized and should vary when needs be. The goal of this project is to provide theories, methods and techniques to make vary variability. Specifically, we consider machine learning and software engineering techniques for narrowing the space of possible configurations to a good approximation of those satisfying the needs of users. Based on an oracle (e.g., a runtime test) that tells us whether a given configuration meets the requirements (e.g. speed or memory footprint), we leverage machine learning to retrofit the acquired constraints into a variability that can be used to automatically specialize the configurable system. Based on a relative small number of configuration samples, we expect to reach high accuracy for many different kinds of oracles and subject systems. Our preliminary experiments suggest that varying variability can be practically useful and effective. However, much more work is needed to investigate sampling, testing, and learning techniques within a variety of cases and application scenarios. We plan to further collect large experimental data and apply our techniques on popular, open-source, configurable software (like Linux, Firefox, ffmpeg, VLC, Apache or JHipster) and generators for media content (like videos, models for 3D printing, or technical papers written in LaTeX).

9.1.1.3. 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).

9.1.1.4. 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.

9.1.2. DGA

9.1.2.1. FPML (CYBERDEFENSE)

- Coordinator: DGA
- Partners: DGA MI, Inria
- Dates: 2014-2017
- 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.

9.1.3. Cominlabs

9.1.3.1. 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.

9.2. European Initiatives

9.2.1. FP7 & H2020 Projects

9.2.1.1. FP7 STREP HEADS

- Coordinator: SINTEF
- Other partners: Inria, Software AG, ATC, Tellu, eZmonitoring
- Dates: 2013-2017

- 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.

9.2.1.2. H2020 ICT-10-2016 STAMP

- Coordinator: Inria Rennes
- 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.

9.2.2. 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.
- KTH, the Royal Institute of Technology (Sweden): continuous software testing, perturbation and diversification.

9.3. International Initiatives

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

9.3.1.1. ALE

- Title: Agile Language Engineering
- International Partner (Institution - Laboratory - Researcher):
  - CWI (Netherlands)
- Start year: 2017
- See also: http://gemoc.org/ale/

- Software engineering faces new challenges with the advent of modern software-intensive systems such as complex critical embedded systems, cyber-physical systems and the Internet of things. Application domains range from robotics, transportation systems, defense to home automation, smart cities, and energy management, among others. Software is more and more pervasive, integrated into large and distributed systems, and dynamically adaptable in response to a complex and open environment. As a major consequence, the engineering of such systems involves multiple stakeholders, each with some form of domain-specific knowledge, and with an increasingly use of software as an integration layer. Hence more and more organizations are adopting Domain Specific Languages (DSLs) to allow domain experts to express solutions directly in terms of relevant domain concepts. This new trend raises new challenges about designing DSLs, evolving a set of DSLs and coordinating the use of multiple DSLs for both DSL designers and DSL users. ALE will contribute to the field of Software Language Engineering, aiming to provide more agility to both language designers and language users. The main objective is twofold. First, we aim to help language designers to leverage previous DSL implementation efforts by reusing and combining existing language modules. Second, we aim to provide more flexibility to language users by ensuring interoperability between different DSLs and offering live feedback about how the model or program behaves while it is being edited (aka. live programming/modeling).

9.3.2. Inria International Partners

9.3.2.1. Informal International Partners

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

9.3.3. Participation in Other International Programs

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

9.3.4. 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.

9.4. International Research Visitors

9.4.1. Visits of International Scientists

Yves Le Traon, Professor at the University of Luxembourg, visited the team in June and July 2017.

Tanja Mayerhofer, Junior Researcher at the TU Wien, visited the team in March 2017.

François Fouquet, Junior Researcher at the SnT (Lux), visited the team in November 2017.

9.4.1.1. Internships

Koko armando Nguepi kenfack, Master interships at the University of Namur, visited the team from September 2017 to January 2018.

9.4.2. Visits to International Teams

Manuel Leduc visited CWI for 3 weeks in September 2017

Benoit Combemale visited Professor Jorg Kienzle at McGill University (Canada) for 3 months in 2017; and made several short visits at CWI (The Netherlands).

9.4.2.1. Research Stays Abroad

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

10. Dissemination

10.1. Promoting Scientific Activities

10.1.1. Scientific Events Organisation

10.1.1.1. General Chair, Scientific Chair

- Benoit Combemale has been general chair for SLE 2017, the major international conference in the area of software language engineering.
- Benoit Combemale has been main organizer of the Dagstuhl Seminar #17342 on “The Software Language Engineering Body of Knowledge” (SLEBoK).
- Mathieu Acher has been program committee co-chair of SPLC 2017, the major international conference in the area of variability and software product line engineering.

10.1.1.2. Member of the Organizing Committees

Arnaud Blouin:

- Publicity co-chair, EICS’17, 2017

Mathieu Acher:

- Publicity chair, ICSR’18
- Co-organizer of REVE’17 (international workshop on reverse engineering variability)

10.1.2. Scientific Events Selection

10.1.2.1. Member of the Conference Program Committees

Mathieu Acher:

- PC member ICSR 2017
- PC member VaMoS 2017
- PC member SAC 2017
- PC member SEAA 2017
- PC member GramSec’17

Olivier Barais:

- PC member SEAA 2017
- PC member SAC 2017
- PC member ICWE 2017

Benoit Baudry:

- PC member ICSE 2017
- PC member ASE 2017

Arnaud Blouin:

- PC member of the ACM Student Research Competition (SRC) at MODELS 2017

Benoit Combemale:

- PC member for MODELS'17
- PC member for ICMT'17
- PC member for the GEMOC'17 workshop at MODELS'17
- PC member for the MDEbug'17 workshop at MODELS'17
- PC member for the EXE'17 workshop at MODELS'17
- PC member for the MiSE'17 workshop at ICSE'17
- PC member for the MoMo'17 workshop at Modularity'17

Jean-Marc Jézéquel:

- PC member ICSE 2017
- PC member SPLC 2017 Vision track
- PC member SEAMS 2017

Johann Bourcier:

- PC member for the SEsCPS'17 workshop at ICSE'17

10.1.2.2. Reviewer

Arnaud Blouin was an external reviewer for ICSE 2017, EICS 2017, MODELS 2017, ICSA 2017, SEAMS 2017. Olivier Barais was a reviewer for ICSE 2017, Sosym 2017, JSS, JISA.

Johann Bourcier was a reviewer for ASE 2017, ICSE 2017 and Models 2017.

10.1.3. Journal

10.1.3.1. 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.

Jean-Marc Jézéquel is Associate Editor in Chief of the Journal of Software and Systems Modeling: SoSyM, and member of the Editorial Board of:

- IEEE Computer
- Journal of Systems and Software: JSS
- Journal of Object Technology: JOT

10.1.3.2. Reviewer - Reviewing Activities

Arnaud Blouin: JSS, Ingénierie des Systèmes d'Information

Johann Bourcier was reviewer for the following magazine : IEEE Communication Magazine.

10.1.4. Invited Talks

Benoit Baudry:

- Software technology and DevOps. Scientific days of Orange
- Reconciling Diversity and Privacy at the Dagstuhl Seminar on Online Privacy and Web Transparency
- Software diversification as an obfuscation technique, at the International Workshop on Obfuscation: Science, Technology, and Theory

Benoit Combemale:

- From Model (driven) Engineering, to Language (driven) Engineering. Invited talk at CEA DAM, France.
- Towards Language-Oriented Modeling. Invited talk at UQAM (Montreal, Canada).
- Sound, yet Flexible, Modeling: A Language Engineering Point Of View. Keynote at FlexMDE'17.
- Modeling For Sustainability – Or How to Make Smart CPS Smarter? Invited talk at CWI (Amsterdam, The Netherlands).
- Model Simulation, Graphical Animation, and Omniscient Debugging with Sirius Animator. Invited talk at SiriusCon'17.

10.1.5. 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.

Arnaud Blouin:

- Founding member and member of the GL-IHM (software engineering and human-computer interaction) working group (action spécifique GDR-GPL 2017)

10.1.6. Scientific Expertise

Arnaud Blouin: external ANR (French national research agency) reviewer 2017 Olivier Barais: International collaborations foreign minister. Expert Olivier Barais: AutoActive proposal board of expert (Norvegian project) Olivier Barais: AutoActive proposal board of expert (Norvegian project)

Johann Bourcier : scientific reviewer for CIR (Credit Impot Recherche).

10.1.7. Research Administration

Benoit Baudry is in the scientific advisory board of the SVV lab, University of Luxembourg.

Jean-Marc Jézéquel is the Director of IRISA, a 800+ people joint labs of CNRS, ENS Rennes, INSA Rennes, Inria, Telecom Bretagne, CentraleSupelec, the University of Rennes 1 and the University of South Brittany. He is also the head of Research of the French CyberSecurity Excellence Cluster, as well as the EIT Digital Rennes Satellite Node Director.

10.2. Teaching - Supervision - Juries

10.2.1. 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, Mathieu Acher and Benoit Combemale teaches about 200h in these domains, with Benoit Baudry and teaching about 50h, for a grand total of about 1500 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 at the board of the ESIR engineering school.

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

10.2.2. Supervision

- PhD : Mohamed Boussaa, *Automatic non-functional testing and tuning of configurable generators*, 06/09/2017, B. Baudry, O. Barais
- PhD : Marcelino Rodriguez Cancio, *Automatic computation diversification*, 19/12/2017, B. Baudry and B. Combemale
- PhD: Pierre Laperdrix, *Browser Fingerprinting: Exploring Device Diversity to Augment Authentication and Build Client-Side Countermeasures*, 03/11/17, B. Baudry, G. Avoine
- 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: Johan Pelay, *Langage pour une programmation incrémentale de réseau*, 2016, O. Barais, F. Guillemin
- PhD in progress: Quentin Plazar, *Bridging the gap between SAT and SNP*, 2015, M. Acher, A. Goetib
- PhD in progress: Paul Temple, *Leraning variability models*, 2015, M. Acher, J.-M. Jézéquel
- PhD in progress: Manuel Leduc, *Formal and Executable Specification of domain-specific language families*, 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, *Performance resources modeling for container based applications*, 2016, O. Barais
- PhD in progress: Oscar Luis, *Automatic test amplification*, 2016, B. Baudry
- PhD in progress: Alexandre Rio, *Activity modeling for better renewable energy usage*, 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

- PhD: in progress, Ludovic Mouline, *A modeling Framework for context representation*, 2015, J. Bourcier, O. Barais, F. Fouquet, Y. Le Traon
- PhD: in progress, Erwan Picard, *Legal aspect of online profiling*, 2015, M.Boizard, B. Baudry

10.2.3. Juries

10.2.3.1. Benoit Baudry

was in the examination committee of the following PhD thesis:

- Pierre Laperdrix, September 2017, Univ Rennes I, Supervisor
- Marcelino Rodriguez-Cancio, December 2017, Univ Rennes I, Supervisor
- Xavier Devroey, June 2017, Univ Namur, Referee

10.2.3.2. Olivier Barais

was in the examination committee of the following HDR:

- Mohammed Boussa, 2017, Univ RennesI, Supervisor
- hamza ouarnoughi, 2017, Univ Brest, Reviewer
- Ternava Xhevahire, 2017, Univ Nice, Reviewer
- Colin AYGALINC, 2017, Univ Grenoble, Reviewer
- Fadwa Rekik, May 2017, Univ Paris Saclay, Reviewer

10.2.3.3. Benoit Combemale

was in the examination committee of the following PhD thesis:

- Ulyana Tikhonova, November 2017, TU/e, The Netherlands, Reviewer
- Cyril Cecchinel, November 2017, Univ. Nice, Examiner
- Rodriguez Cancio, December 2017, Univ Rennes 1, Supervisor

10.3. Popularization

IT industry forums . Pierre Laperdrix presented his work on the mitigation of browser fingerprinting at the Mozilla Festival [3]. The results of STAMP [4] have been presented at EclipseCon [5], the Paris Open Source Summit [6], OW2'con [7] and the cloud computing world expo [8].

Science festivals . Jean-Marc Jézéquel and Benoit Baudry have presented overviews of research issues in Cybersecurity to general public audiences in several events, such as the 2017 Pint of Science [9] and the science en bobine festival [10].

11. 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^{o} 1, pp. 16:1–16:26, https://hal.inria.fr/hal-01182103

[3] https://mozillafestival.org/
[4] https://www.stamp-project.eu
[5] https://www.eclipsecon.org/europe2017/
[6] http://www.opensourcesummit.paris/
[7] https://www.ow2con.org
[8] http://www.cloudcomputing-world.com/
[9] https://pintofscience.fr/
[10] https://sciencesenbobines.org/rennes-35/

[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^{o} 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, M. ALFÉREZ, 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] 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, L. MOULINE, 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] M. BOUSSAA. *Automatic non-functional testing and tuning of configurable generators*, Université Rennes 1, September 2017, https://tel.archives-ouvertes.fr/tel-01598821

[17] P. LAPERDRIX. *Browser Fingerprinting: Exploring Device Diversity to Augment Authentication and Build Client-Side Countermeasures*, INSA Rennes, October 2017, https://hal.inria.fr/tel-01621257

Articles in International Peer-Reviewed Journals

[18] M. ACHER, R. E. LOPEZ-HERREJON, R. RABISER. *Teaching Software Product Lines: A Snapshot of Current Practices and Challenges*, in "ACM Transactions of Computing Education", May 2017, https://hal.inria.fr/hal-01522779

[19] M. ALFÉREZ, M. ACHER, J. A. GALINDO, B. BAUDRY, D. BENAVIDES. *Modeling Variability in the Video Domain: Language and Experience Report*, in "Software Quality Journal", January 2018, pp. 1-28, https://hal.inria.fr/hal-01688247

[20] 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

[21] E. BOUSSE, D. LEROY, B. COMBEMALE, M. WIMMER, B. BAUDRY. *Omniscient Debugging for Executable DSLs*, in "Journal of Systems and Software", November 2017, vol. 137, pp. 261-288 [*DOI :* 10.1016/J.JSS.2017.11.025], https://hal.inria.fr/hal-01662336

[22] E. BOUSSE, T. MAYERHOFER, B. COMBEMALE, B. BAUDRY. *Advanced and efficient execution trace management for executable domain-specific modeling languages*, in "Software and Systems Modeling", May 2017, pp. 1–37 [*DOI :* 10.1007/S10270-017-0598-5], https://hal.inria.fr/hal-01614377

[23] K. CORRE, O. BARAIS, G. SUNYÉ, V. FREY, J.-M. CROM. *Why can't users choose their identity providers on the web?*, in "Proceedings on Privacy Enhancing Technologies", January 2017, vol. 2017, n^{o} 3, pp. 72-86 [*DOI :* 10.1515/POPETS-2017-0029], https://hal.archives-ouvertes.fr/hal-01611048

[24] K. CORRE, S. BÉCOT, O. BARAIS, G. SUNYÉ. *A WebRTC Extension to Allow Identity Negotiation at Runtime*, in "Lecture Note in Computer Science", 2017, vol. 10360, pp. 412-419 [*DOI :* 10.1007/978-3-319-60131-1_27], https://hal.archives-ouvertes.fr/hal-01611057

[25] B. DANGLOT, P. PREUX, B. BAUDRY, M. MONPERRUS. *Correctness Attraction: A Study of Stability of Software Behavior Under Runtime Perturbation*, in "Empirical Software Engineering", 2017, https://arxiv.org/abs/1611.09187 [*DOI :* 10.1007/S10664-017-9571-8], https://hal.archives-ouvertes.fr/hal-01378523

[26] D. A. MÉNDEZ-ACUÑA, J. A. GALINDO, B. COMBEMALE, A. BLOUIN, B. BAUDRY. *Reverse Engineering Language Product Lines from Existing DSL Variants*, in "Journal of Systems and Software", May 2017 [*DOI :* 10.1016/J.JSS.2017.05.042], https://hal.inria.fr/hal-01524632

[27] P. TEMPLE, M. ACHER, J.-M. JEZEQUEL, O. BARAIS. *Learning-Contextual Variability Models*, in "IEEE Software", November 2017, vol. 34, n^{o} 6, pp. 64-70 [*DOI :* 10.1109/MS.2017.4121211], https://hal.inria.fr/hal-01659137

International Conferences with Proceedings

[28] M. ACHER, P. TEMPLE, J.-M. JEZEQUEL, J. Á. GALINDO DUARTE, J. MARTINEZ, T. ZIADI. *VaryLaTeX: Learning Paper Variants That Meet Constraints*, in "VaMoS 2018 - 12th International Workshop on Variability Modelling of Software-Intensive Systems", Madrid, Spain, February 2018, pp. 1-6, https://hal.inria.fr/hal-01659161

[29] O. BARAIS, B. COMBEMALE, A. WORTMANN. *Language Engineering with the GEMOC Studio*, in "ICSAW 2017 - IEEE International Conference on Software Architecture Workshops", Gothenburg, Sweden, IEEE (editor), April 2017, 3 p. , https://hal.inria.fr/hal-01609576

[30] A. BENELALLAM, T. HARTMANN, L. MOULINE, F. FOUQUET, J. BOURCIER, O. BARAIS, Y. LE TRAON. *Raising Time Awareness in Model-Driven Engineering*, in "ACM/IEEE 20th International Conference on Model Driven Engineering Languages and Systems", Austin, Texas, United States, September 2017, https://hal.archives-ouvertes.fr/hal-01580554

[31] A. GÓMEZ-BOIX, P. LAPERDRIX, B. BAUDRY. *Fingerprinting mobile devices: A short analysis*, in "CIEL 2017 - 6ème Conférence en IngénieriE du Logiciel", Montpellier, France, June 2017, https://hal.inria.fr/hal-01611101

[32] A. HALIN, A. NUTTINCK, M. ACHER, X. DEVROEY, G. PERROUIN, P. HEYMANS. *Yo Variability! JHipster: A Playground for Web-Apps Analyses*, in "11th International Workshop on Variability Modelling of Software-intensive Systems", Eindhoven, Netherlands, February 2017, pp. 44 - 51 [*DOI :* 10.1145/3023956.3023963], https://hal.inria.fr/hal-01468084

[33] P. LAPERDRIX, B. BAUDRY, V. MISHRA. *FPRandom: Randomizing core browser objects to break advanced device fingerprinting techniques*, in "ESSoS 2017 - 9th International Symposium on Engineering Secure Software and Systems", Bonn, Germany, July 2017, 17 p. , https://hal.inria.fr/hal-01527580

[34] G. LE MOULEC, F. ARGELAGUET, V. GOURANTON, A. BLOUIN, B. ARNALDI. *AGENT: Automatic Generation of Experimental Protocol Runtime*, in "ACM Symposium on Virtual Reality Software and Technology (VRST)", Gothenburg, Sweden, Virtual Reality Software and Technology, November 2017, https://hal.archives-ouvertes.fr/hal-01613873

[35] M. LEDUC, T. DEGUEULE, B. COMBEMALE, T. VAN DER STORM, O. BARAIS. *Revisiting Visitors for Modular Extension of Executable DSMLs*, in "ACM/IEEE 20th International Conference on Model Driven Engineering Languages and Systems", Austin, United States, September 2017, https://hal.inria.fr/hal-01568169

[36] D. LEROY, E. BOUSSE, M. WIMMER, B. COMBEMALE, W. SCHWINGER. *Create and Play your Pac-Man Game with the GEMOC Studio (Tool Demonstration)*, in "EXE 2017 - 3rd International Workshop on Executable Modeling", Austin, United States, September 2017, pp. 1-6, https://hal.inria.fr/hal-01651801

[37] Y. NDIAYE, N. AILLERY, O. BARAIS, A. BLOUIN, A. BOUABDALLAH. *Modélisation et Évaluation de la Sécurité des IHM*, in "CIEL 2017 : 6ème Conférence en IngénieriE du Logiciel", Montpellier, France, June 2017, https://hal.inria.fr/hal-01611324

[38] Q. PLAZAR, M. ACHER, S. BARDIN, A. GOTLIEB. *Efficient and Complete FD-Solving for Extended Array Constraints* *, in "IJCAI 2017", Melbourne, Australia, August 2017, https://hal.archives-ouvertes.fr/hal-01545557

[39] A. VASTEL, P. LAPERDRIX, W. RUDAMETKIN, R. ROUVOY. *FP-STALKER: Tracking Browser Fingerprint Evolutions*, in "IEEE S&P 2018 - 39th IEEE Symposium on Security and Privacy", San Francisco, United States, B. PARNO, C. KRUEGEL (editors), Proceedings of the 39th IEEE Symposium on Security and Privacy (S&P), IEEE, May 2018, pp. 1-14, https://hal.inria.fr/hal-01652021

Scientific Books (or Scientific Book chapters)

[40] 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

Books or Proceedings Editing

[41] J. PELAY, F. GUILLEMIN, O. BARAIS (editors). *Verifying the configuration of Virtualized Network Functions in Software Defined Networks*, b<>com, 2017, pp. 1-6, Accepted version of the paper but with two formulas which had to be deleted in the final version for size reasons, forthcoming, https://hal.archives-ouvertes.fr/hal-01657866

[42] S. VAN MIERLO, E. BOUSSE, H. VANGHELUWE, M. WIMMER, C. VERBRUGGE, M. GOGOLLA, M. TICHY, A. BLOUIN (editors). *Report on the 1 st International Workshop on Debugging in Model-Driven Engineering (MDEbug'17)*, December 2017, pp. 1-6, https://hal.inria.fr/hal-01665572

Research Reports

[43] P. TEMPLE, M. ACHER, J.-M. JÉZÉQUEL, L. NOEL-BARON, J. A. GALINDO. *Learning-Based Performance Specialization of Configurable Systems*, IRISA, Inria Rennes ; University of Rennes 1, February 2017, https://hal.archives-ouvertes.fr/hal-01467299

[44] A. WORTMANN, B. COMBEMALE, O. BARAIS. *A Systematic Mapping Study on Modeling for Industry 4.0*, Inria Rennes - Bretagne Atlantique and University of Rennes 1, France, April 2017, n^{o} RR-9062, 25 p. , https://hal.inria.fr/hal-01514421

Scientific Popularization

[45] J.-M. JÉZÉQUEL, J. JONGWANE. *Comment maîtriser la complexité des logiciels ?*, in "Interstices", February 2017, https://hal.inria.fr/hal-01503821

[46] L. MOULINE, T. HARTMANN, F. FOUQUET, Y. LE TRAON, J. BOURCIER, O. BARAIS. *Weaving Rules into Models@run.time for Embedded Smart Systems*, in "Programming 2017 - Companion to the first International Conference on the Art, Science and Engineering of Programming", Brussels, Belgium, April 2017, pp. 1 - 6 [*DOI :* 10.1145/3079368.3079394], https://hal.inria.fr/hal-01609796

Other Publications

[47] A. BLOUIN, V. LELLI, B. BAUDRY, F. COULON. *User Interface Design Smell: Automatic Detection and Refactoring of Blob Listeners*, April 2017, working paper or preprint, https://hal.inria.fr/hal-01499106

[48] G. LE MOULEC, A. BLOUIN, V. GOURANTON, B. ARNALDI. *Automatic Production of End User Documentation for DSLs*, December 2017, working paper or preprint, https://hal.inria.fr/hal-01549042

[49] M. MONPERRUS, B. DANGLOT, O. VERA-PEREZ, Z. YU, B. BAUDRY. *The Emerging Field of Test Amplification: A Survey*, November 2017, https://arxiv.org/abs/1705.10692 - working paper or preprint, https://hal.archives-ouvertes.fr/hal-01634288

References in notes

[50] 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

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

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

[53] 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^o 4, pp. 45–52

[54] 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

[55] 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^o 1, pp. 3–40

[56] 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

[57] 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^o 1, pp. 3–22

[58] 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

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

[60] 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

[61] 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

[62] 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

[63] 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

[64] 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

[65] 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

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

[67] 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

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

[69] 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

[70] 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

[71] 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

[72] 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

[73] 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

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

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

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

[77] 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

[78] 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

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

[80] 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

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

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

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

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

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

[86] 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

[87] 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

[88] 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.

[89] 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

[90] 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

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

[92] 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

[93] 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

[94] 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

[95] 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

[96] 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

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

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

[99] 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

[100] 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

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

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

[103] 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

[104] 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

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

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

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

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

[109] 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

[110] 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

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