Activity Report 2018

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
A1.3.1. - Web
A1.3.6. - Fog, Edge
A2.1.3. - 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
A2.5.5. - Software testing
A2.6.2. - Middleware
A2.6.4. - Ressource management
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
B9.5.1. - Computer science
B9.10. - Privacy

1. Team, Visitors, External Collaborators

Faculty Members

Olivier Barais [Team leader, Univ de Rennes I, Professor, HDR]
Mathieu Acher [Univ de Rennes I, Associate Professor]
Arnaud Blouin [INSA Rennes, Associate Professor]
Johann Bourcier [Univ de Rennes I, Associate Professor, HDR]
Benoit Combemale [Univ de Toulouse 2 Jean Jaurès, Professor, Inria secondment, HDR]
Jean-Marc Jezequel [Univ de Rennes I, Professor, HDR]
Noel Plouzeau [Univ de Rennes I, Associate Professor]

Post-Doctoral Fellow

Juliana Alves Pereira [Univ de Rennes I, from Sep 2018]

PhD Students

Akbar Pranata Alif [Inria, from Oct 2018]
June Benvegnu Sallou [Univ de Rennes I, from Oct 2018]
Antoine Cheron [FaberNoval, from Mar 2018]
Fabien Coulon [Obeo, from Sep 2018]
Jean-Emile Dartois [Institut de recherche technologique B-com]
Alejandro Gomez Boix [Inria]
Pierre Jeanjean [Inria, from Nov 2018]
Romain Lebouc [Univ de Rennes I, from Oct 2018]
Manuel Leduc [Univ de Rennes I]
Dorian Leroy [TU Wien, Austria]
Gauthier Lyan [Keolis, from Feb 2018]
Hugo Martin [Univ de Rennes I, from Sep 2018]
Ludovic Mouline [SnT, Luxembourg]
Youssou Ndiaye [Orange Labs]
Johan Pelay [Institut de recherche technologique B-com, until Sep 2018]
Quentin Plazar [Inria, until Sep 2018]
Paul Temple [Univ de Rennes I, until Nov 2018]
Oscar Luis Vera Perez [Inria]

Technical staff

Amine Benelallam [Univ de Rennes I]
Maxime Bricet [Univ de Rennes I]
Caroline Landry [Inria]
Maxime Tricoire [Inria, until Jun 2018]
Didier Vojtisek [Inria]

Interns

Max Aguirre [Univ de Rennes I, from Jun 2018 until Jul 2018]
Gwendal Didot [Univ de Rennes I, from May 2018 until Aug 2018]
Arnaud Gohier [Univ de Rennes I, from Apr 2018 until Aug 2018]
Alexis Lemasle [Univ de Rennes I, from May 2018 until Aug 2018]
Hugo Martin [Univ de Rennes I, from Feb 2018 until Aug 2018]
Enzo Menegaldo [Univ de Rennes I, from Jun 2018 until Sep 2018]
Yannick Namour [Univ de Rennes I, from Apr 2018 until Aug 2018]
Koko Armando Nguepi Kenfack [Univ de Rennes I, until Jan 2018]
Anthony Orain [Univ de Rennes I, from Jun 2018 until Jul 2018]

Administrative Assistant

Tifenn Donguy [CNRS]

Visiting Scientists

Erwan Bousse [TU Wien, Austria, until Aug 2018]
Marcel Heinz [University of Koblenz-Landau, Jul 2018]
Benoit Combemale [Université de Toulouse, until Aug 2018, HDR]

External Collaborators

Gurvan Le Guernic [DGA]
Benoit Baudry [KTH, HDR]

2. Overall Objectives

2.1. Overall objectives

DIVERSE's research agenda targets core values of software engineering. In this fundamental domain we focus and develop models, methodologies and theories to address major challenges raised by the emergence of several forms of diversity in the design, deployment and evolution of software-intensive systems. Software diversity has emerged as an essential phenomenon in all application domains born by our industrial partners. These application domains range from complex systems brought by systems of systems (addressed in collaboration with Thales, Safran, CEA and DGA) and Instrumentation and Control (addressed with EDF) to pervasive combinations of Internet of Things and Internet of Services (addressed with TellU and Orange) and tactical information systems (addressed in collaboration with civil security). Today these systems seem to be radically different, but we envision a strong convergence of the scientific principles that underpin their construction and validation, bringing forwards sane and reliable methods for the design of **flexible and open yet dependable systems**. Flexibility and openness are critical and challenging software layer properties that must deal with four dimensions of diversity: **diversity of languages**, used by the stakeholders involved in the construction of these systems; **diversity of features**, required by the different customers; **diversity of runtime environments**, in which software has to run and adapt; **diversity of implementations**, which are necessary for resilience by redundancy.

In this context, the central 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 yet valid 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 summarized below:

Scientific objective for DIVERSE: To 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 this aspect stands as 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) [99]. 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 [69]. 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 [51]. 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*, including a unified typing theory that integrate models as first class entities [102].

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. [75] 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 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 [83].

3.1.2. Variability modeling

While the basic vision underlying *Software Product Lines* (SPL) can probably be traced back to David Parnas seminal article [92] 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 [90]. 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* [89]. Bosch provides a different definition [57]: *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 [63]. 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 [71]. 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 [58]. Variability management is thus growingly seen as the cornerstone of SPL development, covering the entire development life cycle, from requirements elicitation [104] to product derivation [109] to product testing [87], [86].

Halmans *et al.* [71] 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.* [94]: 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.* [81] (feature, hardware platform, performances and attributes variability) or Bass *et al.* [49] who discusses 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* [77]. 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 [58]. 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 [53], [65]. Model Driven Engineering (MDE) makes it possible to automate this weaving process [76]. 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 [84]) manipulating these models to produce for instance a detailed design that can ultimately be transformed to code, or to test suites [93], or other software artifacts.

3.1.3. Component-based software development

Component-based software development [103] 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 [64]: description of requirements on services required from the other components; indirect connections between components thanks to ports and connectors constructs [79]; hierarchical definition of components (assemblies of components can define new component types); connectors supporting various communication semantics [61]; quantitative properties on the services [56].

In recent years component-based architectures have evolved from static designs to dynamic, adaptive designs (e.g., SOFA [61], Palladio [54], Frascati [85]). 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 a stop/redeploy/start process). Dynamic adaptation 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 [78]. 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 [62], [106]. 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 [55], 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 [97]". 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 [73] 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 [70]. Multi Objectives Search based techniques [67] 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 [105] mainly relies on model analysis, constraint solving [66] and search-based reasoning [80]. DIVERSE leverages in particular the applications of model-based testing in the context of highly-configurable systems and [107] interactive systems [82] as well as recent advances based on diversity for test cases selection [74].

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 [91], or Xholon [1]. 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 [60] 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 [50] provides modeling elements amenable to semantically equivalent mathematical models. Metropolis offers a precise semantics flexible enough to support different models of computation. ModHel'X [72] 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 [48], recovery blocks [95] and code randomization [52], 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 [100], [98]. Such methods have been used for example to understand the impact of new software development paradigms [59]. 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 [47].

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 axes 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

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

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

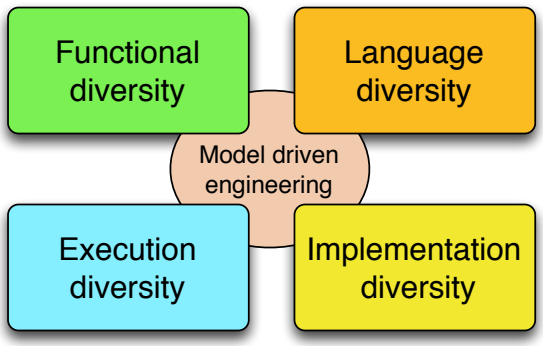

Figure 1. The four research axes of DIVERSE, which rely on a MDE scientific background

3.2.1. Software Language Engineering

The engineering of systems involves many different stakeholders, each with their own domain of expertise. Hence more and more organizations are adopting Domain Specific Modeling Languages (DSMLs) to allow domain experts to express solutions directly in terms of relevant domain concepts [99], [69]. This new trend raises new challenges about designing DSMLs, evolving a set of DSMLs and coordinating the use of multiple DSLs for both DSL designers and DSL users.

3.2.1.1. Challenges

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

Variability in modeling languages occur in the definition of the abstract and concrete syntax as well as in the specification of the language's semantics. The major challenges met when addressing the need for variability are: (i) to set principles for modeling language units that support the modular specification of a modeling language; and (ii) to design mechanisms to assemble these units into 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) gives 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 support to 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 to MoCs and DSMLs design and implementation and to 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 [101] (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 the 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 approaches to deal with the increasing number of concerns in complex systems: 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 reasoning techniques to **automatically analyze** variability models and their interactions with other views on the software intensive system (requirements, architecture, design, code). 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 and learning-based techniques for the exploration of the space of interaction between variability and view models.

We want 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.

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.

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, the number of testing environments to test a software patch or upgrade 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 [95] and security [68] 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" [96], [108].

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. Highlights of the Year

4.1. Highlights of the Year

This year, we would like to highlight the following results:

- In terms of publications:
  - Among the many articles published this year, articles [25] and b [28] have been published at the highest level but above all they represent perfectly the type of research conducted within the team: open research based on studies of major open-source software and in connection with the developer communities.

  - The results of this year's SLE conference also make us very proud. 4 accepted papers including 1 best vision paper [33], 1 best artifact (hal-01890446) and the award for the best reviewer for a former doctoral student of the team recently appointed associate professor at the University of Nantes.

- A former PhD student of the team, Pierre Laperdrix was awarded the "*Le prix de thèse Gilles Kahn 2018 (premier accessit), décerné par la SiF et patronné par l'Académie des Sciences*" for his PhD entitled *Browser Fingerprinting: Exploring Device Diversity to Augment Authentication and Build Client-Side Countermeasures*.
- Three new PhDs and one new HDR have been succesfully defended this year.
- A new CNRS junior researcher, Djamel Eddine Khelladi, will join the team in 2019.
- Mathieu Acher succesfully submitted its ERC starting grant program: Killing and Resurrecting Software Variability (REVARY). This research program fully structures the Variability axis of the team for the next years.
- Didier Vojtisek, research engineer hosted since many years within the team was awarded the Inria award (*appui à la recherche*) with Guillaume Cassonnet, Christophe Demarey, Herve Mathieu, Florent Pruvost for the Sonarqube project. As a research team in the field of software engineering, we study and produce many open source software artefacts. In this context, we regularly test and deploy internally support services to produce high-quality software. Sonar (SonarQube ancestor) had been deployed internally since 2008. Embedding research engineers into software engineering research teams as often as possible is undoubtedly a win-win operation for both parties (the research team but also the SED and therefore Inria as a whole)

4.1.1. Awards

Paper was awarded the best vision paper at SLE'18.
Paper was awarded the best artefact associated to a scientific paper at SLE'18.

Paper was awarded the best paper at ICMT'18.

BEST PAPERS AWARDS:

[33]
F. COULON, T. DEGUEULE, T. VAN DER STORM, B. COMBEMALE. *Shape-Diverse DSLs: Languages without Borders (Vision Paper)*, in "SLE 2018 - 11th ACM SGIPLAN International Conference on Software Language Engineering", Boston, United States, ACM, November 2018, pp. 215-219 [*DOI :* 10.1145/3276604.3276623], https://hal.archives-ouvertes.fr/hal-01889155

[36]
M. LEDUC, T. DEGUEULE, B. COMBEMALE. *Modular Language Composition for the Masses*, in "SLE 2018 - 11th ACM SIGPLAN International Conference on Software Language Engineering", Boston, United States, November 2018, pp. 1-12 [*DOI :* 10.1145/3276604.3276622], https://hal.inria.fr/hal-01890446

[32]
J.-M. BRUEL, B. COMBEMALE, E. GUERRA, J.-M. JÉZÉQUEL, J. KIENZLE, J. DE LARA, G. MUSSBACHER, E. SYRIANI, H. VANGHELUWE. *Model Transformation Reuse across Metamodels - A classification and comparison of approaches*, in "ICMT 2018 - International Conference on Theory and Practice of Model Transformations", Toulouse, France, LNCS, Springer, June 2018, vol. 10888, pp. 92-109 [*DOI :* 10.1007/978-3-319-93317-7_4], https://hal.inria.fr/hal-01910113

5. New Software and Platforms

5.1. 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/

5.2. 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

5.3. 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

5.4. 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/

5.5. Melange

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

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

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

- Participants: 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

5.6. 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

5.7. DSpot

KEYWORDS: Software testing - Test amplification
FUNCTIONAL DESCRIPTION: DSpot is a tool that generates missing assertions in JUnit tests. DSpot takes as input a Java project with an existing test suite. As output, DSpot outputs new test cases on console. DSpot supports Java projects built with Maven and Gradle

- Participants: Benoit Baudry, Martin Monperrus and Benjamin Danglot
- Partner: KTH Royal Institute of Technology
- Contact: Benjamin Danglot
- URL: https://github.com/STAMP-project/dspot

5.8. ALE

Action Language for Ecore
KEYWORDS: Meta-modeling - Executable DSML
FUNCTIONAL DESCRIPTION: Main features of ALE include:

- Executable metamodeling: Re-open existing EClasses to insert new methods with their implementations - Metamodel extension: The very same mechanism can be used to extend existing Ecore metamodels and insert new features (eg. attributes) in a non-intrusive way - Interpreted: No need to deploy Eclipse plugins, just run the behavior on a model directly in your modeling environment - Extensible: If ALE doesn't fit your needs, register Java classes as services and invoke them inside your implementations of EOperations.

- Partner: OBEO
- Contact: Benoît Combemale
- URL: http://gemoc.org/ale-lang/

5.9. InspectorGuidget

KEYWORDS: Static analysis - Software testing - User Interfaces
FUNCTIONAL DESCRIPTION: InspectorGuidget is a static code analysing tool. InspectorGuidget analyses UI (user interface/interaction) code of a software system to extract high level information and metrics. InspectorGuidget also finds bad UI coding pratices, such as Blob listener instances. InspectorGuidget analyses Java code.

- Participants: Arnaud Blouin and Benoit Baudry
- Contact: Arnaud Blouin
- Publications: User Interface Design Smell: Automatic Detection and Refactoring of Blob Listeners - Automatic Detection of GUI Design Smells: The Case of Blob Listener
- URL: https://github.com/diverse-project/InspectorGuidget

6. New Results

6.1. Results on Variability modeling and management

6.1.1. 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. Publication at Empirical Software Engineering: [25] See also, in the rest of the report, the work on *Multimorphic Testing* that actually relies on variability techniques.

6.1.2. 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. Publication at SPLC (journal first) [29], workshop SPLTea'18 http://spltea.irisa.fr/ and repository: https://teaching.variability.io

6.1.3. Variability and machine learning

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 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. Publications: Temple et al. *Towards Adversarial Configurations for Software Product Lines* https://arxiv.org/abs/1805.12021, VaryLaTeX [30] a variability and learning-based tool to generate relevant paper variants written in latex.

TUXML (Tux is the mascotte of the Linux Kernel while ML stands for statistical machine learning). The goal of TuxML is to predict properties of Linux Kernel configurations (e.g., does the kernel compile? what's its size? does it boot?). The Linux Kernel provides near 15000 configuration options: there is an infinity of different kernels. As we cannot compile, measure, and observe all combinations of options (aka configurations), we're trying to learn Linux kernel properties out of a sample of configurations. The TuxML project is developing tools, mainly based on Docker and Python, to massively compile and gather data about thousand of configuration kernels https://github.com/TuxML/.

In general, we are currently exploring the use of machine learning for variability-intensive systems in the context of VaryVary ANR project https://varyvary.github.io.

6.2. Results on Software Language Engineering

6.2.1. 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.

6.2.2. Trace Comprehension Operators for Executable DSLs

Recent approaches contribute facilities to breathe life into metamodels, thus making behavioral models directly executable. Such facilities are particularly helpful to better utilize a model over the time dimension, e.g., for early validation and verification. However, when even a small change is made to the model, to the language definition (e.g., semantic variation points), or to the external stimuli of an execution scenario, it remains difficult for a designer to grasp the impact of such a change on the resulting execution trace. This prevents accessible trade-off analysis and design-space exploration on behavioral models. In [44], we propose a set of formally defined operators for analyzing execution traces. The operators include dynamic trace filtering, trace comparison with diff computation and visualization, and graph-based view extraction to analyze cycles. The operators are applied and validated on a demonstrative example that highlight their usefulness for the comprehension specific aspects of the underlying traces.

6.2.3. Model Transformation Reuse across Metamodels

Model transformations (MTs) are essential elements of model-driven engineering (MDE) solutions. MDE promotes the creation of domain-specific metamodels, but without proper reuse mechanisms, MTs need to be developed from scratch for each new metamodel. In [32], awarded by the **best paper award at ICMT 2018**, we classify reuse approaches for MTs across different metamodels and compare a sample of specific approaches – model types, concepts, a-posteriori typing, multilevel modeling, and design patterns for MTs – with the help of a feature model developed for this purpose, as well as a common example. We discuss strengths and weaknesses of each approach, provide a reading grid used to compare their features, and identify gaps in current reuse approaches.

6.2.4. Modular Language Composition for the Masses

The goal of modular language development is to enable the definition of new languages as assemblies of pre-existing ones. Recent approaches in this area are plentiful but usually suffer from two main problems: either they do not support modular language composition both at the specification and implementation levels, or they require advanced knowledge of specific paradigms which hampers wide adoption in the industry. In [36], awarded by the **best artefact award at SLE 2018**, we introduce a non-intrusive approach to modular development of language concerns with well-defined interfaces that can be composed modularly at the specification and implementation levels. We present an implementation of our approach atop the Eclipse Modeling Framework, namely Alex-an object-oriented metalanguage for semantics definition and language composition. We evaluate Alex in the development of a new DSL for IoT systems modeling resulting from the composition of three independently defined languages (UML activity diagrams, Lua, and the OMG Interface Description Language). We evaluate the effort required to implement and compose these languages using Alex with regards to similar approaches of the literature.

6.2.5. Shape-Diverse DSLs

Domain-Specific Languages (DSLs) manifest themselves in remarkably diverse shapes, ranging from internal DSLs embedded as a mere fluent API within a programming language, to external DSLs with dedicated syntax and tool support. Although different shapes have different pros and cons, combining them for a single language is problematic: language designers usually commit to a particular shape early in the design process, and it is hard to reconsider this choice later. In the new ideas paper [33] awarded as the **best new ideas paper at SLE 2018**, we envision a language engineering approach enabling (i) language users to manipulate language constructs in the most appropriate shape according to the task at hand, and (ii) language designers to combine the strengths of different technologies for a single DSL. We report on early experiments and lessons learned building Prism, our prototype approach to this problem. We illustrate its applicability in the engineering of a simple shape-diverse DSL implemented conjointly in Rascal, EMF, and Java. We hope that our initial contribution will raise the awareness of the community and encourage future research.

6.2.6. Fostering metamodels and grammars

Advanced and mature language workbenches have been proposed in the past decades to develop Domain-Specific Languages (DSL) and rich associated environments. They all come in various flavors, mostly depending on the underlying technological space (e.g., grammarware or modelware). However, when the time comes to start a new DSL project, it often comes with the choice of a unique technological space which later bounds the possible expected features. In [37], we introduce NabLab, a full-fledged industrial environment for scientific computing and High Performance Computing (HPC), involving several metamodels and grammars. Beyond the description of an industrial experience of the development and use of tool-supported DSLs, we report in this paper our lessons learned, and demonstrate the benefits from usefully combining metamodels and grammars in an integrated environment.

6.2.7. Automatic Production of End User Documentation for DSLs

The development of DSLs requires a significant software engineering effort: editors, code generators, etc., must be developed to make a DSL usable. Documenting a DSL is also a major and time-consuming task

required to promote it and address its learning curve. Recent research work in software language engineering focus on easing the development of DSLs. This work focuses on easing the production of documentation of textual DSLs [27], [17]. The API documentation domain identified challenges we adapted to DSL documentation. Based on these challenges we propose a model-driven approach that relies on DSL artifacts to extract information required to build documentation. Our implementation, called Docywood, targets two platforms: Markdown documentation for static web sites and Xtext code fragments for live documentation while modeling. We used Docywood on two DSLs, namely ThingML and Target Platform Definition. Feedback from end users and language designers exhibits qualitative benefits of the proposal with regard to the DSL documentation challenges. End user experiments conducted on ThingML and Target Platform Definition show benefits on the correctness of the created models when using Docywood on ThingML.

6.3. 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 of resources for dynamically adaptive system, one in the field of temporal context model for dynamically adaptive system and a last one to improve the exploration of hidden real-time structures of programming behavior at runtime.

6.3.1. Resource-aware models@runtime layer for dynamically adaptive system

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 a whole development team. This year, we mainly explore two main axes.

In the first one, we try to improve the proposed models that could be used at runtime to improve resource usage in two domains: cloud computing and energy [34]. In the cloud computing domain, we try to improve resources usage in providing models to cloud provider to allow the reselling of unused resources to peers. Indeed, although Cloud computing techniques have reduced the total cost of ownership thanks to virtualization, the average usage of resources (e.g., CPU, RAM, Network, I/O) remains low. To address such issue, one may sell unused resources. Such a solution requires the Cloud provider to determine the resources available and estimate their future use to provide availability guarantees. In this work, we propose a technique that uses machine learning algorithms (Random Forest, Gradient Boosting Decision Tree, and Long Short Term Memory) to forecast 24-hour of available resources at the host level. Our technique relies on the use of quantile regression to provide a flexible trade-off between the potential amount of resources to reclaim and the risk of SLA violations. In addition, several metrics (e.g., CPU, RAM, disk, network) were predicted to provide exhaustive availability guarantees. Our methodology was evaluated by relying on four in production data center traces and our results show that quantile regression is relevant to reclaim unused resources. Our approach may increase the amount of savings up to 20% compared to traditional approaches.
In the energy domain, we work at proposing models that could be used at runtime to improve self-consumption of renewable energies [46]. Self-consumption of renewable energies is defined as electricity that is produced from renewable energy sources, not injected to the distribution or transmission grid or instantaneously withdrawn from the grid and consumed by the owner of the power production unit or by associates directly contracted to the producer. Designing solutions in favor of self-consumption for small industries or city districts is challenging. It consists in designing an energy production system made of solar panels, wind turbines, batteries that fit the annual weather prediction and the industrial or human activity. In this context, this we highlight the essentials of a domain specific modeling language designed to let domain experts run their own simulations.

6.3.2. A Temporal Model for Interactive Diagnosis of Adaptive Systems

The evolving complexity of adaptive systems impairs our ability to deliver anomaly-free solutions. Fixing these systems require a deep understanding on the reasons behind decisions which led to faulty or suboptimal system states. Developers thus need diagnosis support that trace system states to the previous circumstances targeted requirements, input context that had resulted in these decisions. However, the lack of efficient temporal representation limits the tracing ability of current approaches. To tackle this problem, we describe

a novel temporal data model to represent, store and query decisions as well as their relationship with the knowledge (context, requirements, and actions) [38]. We validate our approach through a use case based-on the smart grid at Luxembourg.

Based on this work, we also enable a models@runtime approach in which we integrate the time required for a reconfiguration action to achieve the expected impact [39]. Indeed in most of the MAPE-K loop system, unfinished actions as well as their expected effects over time are not taken into consideration in MAPE-K loop processes, leading upcoming analysis phases potentially take sub-optimal actions. In this work, we propose an extended context model for MAPE-K loop that integrates the history of planned actions as well as their expected effects over time into the context representations. This information can then be used during the upcoming analysis and planning phases to compare measured and expected context metrics. We demonstrate on a cloud elasticity manager case study that such temporal action-aware context leads to improved reasoners while still be highly scalable.

6.3.3. Detection and analysis of behavioral T-patterns in debugging activities

A growing body of research in empirical software engineering applies recurrent patterns analysis in order to make sense of the developers' behavior during their interactions with IDEs. However, the exploration of hidden real-time structures of programming behavior remains a challenging task. In this work [40], we investigate the presence of temporal behavioral patterns (T-patterns) in debugging activities using the THEME software. Our preliminary exploratory results show that debugging activities are strongly correlated with code editing, file handling, window interactions and other general types of programming activities. The validation of our T-patterns detection approach demonstrates that debugging activities are performed on the basis of repetitive and well-organized behavioral events. Furthermore, we identify a large set of T-patterns that associate debugging activities with build success, which corroborates the positive impact of debugging practices on software development.

6.4. 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) observations of software diversity mainly through browser fingerprinting, and (ii) software testing to study and assess the validity of software.

6.4.1. Privacy and Security

6.4.1.1. FP-STALKER: Tracking Browser Fingerprint Evolutions

Browser fingerprinting has emerged as a technique to track users without their consent. Unlike cookies, fingerprinting is a stateless technique that does not store any information on devices, but instead exploits unique combinations of attributes handed over freely by browsers. The uniqueness of fingerprints allows them to be used for identification. However, browser fingerprints change over time and the effectiveness of tracking users over longer durations has not been properly addressed. In this work [42], we show that browser fingerprints tend to change frequently—from every few hours to days—due to, for example, software updates or configuration changes. Yet, despite these frequent changes, we show that browser fingerprints can still be linked, thus enabling long-term tracking. FP-STALKER is an approach to link browser fingerprint evolutions. It compares fingerprints to determine if they originate from the same browser. We created two variants of FP-STALKER, a rule-based variant that is faster, and a hybrid variant that exploits machine learning to boost accuracy. To evaluate FP-STALKER, we conduct an empirical study using 98,598 fingerprints we collected from 1,905 distinct browser instances. We compare our algorithm with the state of the art and show that, on average, we can track browsers for 54.48 days, and 26% of browsers can be tracked for more than 100 days.

6.4.1.2. Hiding in the Crowd: an Analysis of the Effectiveness of Browser Fingerprinting at Large Scale

Browser fingerprinting is a stateless technique, which consists in collecting a wide range of data about a device through browser APIs. Past studies have demonstrated that modern devices present so much diversity that fingerprints can be exploited to identify and track users online. With this work [35], we want to evaluate if browser fingerprinting is still effective at uniquely identifying a large group of users when analyzing millions of fingerprints over a few months.We analyze 2,067,942 browser fingerprints collected from one of the top 15 French websites. The observations made on this novel dataset shed a newlight on the ever-growing browser fingerprinting domain. The key insight is that the percentage of unique fingerprints in this dataset is much lower than what was reported in the past: only 33.6% of fingerprints are unique by opposition to over 80% in previous studies. We show that non-unique fingerprints tend to be fragile. If some features of the fingerprint change, it is very probable that the fingerprint will become unique. We also confirm that the current evolution of web technologies is benefiting users' privacy significantly as the removal of plugins brings down substantively the rate of unique desktop machines.

6.4.1.3. User Controlled Trust and Security Level of Web Real-Time Communications

In this work [16], we propose three main contributions. In our first contribution we study the WebRTC identity architecture and more particularly its integration with existing authentication delegation protocols. This integration has not been studied yet. To fill this gap, we implement components of the WebRTC identity architecture and comment on the issues encountered in the process. We then study this specification from a privacy perspective an identify new privacy considerations related to the central position of identity provider. In our second contribution, we aim at giving more control to users. To this end, we extend the WebRTC specification to allow identity parameters negotiation. We then propose a web API allowing users to choose their identity provider in order to authenticate on a third-party website. We validate our propositions by presenting prototype implementations. Finally, in our third contribution, we propose a trust and security model of a WebRTC session to help non-expert users to better understand the security of their WebRTC session. Our proposed model integrates in a single metric the security parameters used in the session establishment, the encryption parameters for the media streams, and trust in actors of the communication setup as defined by the user. We conduct a preliminary study on the comprehension of our model to validate our approach.

6.4.2. Software Testing

6.4.2.1. A Comprehensive Study of Pseudo-tested Methods

Pseudo-tested methods are defined as follows: they are covered by the test suite, yet no test case fails when the method body is removed, i.e., when all the effects of this method are suppressed. This intriguing concept was coined in 2016, by Niedermayr and colleagues [88], who showed that such methods are systematically present, even in well-tested projects with high statement coverage. This work presents a novel analysis of pseudo-tested methods [28]. First, we run a replication of Niedermayr's study with 28K+ methods, enhancing its external validity thanks to the use of new tools and new study subjects. Second, we perform a systematic characterization of these methods, both quantitatively and qualitatively with an extensive manual analysis of 101 pseudo-tested methods. The first part of the study confirms Niedermayr's results: pseudo-tested methods exist in all our subjects. Our in-depth characterization of pseudo-tested methods leads to two key insights: pseudo-tested methods are significantly less tested than the other methods; yet, for most of them, the developers would not pay the testing price to fix this situation. This calls for future work on targeted test generation to specify those pseudo-tested methods without spending developer time.

This work uses Descartes is a tool that implements extreme mutation operators and aims at finding pseudo-tested methods in Java projects [43]. It leverages the efficient transformation and runtime features of PITest.

6.4.2.2. Automatic Test Improvement with DSpot: a Study with Ten Mature Open-Source Projects

In the literature, there is a rather clear segregation between manually written tests by developers and automatically generated ones. In this work [23], we explore a third solution: to automatically improve existing test cases written by developers. We present the concept, design and implementation of a system called DSpot, that takes developer-written test cases as input (JUnit tests in Java) and synthesizes improved versions of them

as output. Those test improvements are given back to developers as patches or pull requests, that can be directly integrated in the main branch of the test code base. We have evaluated DSpot in a deep, systematic manner over 40 real-world unit test classes from 10 notable and open-source software projects. We have amplified all test methods from those 40 unit test classes. In 26/40 cases, DSpot is able to automatically improve the test under study, by triggering new behaviors and adding new valuable assertions. Next, for ten projects under consideration, we have proposed a test improvement automatically synthesized by DSpot to the lead developers. In total, 13/19 proposed test improvements were accepted by the developers and merged into the main code base. This shows that DSpot is capable of automatically improving unit-tests in real-world, large scale Java software.

6.4.2.3. Multimorphic Testing

The functional correctness of a software application is, of course, a prime concern, but other issues such as its execution time, precision , or energy consumption might also be important in some contexts. Systematically testing these quantitative properties is still extremely difficult, in particular, because there exists no method to tell the developer whether such a test set is "good enough" or even whether a test set is better than another one. This work [41] proposes a new method, called Multimorphic testing, to assess the relative effectiveness of a test suite for revealing performance variations of a software system. By analogy with mutation testing, our core idea is to vary software parameters, and to check whether it makes any difference on the outcome of the tests: i.e. are some tests able to " kill " bad morphs (configurations)? Our method can be used to evaluate the quality of a test suite with respect to a quantitative property of interest, such as execution time or computation accuracy.

6.4.2.4. User Interface Design Smell: Automatic Detection and Refactoring of Blob Listeners

User Interfaces (UIs) intensively rely on event-driven programming: widgets send UI events, which capture users' interactions, to dedicated objects called controllers. Controllers use several UI listeners that handle these events to produce UI commands. In this work [20], we reveal the presence of design smells in the code that describes and controls UIs. We then demonstrate that specific code analyses are necessary to analyze and refactor UI code, because of its coupling with the rest of the code. We conducted an empirical study on four large Java Swing and SWT open-source software systems: Eclipse, JabRef, ArgouML, and FreeCol. We study to what extent the number of UI commands that a UI listener can produce has an impact on the change- and fault-proneness of the UI listener code. We develop a static code analysis for detecting UI commands in the code. We identify a new type of design smell, called Blob listener that characterizes UI listeners that can produce more than two UI commands. We propose a systematic static code analysis procedure that searches for Blob listener that we implement in InspectorGuidget. We conducted experiments on the four software systems for which we manually identified 53 instances of Blob listener. The results exhibit a precision of 81.25 % and a recall of 98.11 %. We then developed a semi-automatically and behavior-preserving refactoring process to remove Blob listeners. 49.06 % of the Blob listeners were automatically refactored. Patches for JabRef, and FreeCol have been accepted and merged. Discussions with developers of the four software systems assess the relevance of the Blob listener.

7. Bilateral Contracts and Grants with Industry

7.1. Bilateral Contracts with Industry

7.1.1. ADR Nokia

- Coordinator: Inria
- Dates: 2017-2021
- Abstract: The goal of this project is to integrate chaos engineering principles to IoT Services frameworks to improve the robustness of the software-defined network services using this approach, to explore the concept of equivalence for software-defined network services, and to propose an approach to constantly alter the attack surface of the network services.

7.1.2. BCOM

- Coordinator: UR1
- Dates: 2018-2024
- Abstract: The purpose of the Falcon project is to investigate how to improve the resale of available resources in private clouds to third parties. In this context, the collaboration with DiverSE mainly aims to work on efficient techniques for the design of consumption models and resource consumption forecasting models. These models are then used as a knowledge base in a classical autonomous loop.

7.1.3. 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.

7.1.4. OneShotSoftware

- 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 five locations 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.

7.1.5. Agileo

- Partners: Inria/Agileo
- Dates: 2017-2018
- Abstract: In this project we mainly design a systematic mapping study on modeling for Industry 4.0 in order to share a common scientific roadmap.

7.1.6. Obeo

- Partners: Inria/Obo
- Dates: 2017-2020
- Abstract: Web engineering for domain-specific modeling languages, Fabien Coulon's PhD Cifre project.

7.1.7. OKWind

- Partners: UR1/OKWind
- Dates: 2017-2020
- Abstract: Models@runtime to improve self-consumption of renewable energies, Alexandre Rio's PhD Cifre project. .

7.1.8. Orange

- Partners: UR1/Orange
- Dates: 2016-2019
- Abstract: Security level modelling of user interface, Youssou Ndiaye's PhD Cifre project. .

7.1.9. Keolis

- Partners: UR1/Keolis
- Dates: 2018-2021
- Abstract: Urban mobility: machine learning for building simulators using large amounts of data, Gauthier LYAN's PhD Cifre project. .

7.1.10. FaberNovel

- Partners: UR1/FaberNovel
- Dates: 2018-2021
- Abstract: Abstractions for linked data and the programmable web, Antoine Cheron's PhD Cifre project. .

8. Partnerships and Cooperations

8.1. Regional Initiatives

8.1.1. PEC – Pôle d'Excellence Cyber

- Coordinator: Université de Rennes 1
- Dates: 2016-2019
- Abstract: Formal and Executable Specification of domain-specific language families

8.2. National Initiatives

8.2.1. ANR

8.2.1.1. SOPRANO

- Coordinator: CEA
- CEA, University of Paris-Sud, Inria Rennes, OcamlPro, Adacore
- Dates: 2014-2018
- Abstract: Today most major verification approaches rely on automatic external solvers. However these solvers do not fill the current and future needs for verification: lack of satisfying model generation, lack of reasoning on difficult theories (e.g. floating-point arithmetic), lack of extensibility for specific or new needs. The SOPRANO project aims at solving these problems and prepare the next generation of verification-oriented solvers by gathering experts from academia and industry. We will design a new framework for the cooperation of solvers, focused on model generation and borrowing principles from SMT (current standard) and CP (well-known in optimisation). These ideas will be implemented in an open-source platform, with regular evaluations from the industrial partners.

8.2.1.2. 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).

8.2.2. Competitivity Clusters

8.2.2.1. 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: 2014-2018
- Abstract: The Occiware project aims to establish a formal and equipped framework for the management of all cloud resource based on the OCCI standard.

8.2.3. Cominlabs

8.2.3.1. 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) 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 workings, and make them more accountable to users. PROFILE will also propose an innovative protection to mitigate browser fingerprinting, based on the collaborative reconfiguration of browsers.

8.3. European Initiatives

8.3.1. FP7 & H2020 Projects

8.3.1.1. 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 three academic partners with strong software testing expertise, five software companies (in: e-Health, Content Management, Smart Cities and Public Administration), and an open source consortium. This industry-near research addresses concrete, business-oriented objectives. All solutions are open source and developed as microservices to facilitate exploitation, with a target at TRL 6.

8.3.2. Collaborations in European Programs, Except FP7 & H2020

- Coordinator: UR1
- Other partners: Airbreizh - Surveillance of Brittany air quality association, branch of the French national air surveillance network, AmpliSIM - SME specialized in pollution numerical simulation, CNRS / IDRIS - Institute for Development and Resources in Intensive Scientific Computing, GENCI - Supercomputing centres association, Keolis - Public transport operator for Rennes Metropole, Neovia – SME specialized in HPC and project management, Rennes Métropole – Local authority in charge of the Service Public Métropolitain de la Donnée (SPMD), Ryax Technologies – SME providing Data Analytics workflows' automation and seamless orchestration on hybrid distributed infrastructures UCit – SME providing HPC as a Service
- Dates: 2018-2020
- Abstract: The AQMO project is co-financed by the European Union through its CEF programme. It addresses the air quality challenge, thanks to the development of a smart city pilot in the area of Rennes Metropole. The project will provide an end-to-end urban platform that extends current practices in air quality measurements. The AQMO platform will provide citizens, local authorities, scientific organizations and private companies with new data and innovative services based on computing simulation.
- Program: EIT Digital
- Project acronym: UAV-Retina
- Project title: Unmanned Aerial Vehicles - Retina
- Duration: from 2018-10 to 2019-12
- Coordinator: UR1
- Other partners: Bright Cape (company from the Netherlands), FBK (Trento University, Italy), JCP Connect (company from France), Tellus Environment (company from France), Fire Department of Ille et Vilaine (France), Fire Department of Trento (Italy)
- Abstract: The UAV-Retina objectives aims at the creation of s startup company for a flexible autonomous drone platform for search and rescue, using advanced unmanned vehicles and data analytics. The markets for the company will be Improved Explosive Devices detection, support for firefighting operations, and support for avalanche search and rescue operations.

8.3.3. Collaborations with Major European Organizations

- SINTEF, ICT (Norway): Model-driven systems development for the construction of distributed, heterogeneous applications. We collaborate since 2008 and are currently in two FP7 projects together.
- Université du Luxembourg, (Luxembourg): Models runtime for dynamic adaptation and multi-objective elasticity in cloud management; model-driven development.
- KTH, the Royal Institute of Technology (Sweden): continuous software testing, perturbation and diversification.

8.4. International Initiatives

8.4.1. Inria International Labs

IIL CWI-Inria

Associate Team involved in the International Lab:

8.4.1.1. ALE

- Title: Agile Language Engineering
- International Partner (Institution - Laboratory - Researcher):
  - CWI (Netherlands) Tijs van der Storm
- 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).

8.4.2. Inria International Partners

8.4.2.1. 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)

8.4.3. Participation in Other International Programs

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

8.4.3.1. International initiative GEMOC

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

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

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

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

8.5. International Research Visitors

8.5.1. Visits of International Scientists

- Yves Le Traon, Professor at the University of Luxembourg, visited the team in June and July 2018.
- François Fouquet, Junior Researcher at the SnT (Lux), visited the team in March 2018.
- Jordi Cabot, Research Professor at Internet Interdisciplinary Institute, the Research center of the Open University of Catalonia (UOC), SOM Research Lab leader, visited the team in December 2018.
- Erwan Bousse, postdoctoral researcher at TU Wien, Austria, visited the team from January until Aug 2018
- Marcel Heinz, research assistant and PhD student at University of Koblenz-Landau visited the team in Jul 2018

8.5.1.1. Internships

- Enzo Menegaldo, from Jun 2018 until Sep 2018
- Yannick Namour from Apr 2018 until Aug 2018
- Koko Armando Nguepi Kenfack Until Jan 2018
- Anthony Orain from Jun 2018 until Jul 2018
- Max Aguirre, from Jun 2018 until Jul 2018
- Gwendal Didot, from May 2018 until Aug 2018
- Arnaud Gohier, from Apr 2018 until Aug 2018
- Alexis Lemasle, from May 2018 until Aug 2018
- Hugo Martin, from Feb 2018 until Aug 2018

8.5.2. Visits to International Teams

- Fabien Coulon visited CWI for 1 week in June 2018 in the context of the Associated Team ALE.
- Manuel Leduc visited CWI for 1 week in December 2018 in the context of the Associated Team ALE.

- Benoit Combemale made several short visits at CWI in the context of the Associated Team ALE, and visited TU Eindhoven in November 2018.
- Olivier Barais and Amine Benelallam made several short visits at KTH in the context of a collaboration with Prof Monperrus and Prof Baudry.

9. Dissemination

9.1. Promoting Scientific Activities

9.1.1. Scientific Events Organisation

9.1.1.1. General Chair, Scientific Chair

Benoit Combemale has been appointed in 2018 Deputy Editor in Chief of the international journal JOT, and chair of the steering committee of the ACM SIGPLAN conference SLE.
Benoit Baudry was General chair for SSBSE'2018

9.1.1.2. Member of the Organizing Committees

Benoit Combemale has been a co-organizer of the session "Application of Advanced Software Engineering Tools and Methods in the Environmental Sciences" at CMWR'18.

9.1.2. Scientific Events Selection

9.1.2.1. Chair of Conference Program Committees

Benoit Combemale has been co-chair for the Industry Day at MODELS 2018, and the GEMOC Workshop at MODELS 2018.

9.1.2.2. Member of the Conference Program Committees

Mathieu Acher:

- 33rd IEEE/ACM International Conference on Automated Software Engineering
- 22nd International Systems and Software Product Line Conference
- 12th International Workshop on Variability Modelling of Software-Intensive Systems

Olivier Barais:

- PC member for ICWE 2018. Full Research Papers
- PC member for ICWE 2018. Short and Vision Papers
- PC member for SAC'2018
- PC member for SecureMDE 2018 workshop at MODELS'18

Benoit Baudry:

- PC member for SPLC 2018
- PC member for SAC 2018

Arnaud Blouin:

- 30th French speaking conference on Human-Computer Interaction (IHM), 2018
- Second International Workshop on Debugging in Model-Driven Engineering at MODELS'2018 (MDEBug), 2018
- Third International Workshop on Human Factors in Modeling at MODELS'2018 (HuFaMo), 2018
- PC member of the ACM Student Research Competition (SRC) at MODELS 2018

Benoit Combemale:

- PB member for MODELS'18
- PC member for ICMT'18
- PC member for ECMFA'18
- PC member for the MiSE'18 workshop at ICSE'18
- PC member for the EXE'18 workshop at MODELS'18
- PC member for the MDEBug'17 workshop at MODELS'18
- PC member for the DevOps'18 workshop

Jean-Marc Jézéquel:

- PC member for SPLC 2018 Industry Track
- PC member for SEAMS 2018

Johann Bourcier:

- PC member for SEsCPS workshop at ICSE, 2018
- PC member for the eCAS workshop at SASO 2018

Noël Plouzeau:

- PC member for ICSA 2018, Technical Track
- PC member for SEAA 2018, Cloud Based Systems and CPS Systems tracks

9.1.2.3. Reviewer

Arnaud Blouin, external reviewers for:

- EICS'18, ICWE'18

Olivier Barais, external reviewers for:

- ASE'18

Johann Bourcier, external reviewers for:

- Models'18

9.1.3. Journal

9.1.3.1. Member of the Editorial Boards

Benoit Combemale is a member of the editorial board of the journals SoSyM (Springer), COMLAN (Elsevier), and SCP (Elsevier, Advisory Board of the Software Section).

Jean-Marc Jézéquel:

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

Benoit Baudry:

- Associate Editor in Chief of STVR
- Associate Editor in Chief of SOSYM

,

9.1.3.2. Reviewer - Reviewing Activities

In addition to his editorial duties, Benoit Combemale is a regular reviewer for the journals ACM Survey, and IEEE Transactions on Software Engineering (TSE).

Arnaud Blouin:

- COMLAN (Computer Languages, Systems and Structures), JVLC (Journal of Visual Languages and Computing), SCP (Science of Computer Programming), KAIS (Knowledge and Information Systems)

Olivier Barais:

- COMLAN (Computer Languages, Systems and Structures), JVLC (Journal of Visual Languages and Computing), Software and Systems Modeling (SoSyM), IEEE Computer

Noël Plouzeau:

- Software and Systems Modeling (SoSyM)

Johann Bourcier:

- Software and Systems Modeling (SoSyM), Journal of System and Software (JSS), IEEE Communication Magazine

9.1.4. Invited Talks

Mathieu Acher:

- *Reverse Engineering Language Product Lines from Existing DSL Variants*, David Méndez-Acuña, José Galindo, Benoit Combemale, Arnaud Blouin, Benoit Baudry, at GDR-GPL 2018
- *Re-engineering Software Variability into Software Product Lines*, Tewfik Ziadi, Mathieu Acher, at ASE 2018 (tutorial: http://www.ase2018.com/?p=tutorials#spl)

Jean-Marc Jézéquel:

- *Keynote speech for the LIG seminar on the Future of Informatics: "On Turning Domain Knowledge into Tools"*

Benoit Combemale:

- Model Execution: Past, Present and Future. Keynote at EXE'18, DK (14/10/18).
- Modeling For Sustainability - Or How to Make Smart CPS Smarter?. Keynote at Models@run.time'18, DK (14/10/18), and invited talk at TU Wien, AT (15/05/18)
- Execution Framework of the GEMOC Studio. Talk at LangDev'18, NL (08/03/18).

9.1.5. Leadership within the Scientific Community

Mathieu Acher is member of the steering committee of SPLC, the major venue in software product line and variability http://splc.net/.
Arnaud Blouin:

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

Benoit Baudry organized a dagstuhl-Seminar on the relationship between DevOps and quality assurance from a software engineering perspective https://www.dagstuhl.de/de/programm/kalender/semhp/?semnr=18122.

Benoit Combemale:

- Chair of the Steering committee of the ACM SIGPLAN conference SLE
- Founding member and member of the advisory board of the GEMOC initiative.
- Chair of the Eclipse Research Consortium GEMOC and the Eclipse Project GEMOC Studio.

9.1.6. Scientific Expertise

Olivier Barais is an external expert for the H2020 ENACT project. Olivier Barais provides scientific expertises for DGRI international program (20 proposals per year)

9.1.7. Research Administration

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

9.2. Teaching - Supervision - Juries

9.2.1. Teaching

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

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

Olivier Barais is deputy director of the electronics and computer science teaching department of the University of Rennes 1. Olivier Barais is the head of the final year of the Master in Computer Science at the University of Rennes 1. Johann Bourcier is co-manager of the Home-Automation option at the ESIR engineering school in Rennes. Arnaud Blouin is in charge of industrial relationships for the computer science department at INSA Rennes.

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

Mathieu Acher gave courses to EJCP (Ecole Jeune Chercheur en Programmation), a well-known, national training school for PhD students https://ejcp2018.sciencesconf.org/resource/page/id/5

9.2.2. Supervision

- PhD in progress: Alejandro Gomez Boix, *Distributed counter-measure against browser fingerprinting*, 2016, B. Baudry, D. Bromberg
- 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, *Efficient resources management for hybrid cloud computing*, 2016, O. Barais
- PhD in progress: Ludovic Mouline, *Omniscient fault localization in IOT systems using model-driver data analytics*, 2016, O. Barais, Y. Le-Traon, J. Bourcier.
- PhD in progress: Oscar Luis, *Automatic test amplification*, 2016, B. Baudry
- PhD in progress: Alexandre Rio, *Demand Side Management A model driven approach to promote energy self-consumption* , 2016, O. Barais, Y. Maurel
- PhD in progress: Dorian Leroy, *A generic and generative white-box testing framework for model transformations*, 2017, B. Combemale.
- PhD in progress: Fabien Coulon, *Web engineering for domain-specific modeling languages*, 2017, B. Combemale, S. Begaudeau.
- PhD in progress: Romain Lebouc, *Techniques de test front-end en DevOps*, 2018, A. Blouin, N. Plouzeau, A. Ribault

- PhD in progress: June Benvegnu-Sallou, Decision support for the assessment of risks associated with the operation of underground environments, 2018, J-R. De dreuzy, B. Combemale, J. Bourcier.
- PhD in progress: Pierre JeanJean, *Refining simulators by analyzing execution traces of complex systems*, 2018, O. Barais, B. Combemale.
- PhD in progress: Akbar-pranata Alif, *Chaos Engineering for IoT and Network Services*, 2018, O. Barais, J. Bourcier.
- PhD in progress: Antoine Cheron, *Abstractions for linked data and the programmable web*, 2018, O. Barais, J. Bourcier.
- PhD in progress: Gauthier LYAN, *Urban mobility: machine learning for building simulators using large amounts of data*, 2018, J-M. Jééquel, D. Gross Amblard.
- PhD in progress: Hugo Martin, *Learning variability*, 2018, M. Acher.
- PhD: Gwendal Le Moulec, *Synthèse d'applications de Réalité Virtuelle à partir de modèles*, sept. 2018, B. Arnaldi, A. Blouin, V. Gouranton
- PhD: Paul Temple, *Investigate the Matrix: Leveraging Variability to Specialize Software and Test Suites*, dec. 2018, J-M. Jézéquel, M. Acher
- PhD: Kevin Corre, *User controlled trust and security level of Web real-time communications*, May. 2018, O. Barais, G. Sunye

9.2.3. Juries

9.2.3.1. Jean-Marc Jézéquel

was in the examination committee of the following PhD and HDR thesis:

- Jessie carbonnel, November 2018, Univ Montpellier (examiner)
- Paul Temple, December 2018, Univ Rennes I (Advisor)
- Johan Bourcier, HDR, December 2018, Univ Rennes I

9.2.3.2. Mathieu Acher

was in the examination committee of the following PhD thesis:

- Jessie carbonnel, November 2018, Univ Montpellier (examiner)
- Paul Temple, December 2018, Univ Rennes I (co-supervisor)

9.2.3.3. Olivier Barais

was in the examination committee of the following PhD thesis:

- Kevin Corre, May 2018, Univ Rennes I, Supervisor
- Thomas Durieux, Sep 2018, Univ Lille, Reviewer
- Mathieu Allon, Sep 2018, Univ Lille, Reviewer
- Tayeb LEMLOUMA, HDR, June 2018, Univ Lille, President
- Simon Bouget, Oct 2018, Univ Lille, President
- Mounir Chadli, Nov 2018, Univ Rennes I, President
- Arezki Laga, Dec 2018, Univ Bretagne Occidentale, President
- Franck Petitdemange, Dec 2018, Univ Bretagne Sud, President

9.2.3.4. Benoit Baudry

was in the examination committee of the following PhD thesis:

- Hugo Brunelière (IMT), Dec 2018, Reviewer
- Johanna Binti Ahmad (Univ. Putra Malaysia), 2018, Reviewer
- Daniel Schoepe (Chalmers), 2018, Reviewer

9.2.3.5. Benoit Combemale

was in the examination committee of the following PhD thesis:

- Stephanie Challita, Univ Lille 1 (December 2018), Reviewer
- Pablo Inostroza Valdera, Univ. Amsterdam & CWI (November 2018), The Netherlands

9.3. Popularization

- Exhibitions to Open Cloud Forum, Paris 21/03/2018
- Exhibitions to EclipsCon Toulouse 13-14/04 2018
- Exhibitions to Paris Open Source Submit 05-06/12/2018

9.3.1. Internal or external Inria responsibilities

- Olivier Barais was in charge of the recruitment committee for the position of associate professor at the University of Rennes 1 in cyber security.
- Olivier Barais was a member the recruitment committee for the position of associate professor at the University of Nantes in Software Engineering.
- Olivier Barais was a member of the recruitment committee for the position of associate professor at the University of Bordeaux in Software Engineering.

9.3.2. Interventions

- Exhibitions to the Lycée Sévigné to present career paths in Research
- Exhibitions to the "forum des métiers du collège Bourgchevreuil"

9.3.3. Internal action

- Olivier Barais provide one day course on Modern Web Engineering to SED Engineer.

10. Bibliography

Major publications by the team in recent years

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

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

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

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

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

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

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

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

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

[10] J. A. GALINDO DUARTE, M. 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] K. CORRE. *User controlled trust and security level of Web real-time communications*, Université Rennes 1, May 2018, https://tel.archives-ouvertes.fr/tel-01943728

[17] G. LE MOULEC. *Model driven synthesis of virtual reality applications*, INSA de Rennes, September 2018, https://tel.archives-ouvertes.fr/tel-01959918

Articles in International Peer-Reviewed Journals

[18] 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. 1-41 [*DOI :* 10.1007/s11219-017-9400-8], https://hal.inria.fr/hal-01688247

[19] A. BENELALLAM, A. GÓMEZ, M. TISI, J. CABOT. *Distributing Relational Model Transformation on MapReduce*, in "Journal of Systems and Software", April 2018, vol. 142, pp. 1-20 [*DOI :* 10.1016/J.JSS.2018.04.014], https://hal.archives-ouvertes.fr/hal-01863885

[20] A. BLOUIN, V. LELLI, B. BAUDRY, F. COULON. *User Interface Design Smell: Automatic Detection and Refactoring of Blob Listeners*, in "Information and Software Technology", May 2018, vol. 102, pp. 49-64 [*DOI :* 10.1016/J.INFSOF.2018.05.005], https://hal.inria.fr/hal-01499106

[21] E. BOUSSE, D. LEROY, B. COMBEMALE, M. WIMMER, B. BAUDRY. *Omniscient Debugging for Executable DSLs*, in "Journal of Systems and Software", March 2018, vol. 137, pp. 261-288 [*DOI :* 10.1016/J.JSS.2017.11.025], https://hal.inria.fr/hal-01662336

[22] B. COMBEMALE, J. KIENZLE, G. MUSSBACHER, O. BARAIS, E. BOUSSE, W. CAZZOLA, P. COLLET, T. DEGUEULE, R. HEINRICH, J.-M. JÉZÉQUEL, M. LEDUC, T. MAYERHOFER, S. MOSSER, M. SCHÖTTLE, M. STRITTMATTER, A. WORTMANN. *Concern-Oriented Language Development (COLD): Fostering Reuse in Language Engineering*, in "Computer Languages, Systems and Structures", 2018, vol. 54, pp. 139-155 [*DOI :* 10.1016/J.CL.2018.05.004], https://hal.archives-ouvertes.fr/hal-01803008

[23] B. DANGLOT, O. LUIS VERA-PÉREZ, B. BAUDRY, M. MONPERRUS. *Automatic Test Improvement with DSpot: a Study with Ten Mature Open-Source Projects*, in "Empirical Software Engineering", 2018, pp. 1-35, https://hal.inria.fr/hal-01923575

[24] B. DANGLOT, P. PREUX, B. BAUDRY, M. MONPERRUS. *Correctness Attraction: A Study of Stability of Software Behavior Under Runtime Perturbation*, in "Empirical Software Engineering", August 2018, vol. 23, n° 4, pp. 2086–2119, https://arxiv.org/abs/1611.09187 [*DOI :* 10.1007/s10664-017-9571-8], https://hal.archives-ouvertes.fr/hal-01378523

[25] A. HALIN, A. NUTTINCK, M. ACHER, X. DEVROEY, G. PERROUIN, B. BAUDRY. *Test them all, is it worth it? Assessing configuration sampling on the JHipster Web development stack*, in "Empirical Software Engineering", July 2018, pp. 1–44, https://arxiv.org/abs/1710.07980 [*DOI :* 10.1007/s10664-018-9635-4], https://hal.inria.fr/hal-01829928

[26] J. KIENZLE, G. MUSSBACHER, B. COMBEMALE, J. DEANTONI. *A Unifying Framework for Homogeneous Model Composition*, in "Software & Systems Modeling", January 2019, pp. 1-19 [*DOI :* 10.1007/s10270-018-00707-8], https://hal.inria.fr/hal-01949050

[27] G. LE MOULEC, A. BLOUIN, V. GOURANTON, B. ARNALDI. *Automatic Production of End User Documentation for DSLs*, in "Computer Languages, Systems and Structures", July 2018, vol. 54, pp. 337-357, Accepted for publication in COMLAN [*DOI :* 10.1016/J.CL.2018.07.006], https://hal.inria.fr/hal-01549042

[28] O. L. VERA-PÉREZ, B. DANGLOT, M. MONPERRUS, B. BAUDRY. *A Comprehensive Study of Pseudo-tested Methods*, in "Empirical Software Engineering", 2018, pp. 1-33 [*DOI :* 10.1007/s10664-018-9653-2], https://hal.inria.fr/hal-01867423

International Conferences with Proceedings

[29] M. ACHER, R. E. LOPEZ-HERREJON, R. RABISER. *Teaching Software Product Lines: A Snapshot of Current Practices and Challenges (Journal-First Abstract)*, in "SPLC2018 - 22nd International Systems and Software Product Line Conference", Gothenburg, Sweden, September 2018, 1 p. , https://hal.inria.fr/hal-01829933

[30] 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, ACM, February 2018, pp. 83-88 [*DOI :* 10.1145/3168365.3168372], https://hal.inria.fr/hal-01659161

[31] F. BORDELEAU, B. COMBEMALE, R. ERAMO, M. V. D. BRAND, M. WIMMER. *Tool-Support of Socio-Technical Coordination in the Context of Heterogeneous Modeling: A Research Statement and Associated Roadmap*, in "GEMOC 2018", Copenhagen, Denmark, 2018, https://hal.inria.fr/hal-01958443

[32] *Best Paper*
J.-M. BRUEL, B. COMBEMALE, E. GUERRA, J.-M. JÉZÉQUEL, J. KIENZLE, J. DE LARA, G. MUSSBACHER, E. SYRIANI, H. VANGHELUWE. *Model Transformation Reuse across Metamodels - A classification and comparison of approaches*, in "ICMT 2018 - International Conference on Theory and Practice of Model Transformations", Toulouse, France, LNCS, Springer, June 2018, vol. 10888, pp. 92-109 [*DOI :* 10.1007/978-3-319-93317-7_4], https://hal.inria.fr/hal-01910113.

[33] *Best Paper*
F. COULON, T. DEGUEULE, T. VAN DER STORM, B. COMBEMALE. *Shape-Diverse DSLs: Languages without Borders (Vision Paper)*, in "SLE 2018 - 11th ACM SGIPLAN International Conference on Software Language Engineering", Boston, United States, ACM, November 2018, pp. 215-219 [*DOI :* 10.1145/3276604.3276623], https://hal.archives-ouvertes.fr/hal-01889155.

[34] J.-E. DARTOIS, A. KNEFATI, J. BOUKHOBZA, O. BARAIS. *Using Quantile Regression for Reclaiming Unused Cloud Resources while achieving SLA*, in "CloudCom 2018 - 10th IEEE International Conference on Cloud Computing Technology and Science", Nicosia, Cyprus, IEEE, December 2018, pp. 1-10, https://hal.inria.fr/hal-01898438

[35] A. GÓMEZ-BOIX, P. LAPERDRIX, B. BAUDRY. *Hiding in the Crowd: an Analysis of the Effectiveness of Browser Fingerprinting at Large Scale*, in "WWW2018 - TheWebConf 2018 : 27th International World Wide Web Conference", Lyon, France, April 2018, pp. 1-10 [*DOI :* 10.1145/3178876.3186097], https://hal.inria.fr/hal-01718234

[36] *Best Paper*
M. LEDUC, T. DEGUEULE, B. COMBEMALE. *Modular Language Composition for the Masses*, in "SLE 2018 - 11th ACM SIGPLAN International Conference on Software Language Engineering", Boston, United States, November 2018, pp. 1-12 [*DOI :* 10.1145/3276604.3276622], https://hal.inria.fr/hal-01890446.

[37] B. LELANDAIS, M.-P. OUDOT, B. COMBEMALE. *Fostering metamodels and grammars within a dedicated environment for HPC: the NabLab environment (tool demo)*, in "SLE 2018 - International Conference on Software Language Engineering", Boston, United States, November 2018, pp. 1-9 [*DOI :* 10.1145/3276604.3276620], https://hal.inria.fr/hal-01910139

[38] L. MOULINE, A. BENELALLAM, F. FOUQUET, J. BOURCIER, O. BARAIS. *A Temporal Model for Interactive Diagnosis of Adaptive Systems*, in "ICAC 2018 - IEEE International Conference on Autonomic Computing", Trento, Italy, September 2018, pp. 1-6, https://hal.inria.fr/hal-01862964

[39] L. MOULINE, A. BENELALLAM, T. HARTMANN, F. FOUQUET, J. BOURCIER, B. MORIN, O. BARAIS. *Enabling Temporal-Aware Contexts for Adaptative Distributed Systems Temporal Context Representation System updates*, in "SAC 2018 - The 33rd ACM/SIGAPP Symposium On Applied Computing", Pau, France, SAC 2018: SAC 2018: Symposium on Applied Computing , April 9–13, 2018, Pau, France, April 2018, pp. 1-8 [*DOI :* 10.1145/3167132.3167286], https://hal.inria.fr/hal-01723451

[40] C. SOTO-VALERO, J. BOURCIER, B. BAUDRY. *Detection and Analysis of Behavioral T-patterns in Debugging Activities*, in "MSR 2018 - Mining Software Repositories", Gothenburg, Sweden, May 2018, pp. 1-4 [*DOI :* 10.1145/3196398.3196452], https://hal.inria.fr/hal-01763369

[41] P. TEMPLE, M. ACHER, J.-M. JÉZÉQUEL. *Multimorphic Testing*, in "ICSE '18 - ACM/IEEE 40th International Conference on Software Engineering", Gothenburg, Sweden, May 2018, pp. 1-2 [*DOI :* 10.1145/3183440.3195043], https://hal.inria.fr/hal-01730163

[42] 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. 728-741 [*DOI :* 10.1109/SP.2018.00008], https://hal.inria.fr/hal-01652021

[43] O. L. VERA-PÉREZ, M. MONPERRUS, B. BAUDRY. *Descartes: a PITest engine to detect pseudo-tested methods - tool demonstration*, in "ASE 2018 - 33rd ACM/IEEE International Conference on Automated Software Engineering", Montpellier, France, ACM Press, September 2018, pp. 908-911 [*DOI :* 10.1145/3238147.3240474], https://hal.inria.fr/hal-01870976

Conferences without Proceedings

[44] D. LEROY, E. BOUSSE, A. MEGNA, B. COMBEMALE, M. WIMMER. *Trace Comprehension Operators for Executable DSLs*, in "ECMFA 2018 - 14th European Conference on Modelling Foundations and Applications", Toulouse, France, LNCS, Springer, June 2018, vol. 10890, pp. 293-310 [*DOI :* 10.1007/978-3-319-92997-2_19], https://hal.inria.fr/hal-01803031

[45] J. MARTINEZ, J.-S. SOTTET, A. GARCÍA FREY, T. BISSYANDÉ, T. ZIADI, J. KLEIN, P. TEMPLE, M. ACHER, Y. LE TRAON. *Towards Estimating and Predicting User Perception on Software Product Variants*, in

"ICSR 2018 - International Conference on Software Reuse", Madrid, Spain, LNCS, Springer, May 2018, vol. 10826, pp. 23-40 [*DOI :* 10.1007/978-3-319-90421-4_2], https://hal.sorbonne-universite.fr/hal-01720519

[46] A. RIO, Y. MAUREL, O. BARAIS, Y. BUGNI. *Efficient use of local energy: An activity oriented modeling to guide Demand Side Management*, in "MODELS 2018 - 21th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems", Copenhagen, Denmark, ACM Press, October 2018, pp. 458-468 [*DOI :* 10.1145/3239372.3239391], https://hal.archives-ouvertes.fr/hal-01913169

References in notes

[47] 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

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

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

[50] 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

[51] 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

[52] 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

[53] 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

[54] 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

[55] 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

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

[57] 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

[58] 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

[59] 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° 4, pp. 387–404

[60] 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

[61] 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

[62] 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

[63] J. COPLIEN, D. HOFFMAN, D. WEISS. *Commonality and Variability in Software Engineering*, in "IEEE Software", 1998, vol. 15, n° 6, pp. 37–45

[64] 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° 5, pp. 593–615

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

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

[67] 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° 2, pp. 182–197

[68] 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

[69] 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

[70] 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

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

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

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

[74] 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

[75] 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

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

[77] 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

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

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

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

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

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

[83] 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

[84] 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

[85] 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.

[86] 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

[87] 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

[88] R. NIEDERMAYR, E. JUERGENS, S. WAGNER. *Will my tests tell me if I break this code?*, in "Proceedings of the International Workshop on Continuous Software Evolution and Delivery", ACM, 2016, pp. 23–29

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

[90] 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

[91] 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

[92] 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

[93] 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

[94] 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

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

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

[97] 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

[98] 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

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

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

[101] 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

[102] 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

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

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

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

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

[107] 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

[108] 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

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