IN PARTNERSHIP WITH:
Institut national des sciences appliquées de Rennes

Université Rennes 1

Activity Report 2019

Project-Team DIVERSE

Diversity-centric Software Engineering

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

RESEARCH CENTER
Rennes - Bretagne-Atlantique

THEME
Distributed programming and software engineering

Table of contents

Project-Team DIVERSE

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

Keywords:

Computer Science and Digital Science:

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

Research Scientist

Djamel Eddine Khelladi [CNRS, Researcher]

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 1 Capitole, Professor, 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, Post-Doctoral Fellow]

PhD Students

Alif Akbar Pranata [Inria, PhD Student]
June Benvegnu Sallou [Univ de Rennes I, PhD Student]
Emmanuel Chebbi [Inria, PhD Student, from Oct 2019]
Antoine Cheron [FaberNovel, PhD Student, granted by CIFRE]
Fabien Coulon [Obeo, PhD Student, granted by CIFRE]
Jean-Emile Dartois [Institut de recherche technologique B-com, PhD Student]
Alejandro Gomez Boix [Inria, PhD Student, until Oct 2019]
Pierre Jeanjean [Inria, PhD Student]
Romain Lebouc [KEREVAL, PhD Student, granted by CIFRE]
Manuel Leduc [Univ de Rennes I, PhD Student, until Nov 2019]
Dorian Leroy [TU Wien, PhD Student]
Gauthier Lyan [Keolis Rennes, PhD Student, granted by CIFRE]
Hugo Martin [Univ de Rennes I, PhD Student]
Ludovic Mouline [SnT Luxembourg, PhD Student, until Nov 2019]
Youssou Ndiaye [Orange Labs, PhD Student, until Nov 2019, granted by CIFRE]
Oscar Luis Vera Perez [Inria, PhD Student]

Technical staff

Amine Benelallam [Inria, Engineer, until Feb 2019]
Emmanuel Chebbi [Inria, Engineer, from Aug 2019 until Sep 2019]
Caroline Landry [Inria, Engineer, until Nov 2019]
Didier Vojtisek [Inria, Engineer]

Interns and Apprentices

Benoit Bernier [Inria, from Jun 2019 until Sep 2019]
Alexandre Blandel [Inria, from Jun 2019 until Sep 2019]
Guillaume Collet [Inria, from Apr 2019 until Aug 2019]
Valentin Duval [Univ de Rennes I, from Apr 2019 until Jul 2019]
Duncan Goldschild [Univ de Rennes I, from Jul 2019 until Aug 2019]
Philemon Houdaille [Inria, from Jun 2019 until Jul 2019]
Gwendal Jouneaux [Univ de Rennes I, from Jun 2019 until Aug 2019]
Ludovic Richoux [Univ de Rennes I, from Jun 2019 until Aug 2019]
Paul Saffray [Univ de Rennes I, from Jun 2019 until Aug 2019]
Tony Werth [Inria, from Jun 2019 until Sep 2019]

Administrative Assistants

Fanny Banor [CNRS, Administrative Assistant, until Mar 2019]
Sophie Maupile [CNRS, Administrative Assistant, from Apr 2019]

Visiting Scientists

Nelly Bencomo [Aston University, UK, from Sep 2019]
Sophie Ebersold [Université de Toulouse, Mar 2019]
Paul Temple [University de Namur, until Feb 2019]

External Collaborator

Gurvan Le Guernic [DGA]

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 all 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 both critical and challenging software layer properties that must deal with the following 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 diversity is a great opportunity to allow software to spontaneously explore alternative design solutions. Concretely, we want to provide software engineers with the following abilities:

- 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 evolution 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) [120]. 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 [91]. 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 [73]. 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 in the development of a sound *Software Language Engineering* [1], including a unified typing theory that integrate models as first class entities [123].

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. [97] indicates 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 runtime 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 runtime behavior is extremely promising [105].

3.1.2. Variability modeling

While the basic vision underlying *Software Product Lines* (SPL) can probably be traced back to David Parnas' seminal article [113] 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 [111]. 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* [110]. Bosch provides a different definition [79]: *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 [85]. 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

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

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 [93]. 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 [80]. Variability management is thus growingly seen as the cornerstone of SPL development, covering the entire development life cycle, from requirements elicitation [125] to product derivation [130] to product testing [109], [108].

Halmans *et al.* [93] 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.* [115]: 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.* [103] (feature, hardware platform, performances and attributes variability) or Bass *et al.* [71] 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* [99]. 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 [80]. 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 [75], [87]. Model Driven Engineering (MDE) makes it possible to automate this weaving process [98]. 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 [106]) manipulating these models to produce for instance a detailed design that can ultimately be transformed to code, or to test suites [114], or other software artifacts.

3.1.3. Component-based software development

Component-based software development [124] 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 [86]: description of requirements on services required from the other components; indirect connections between components thanks to ports and connectors constructs [101]; hierarchical definition of components (assemblies of components can define new component types); connectors supporting various communication semantics [83]; quantitative properties on the services [78].

In recent years component-based architectures have evolved from static designs to dynamic, adaptive designs (e.g., SOFA [83], Palladio [76], Frascati [107]). 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 [100]. 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 runtime, without human intervention, by adapting themselves [84], [127]. 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 [77], dynamic component based systems must separate architectural design and platform compatibility. This requires support for heterogeneity, since platform evolution 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 [118]". In practice, component-based (and/or service-based) platforms offer reflection APIs that make it possible to introspect the system (to determine 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 [95] 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 [92]. Multi Objectives Search based techniques [89] 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 [126] mainly relies on model analysis, constraint solving [88] and search-based reasoning [102]. DIVERSE leverages in particular the applications of model-based testing in the context of highly-configurable systems and [128] interactive systems [104] as well as recent advances based on diversity for test cases selection [96].

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 [112], or Xholon [2]. All these simulation environments operate on homogeneous environment models. However, to handle diversity in software systems, we also leverage recent advances in heterogeneous simulation. Ptolemy [82] 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 [72] provides modeling elements amenable to semantically equivalent mathematical models. Metropolis offers a precise semantics flexible enough to support different models of computation. ModHel'X [94] 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 [70], recovery blocks [116] and code randomization [74], 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 [121], [119]. Such methods have been used for example to understand the impact of new software development paradigms [81]. 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 [69].

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

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

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

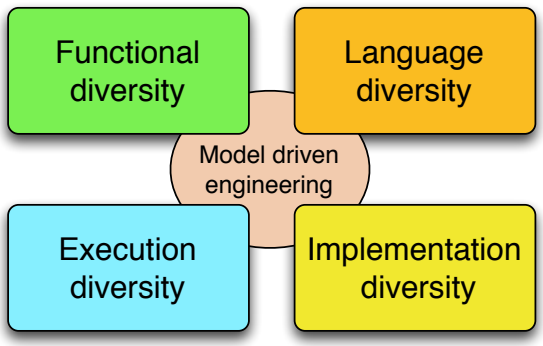

Figure 1. The four research axes of DIVERSE, which rely on a MDE scientific background

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 [120], [91]. 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. This framework 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 [122] (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 evolution 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 aim at developing 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. Evolution possibilities 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 (e.g., 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 runtime. 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 unforeseen evolution cases.

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 reconfiguration 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 reconfiguration.

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 execution 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 evolution cases 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 unforeseen 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 evolution, etc. Current fault-tolerance [116] and security [90] 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 rely on implementation 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 and active 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" [117], [129].

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. 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 name 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 logic 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 [37], [28] and [25] 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.
- We received the "Data Showcase Award" (Figure 2) at the MSR'19 conference (Mining Software Repositories 2019) for the dataset described in the following paper [55] and publicly available on Zenodo (https://zenodo.org/record/1489120).
- Since this year two former PhD students of the team now have a full time researcher position at CNRS: Pierre Laperdrix and Thomas Degueule.
- Four new PhDs and one new HDR have been successfully defended this year.
- A new CNRS junior researcher, Djamel Eddine Khelladi, has joined the team in February 2019. Since his arrival, he submitted a Marie Skłodowska-Curie Action (MSCA) Individual Fellowships (IF), as well as an ANR JCJC in phase 1. Both projects, respectively, CoEvoCCT and MC-Evo^2 are on the topics of software evolution and co-evolution. His research amplifies a new axis around software evolution and maintenance. First results led to the publication at 42nd International Conference on Software Engineering, ICSE, 2020, Seoul, South Korea, an A* top conference in the field of software engineering.

Figure 2. Data Showcase Award, MSR'19

4.1.1. Awards

Most Influential Paper (MIP) award at SLE 2019 https://ins2i.cnrs.fr/fr/cnrsinfo/des-scientifiques-primes-pour-leurs-travaux-sur-les-feature-models

BEST PAPER AWARD:

[55]
A. BENELALLAM, N. HARRAND, C. SOTO-VALERO, B. BAUDRY, O. BARAIS. *The Maven Dependency Graph: a Temporal Graph-based Representation of Maven Central*, in "MSR 2019 - 16th International Conference on Mining Software Repositories", Montreal, Canada, ACM, May 2019, pp. 344-348 [*DOI :* 10.1109/MSR.2019.00060], https://hal.archives-ouvertes.fr/hal-02080243

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 detail 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 detail 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.

- Panopticlick https://panopticlick.eff.org/
- BrowserSpy http://browserspy.dk/

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 100,000 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

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/doc/1.4/frascati-userguide.html).
- 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).

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. 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

5.7. 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/

5.8. 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: hal-01499106v5 - hal-01308625v2
- URL: https://github.com/diverse-project/InspectorGuidget

5.9. Descartes

KEYWORDS: Software testing - Mutation analysis

FUNCTIONAL DESCRIPTION: Descartes evaluates the capability of your test suite to detect bugs using extreme mutation testing.

Descartes is a mutation engine plugin for PIT which implements extreme mutation operators as proposed in the paper *Will my tests tell me if I break this code?*.

- Participants: Oscar Luis Vera Perez, Benjamin Danglot, Benoit Baudry and Martin Monperrus
- Partner: KTH Royal Institute of Technology
- Contact: Benoit Baudry
- Publications: Descartes: a PITest engine to detect pseudo-tested methods - Tool Demonstration - A Comprehensive Study of Pseudo-tested Methods
- URL: https://github.com/STAMP-project/pitest-descartes

5.10. PitMP

PIT for Multi-module Project

KEYWORDS: Mutation analysis - Mutation testing - Java - JUnit - Maven

FUNCTIONAL DESCRIPTION: PIT and Descartes are mutation testing systems for Java applications, which allows you to verify if your test suites can detect possible bugs, and so to evaluate the quality of your test suites. They evaluate the capability of your test suite to detect bugs using mutation testing (PIT) or extreme mutation testing (Descartes). Mutation testing does it by introducing small changes or faults into the original program. These modified versions are called mutants. A good test suite should able to kill or detect a mutant. Traditional mutation testing works at the instruction level, e.g., replacing ">" by "<=", so the number of generated mutants is huge, as the time required to check the entire test suite. That's why Extreme Mutation strategy appeared. In Extreme Mutation testing, the whole body of a method under test is removed. Descartes is a mutation engine plugin for PIT which implements extreme mutation operators. Both provide reports combining, line coverage, mutation score and list of weaknesses in the source.

- Partners: CSQE - KTH Royal Institute of Technology - ENGINEERING
- Contact: Caroline Landry
- URL: https://github.com/STAMP-project/pitmp-maven-plugin

6. New Results

6.1. Results on Variability modeling and management

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.

6.1.1. Variability and testing.

The performance of software systems (such as speed, memory usage, correct identification rate) tends to be an evermore important concern, often nowadays on par with functional correctness for critical systems. Systematically testing these performance concerns is however extremely difficult, in particular because there exists no theory underpinning the evaluation of a performance test suite, i.e., to tell the software developer whether such a test suite is "good enough" or even whether a test suite is better than another one. This work [37] proposes to apply **Multimorphic testing** and empirically assess the effectiveness of performance test suites of software systems coming from various domains. By analogy with mutation testing, our core idea is to leverage the typical configurability of these systems, and to check whether it makes any difference in the outcome of the tests: i.e., are some tests able to "kill" underperforming system configurations? More precisely, we propose a framework for defining and evaluating the coverage of a test suite with respect to a quantitative property of interest. Such properties can be the execution time, the memory usage or the success rate in tasks performed by a software system. This framework can be used to assess whether a new test case is worth adding to a test suite or to select an optimal test suite with respect to a property of interest. We evaluate several aspects of our proposal through 3 empirical studies carried out in different fields: object tracking in videos, object recognition in images, and code generators.

6.1.2. Variability, sampling, and SAT.

Uniform or near-uniform generation of solutions for large satisfiability formulas is a problem of theoretical and practical interest for the testing community. Recent works proposed two algorithms (namely UniGen and QuickSampler) for reaching a good compromise between execution time and uniformity guarantees, with empirical evidence on SAT benchmarks. In the context of highly-configurable software systems (e.g., Linux), it is unclear whether UniGen and QuickSampler can scale and sample uniform software configurations. We perform a thorough experiment on 128 real-world feature models. We find that UniGen is unable to produce SAT solutions out of such feature models. Furthermore, we show that QuickSampler does not generate uniform samples and that some features are either never part of the sample or too frequently present. Finally, using a case study, we characterize the impacts of these results on the ability to find bugs in a configurable system. Overall, our results suggest that we are not there: more research is needed to explore the cost-effectiveness of uniform sampling when testing large configurable systems. More details [51]. In general, we are investigating sampling algorithms for cost-effectively exploring configuration spaces (see also [63], [67].

6.1.3. Variability and 3D printing.

Configurators rely on logical constraints over parameters to aid users and determine the validity of a configuration. However, for some domains, capturing such configuration knowledge is hard, if not infeasible. This is the case in the 3D printing industry, where parametric 3D object models contain the list of parameters and their value domains, but no explicit constraints. This calls for a complementary approach that learns what configurations are valid based on previous experiences. In this work [41], we report on preliminary experiments showing the capability of state-of-the-art classification algorithms to assist the configuration process. While machine learning holds its promises when it comes to evaluation scores, an in-depth analysis reveals the opportunity to combine the classifiers with constraint solvers.

6.1.4. Variability and video processing.

In an industrial project [24], we addressed the challenge of developing a software-based video generator such that consumers and providers of video processing algorithms can benchmark them on a wide range of video variants. We have designed and developed a variability modeling language, called VM, resulting from the close collaboration with industrial partners during two years. We expose the specific requirements and advanced variability constructs we developed and used to characterize and derive variations of video sequences. The results of our experiments and industrial experience show that our solution is effective to model complex variability information and supports the synthesis of hundreds of realistic video variants. From the software language perspective, we learned that basic variability mechanisms are useful but not enough; attributes and multi-features are of prior importance; meta-information and specific constructs are relevant for scalable and

purposeful reasoning over variability models. From the video domain and software perspective, we report on the practical benefits of a variability approach. With more automation and control, practitioners can now envision benchmarking video algorithms over large, diverse, controlled, yet realistic datasets (videos that mimic real recorded videos) – something impossible at the beginning of the project.

6.1.5. Variability and adversarial machine learning

Software product line engineers put a lot of effort to ensure that, through the setting of a large number of possible configuration options, products are acceptable and well-tailored to customers' needs. Unfortunately, options and their mutual interactions create a huge configuration space which is intractable to exhaustively explore. Instead of testing all products, machine learning is increasingly employed to approximate the set of acceptable products out of a small training sample of configurations. Machine learning (ML) techniques can refine a software product line through learned constraints and a priori prevent non-acceptable products to be derived. In this work [53], we use adversarial ML techniques to generate adver- sarial configurations fooling ML classifiers and pinpoint incorrect classifications of products (videos) derived from an industrial video generator. Our attacks yield (up to) a 100% misclassification rate and a drop in accuracy of 5%. We discuss the implications these results have on SPL quality assurance.

6.1.6. Variability, Linux and machine learning

Given a configuration, can humans know in advance the build status, the size, the compilation time, or the boot time of a Linux kernel? Owing to the huge complexity of Linux (there are more than 15000 options with hard constraints and subtle interactions), machines should rather assist contributors and integrators in mastering the configuration space of the kernel. We have developed TuxML https://github.com/TuxML/ an open-source tool based on Docker/Python to massively gather data about thousands of kernel configurations. 200K+ configurations have been automatically built and we show how machine learning can exploit this information to predict properties of unseen Linux configurations, with different use cases (identification of influential/buggy options, finding of small kernels, etc.) The vision is that a continuous understanding of the configuration space is undoubtedly beneficial for the Linux community, yet several technical challenges remain in terms of infrastructure and automation.

Two preprints are available [62] and [49].

A talk has been given at Embedded Linux Conference Europe 2019 (co-located with Open Source Summit 2019) in Lyon about "Learning the Linux Kernel Configuration Space: Results and Challenges" [54].

6.1.7. Variability and machine learning

We gave a tutorial [49] at SPLC 2019 and introduce how machine learning can be used to support activities related to the engineering of configurable systems and software product lines. To the best of our knowledge, this is the first practical tutorial in this trending field. The tutorial is based on a systematic literature review [67] and includes practical tasks (specialization, performance prediction) on real-world systems (VaryLaTeX, x264).

6.2. Results on Software Language Engineering

6.2.1. Software Language Extension Problem

The problem of software language extension and composition drives much of the research in *Software Language Engineering* (SLE). Although various solutions have already been proposed, there is still little understanding of the specific ins and outs of this problem, which hinders the comparison and evaluation of existing solutions. In [34], we introduce the Language Extension Problem as a way to better qualify the scope of the challenges related to language extension and composition. The formulation of the problem is similar to the seminal Expression Problem introduced by Wadler in the late nineties, and lift it from the extensibility of single constructs to the extensibility of groups of constructs, i.e., software languages. We provide a comprehensive definition of the actual constraints when considering language extension, and believe the Language Extension Problem will drive future research in SLE, the same way the original Expression

Problem helped to understand the strengths and weaknesses of programming languages and drove much research in programming languages.

6.2.2. A unifying framework for homogeneous model composition

The growing use of models for separating concerns in complex systems has lead to a proliferation of model composition operators. These composition operators have traditionally been defined from scratch following various approaches differing in formality, level of detail, chosen paradigm, and styles. Due to the lack of proper foundations for defining model composition (concepts, abstractions, or frameworks), it is difficult to compare or reuse composition operators. In [33], we stipulate the existence of a unifying framework that reduces all structural composition operators to structural merging, and all composition operators acting on discrete behaviors to event scheduling. We provide convincing evidence of this hypothesis by discussing how structural and behavioral homogeneous model composition operators (i.e., weavers) can be mapped onto this framework. Based on this discussion, we propose a conceptual model of the framework, and identify a set of research challenges, which, if addressed, lead to the realization of this framework to support rigorous and efficient engineering of model composition operators for homogeneous and eventually heterogeneous modeling languages.

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

Executable Domain-Specific Modeling Languages (xDSMLs) enable the application of early dynamic verification and validation (V&V) techniques for behavioral models. At the core of such techniques, execution traces are used to represent the evolution of models during their execution. In order to construct execution traces for any xDSML, generic trace metamodels can be used. Yet, regarding trace manipulations, generic trace metamodels lack efficiency in time because of their sequential structure, efficiency in memory because they capture superfluous data, and usability because of their conceptual gap with the considered xDSML. Our contribution in [26] is a novel generative approach that defines a multidimensional and domain-specific trace metamodel enabling the construction and manipulation of execution traces for models conforming to a given xDSML. Efficiency in time is improved by providing a variety of navigation paths within traces, while usability and memory are improved by narrowing the scope of trace metamodels to fit the considered xDSML. We evaluated our approach by generating a trace metamodel for fUML and using it for semantic differencing, which is an important V&V technique in the realm of model evolution. Results show a significant performance improvement and simplification of the semantic differencing rules as compared to the usage of a generic trace metamodel.

6.2.4. From DSL specification to interactive computer programming environment

The adoption of Domain-Specific Languages (DSLs) relies on the capacity of language workbenches to automate the development of advanced and customized environments. While DSLs are usually well tailored for the main scenarios, the cost of developing mature tools prevents the ability to develop additional capabilities for alternative scenarios targeting specific tasks (e.g., API testing) or stakeholders (e.g., education). In [47], we propose an approach to automatically generate interactive computer programming environments from existing specifications of textual interpreted DSLs. The approach provides abstractions to complement the DSL specification, and combines static analysis and language transformations to automate the transformation of the language syntax, the execution state and the execution semantics. We evaluate the approach over a representative set of DSLs, and demonstrate the ability to automatically transform a textual syntax to load partial programs limited to a single statement, and to derive a Read-Eval-Print-Loop (REPL) from the specification of a language interpreter.

6.2.5. Live-UMLRT: A Tool for Live Modeling of UML-RT Models

In the context of Model-driven Development (MDD) models can be executed by interpretation or by the translation of models into existing programming languages, often by code generation. In [42] we present Live-UMLRT, a tool that supports live modeling of UML-RT models when they are executed by code generation. Live-UMLRT is entirely independent of any live programming support offered by the target language. This

independence is achieved with the help of a model transformation which equips the model with support for, e.g., debugging and state transfer both of which are required for live modeling. A subsequent code generation then produces a self-reflective program that allows changes to the model elements at runtime (through synchronization of design and runtime models). We have evaluated Live-UMLRT on several use cases. The evaluation shows that (1) code generation, transformation, and state transfer can be carried out with reasonable performance, and (2) our approach can apply model changes to the running execution faster than the standard approach that depends on the live programming support of the target language. A demonstration video: https://youtu.be/6GrR-Y9je7Y.

6.2.6. Applying model-driven engineering to high-performance computing: Experience report, lessons learned, and remaining challenges

In [35], we present a framework for generating optimizing compilers for performance-oriented embedded DSLs (EDSLs). This framework provides facilities to automatically generate the boilerplate code required for building DSL compilers on top of the existing extensible optimizing compilers. We evaluate the practicality of our framework by demonstrating a real-world use-case successfully built with it.

6.2.7. Software languages in the wild (Wikipedia)

Wikipedia is a rich source of information across many knowledge domains. Yet, recovering articles relevant to a specific domain is a difficult problem since such articles may be rare and tend to cover multiple topics. Furthermore, Wikipedia's categories provide an ambiguous classification of articles as they relate to all topics and thus are of limited use. In [46], we develop a new methodology to isolate Wikipedia's articles that describe a specific topic within the scope of relevant categories; the methodology uses super- vised machine learning to retrieve a decision tree classifier based on articles' features (URL patterns, summary text, infoboxes, links from list articles). In a case study, we retrieve 3000+ articles that describe software (computer) languages. Available fragments of ground truths serve as an essential part of the training set to detect relevant articles. The results of the classification are thoroughly evaluated through a survey, in which 31 domain experts participated.

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 run time.

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

In Kevoree, one of the goal is to work on the shipping phases 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 run time to improve resource usage in two domains: cloud computing [30], [57] and energy [58].

6.3.2. Investigating Machine Learning Algorithms for Modeling SSD I/O Performance for Container-based Virtualization

One of the cornerstones of the cloud provider business is to reduce hardware resources cost by maximizing their utilization. This is done through smartly sharing processor, memory, network and storage, while fully satisfying SLOs negotiated with customers. For the storage part, while SSDs are increasingly deployed in data centers mainly for their performance and energy efficiency, their internal mechanisms may cause a dramatic SLO violation. In effect, we measured that I/O interference may induce a 10x performance drop. We are building a framework based on autonomic computing which aims to achieve intelligent container placement on storage systems by preventing bad I/O interference scenarios. One prerequisite to such a framework is to design SSD performance models that take into account interactions between running processes/containers,

the operating system and the SSD. These interactions are complex. In this work [30], we investigate the use of machine learning for building such models in a container based Cloud environment. We have investigated five popular machine learning algorithms along with six different I/O intensive applications and benchmarks. We analyzed the prediction accuracy, the learning curve, the feature importance and the training time of the tested algorithms on four different SSD models. Beyond describing modeling component of our framework, this paper aims to provide insights for cloud providers to implement SLO compliant container placement algorithms on SSDs. Our machine learning-based framework succeeded in modeling I/O interference with a median Normalized Root-Mean-Square Error (NRMSE) of 2.5%.

6.3.3. Cuckoo: Opportunistic MapReduce on Ephemeral and Heterogeneous Cloud Resources

Cloud infrastructures are generally over-provisioned for handling load peaks and node failures. However, the drawback of this approach is that a large portion of data center resources remains unused. In this work [57], we propose a framework that leverages unused resources of data centers, which are ephemeral by nature, to run MapReduce jobs. Our approach allows: i) to run efficiently Hadoop jobs on top of heterogeneous Cloud resources, thanks to our data placement strategy, ii) to predict accurately the volatility of ephemeral resources, thanks to the quantile regression method, and iii) for avoiding the interference between MapReduce jobs and co-resident workloads, thanks to our reactive QoS controller. We have extended Hadoop implementation with our framework and evaluated it with three different data center workloads. The experimental results show that our approach divides Hadoop job execution time by up to 7 when compared to the standard Hadoop implementation. In [44], we presented a demo that leverages unused but volatile Cloud resources to run big data jobs. It is based on a learning algorithm that accurately predicts future availability of resources to automatically scale the ran jobs. We also designed a mechanism that avoids interference between the Big data jobs and co-resident workloads. Our solution is based on Open-Source components such as kubernetes and Apache Spark.

6.3.4. Leveraging cloud unused resources for Big data application while achieving SLA

Companies are more and more inclined to use collaborative cloud resources when their maximum internal capacities are reached in order to minimize their TCO. The downside of using such a collaborative cloud, made of private clouds' unused resources, is that malicious resource providers may sabotage the correct execution of third-party-owned applications due to its uncontrolled nature. In this work [43], we propose an approach that allows sabotage detection in a trustless environment. To do so, we designed a mechanism that (1) builds an application fingerprint considering a large set of resources usage (such as CPU, I/O, memory) in a trusted environment using random forest algorithm, and (2) an online remote fingerprint recognizer that monitors application execution and that makes it possible to detect unexpected application behavior. Our approach has been tested by building the fingerprint of 5 applications on trusted machines. When running these applications on untrusted machines (with either homogeneous, heterogeneous or unspecified hardware from the one that was used to build the model), the fingerprint recognizer was able to ascertain whether the execution of the application is correct or not with a median accuracy of about 98% for heterogeneous hardware and about 40% for the unspecified one.

6.3.5. Benefits of Energy Management Systems on local energy efficiency, an agricultural case study

Energy efficiency is a concern impacting both ecology and economy. Most approaches aiming at reducing the energy impact of a site focus on only one specific aspect of the ecosystem: appliances, local generation or energy storage. A trade-off analysis of the many factors to consider is challenging and must be supported by tools. This work proposes a Model-Driven Engineering approach mixing all these concerns into one comprehensive model [58]. This model can then be used to size either local production means, either energy storage capacity and also help to analyze differences between technologies. It also enables process optimization by modeling activity variability: it takes the weather into account to give regular feedback to the end user. This approach is illustrated by simulation using real consumption and local production data from a representative agricultural site. We show its use by: sizing solar panels, by choosing between battery

technologies and specification and by evaluating different demand response scenarios while examining the economic sustainability of these choices.

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. A Collaborative Strategy for Mitigating Tracking through Browser Fingerprinting

Browser fingerprinting is a technique that collects information about the browser configuration and the environment in which it is running. This information is so diverse that it can partially or totally identify users online. Over time, several countermeasures have emerged to mitigate tracking through browser fingerprinting. However, these measures do not offer full coverage in terms of privacy protection, as some of them may introduce inconsistencies or unusual behaviors, making these users stand out from the rest. In this work [45], we address these limitations by proposing a novel approach that minimizes both the identifiability of users and the required changes to browser configuration. To this end, we exploit clustering algorithms to identify the devices that are prone to share the same or similar fingerprints and to provide them with a new non-unique fingerprint. We then use this fingerprint to automatically assemble and run web browsers through virtualization within a docker container. Thus all the devices in the same cluster will end up running a web browser with an indistinguishable and consistent fingerprint.

6.4.2. Software Testing

6.4.2.1. A Snowballing Literature Study on Test Amplification

The adoption of agile development approaches has put an increased emphasis on developer testing, resulting in software projects with strong test suites. These suites include a large number of test cases, in which developers embed knowledge about meaningful input data and expected properties in the form of oracles. This work [29] surveys various works that aim at exploiting this knowledge in order to enhance these manually written tests with respect to an engineering goal (e.g., improve coverage of changes or increase the accuracy of fault localization). While these works rely on various techniques and address various goals, we believe they form an emerging and coherent field of research, which we call ‘test amplification’. We devised a first set of papers from DBLP, looking for all papers containing ‘test’ and ‘amplification’ in their title. We reviewed the 70 papers in this set and selected the 4 papers that fit our definition of test amplification. We use these 4 papers as the seed for our snowballing study, and systematically followed the citation graph. This study is the first that draws a comprehensive picture of the different engineering goals proposed in the literature for test amplification. In particular, we note that the goal of test amplification goes far beyond maximizing coverage only. We believe that this survey will help researchers and practitioners entering this new field to understand more quickly and more deeply the intuitions, concepts and techniques used for test amplification.

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, 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. In this work [28], 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. Leveraging metamorphic testing to automatically detect inconsistencies in code generator families

Generative software development has paved the way for the creation of multiple code generators that serve as a basis for automatically generating code to different software and hardware platforms. In this context, the software quality becomes highly correlated to the quality of code generators used during software development. Eventual failures may result in a loss of confidence for the developers, who will unlikely continue to use these generators. It is then crucial to verify the correct behaviour of code generators in order to preserve software quality and reliability. In this work [25], we leverage the metamorphic testing approach to automatically detect inconsistencies in code generators via so-called "metamorphic relations". We define the metamorphic relation (i.e., test oracle) as a comparison between the variations of performance and resource usage of test suites running on different versions of generated code. We rely on statistical methods to find the threshold value from which an unexpected variation is detected. We evaluate our approach by testing a family of code generators with respect to resource usage and performance metrics for five different target software platforms. The experimental results show that our approach is able to detect, among 95 executed test suites, 11 performance and 15 memory usage inconsistencies.

6.4.3. Software Co-evolution

6.4.3.1. An Empirical Study on the Impact of Inconsistency Feedback during Model and Code Co-changing

Model and code co-changing is about the coordinated modification of models and code during evolution. Intermittent inconsistencies are a common occurrence during co-changing. A partial co-change is the period in which the developer changed, say, the model but has not yet propagated the change to the code. Inconsistency feedback can be provided to developers for helping them to complete partial co-changes. However, there is no evidence whether such inconsistency feedback is useful to developers. To investigate this problem, we conducted a controlled experiment with 36 subjects who were required to complete ten partially completed change tasks between models and code of two non-trivial systems [31]. The tasks were of different levels of complexity depending on how many model diagrams they affected. All subjects had to work on all change tasks but sometimes with and sometimes without inconsistency feedback. We then measured differences between task effort and correctness. We found that when subjects were given inconsistency feedback during tasks, they were 268% more likely to complete the co-change correctly compared to when they were not given inconsistency feedback. We also found that when subjects were not given inconsistency feedback, they nearly always failed in completing co-change tasks with high complexity where the partially completed changes were spread across different diagrams in the model. These findings suggest that inconsistency feedback (i.e. detection and repair) should form an integral part of co-changing, regardless of whether the code or the model changes first. Furthermore, these findings suggest that merely having access to changes (as with the given partially completed changes) is insufficient for effective co-changing.

6.4.3.2. Detecting and Exploring Side Effects when Repairing Model Inconsistencies

When software models change, developers often fail in keeping them consistent. Automated support in repairing inconsistencies is widely addressed. Yet, merely enumerating repairs for developers is not enough. A repair can as a side effect cause new unexpected inconsistencies (negative) or even fix other inconsistencies as well (positive). To make matters worse, repairing negative side effects can in turn cause further side effects. Current approaches do not detect and track such side effects in depth, which can increase developers' effort and time spent in repairing inconsistencies. This work [66] presents an automated approach for detecting and tracking the consequences of repairs, i.e. side effects. It recursively explores in depth positive and negative side effects and identifies paths and cycles of repairs. This work further ranks repairs based on side effect knowledge so that developers may quickly find the relevant ones. Our approach and its tool implementation

have been empirically assessed on 14 case studies from industry, academia, and GitHub. Results show that both positive and negative side effects occur frequently. A comparison with three versioned models showed the usefulness of our ranking strategy based on side effects. It showed that our approach's top prioritized repairs are those that developers would indeed choose. A controlled experiment with 24 participants further highlights the significant influence of side effects and of our ranking of repairs on developers. Developers who received side effect knowledge chose far more repairs with positive side effects and far less with negative side effects, while being 12.3% faster, in contrast to developers who did not receive side effect knowledge.

6.4.3.3. Supporting A Flexible Grouping Mechanism for Collaborating Engineering Teams

Most engineering tools do not provide much support for collaborating teams and today's engineering knowledge repositories lack flexibility and are limited. Engineering teams have different needs and their team members have different preferences on how and when to collaborate. These needs may depend on the individual work style, the role an engineer has, and the tasks they have to perform within the collaborating group. However, individual collaboration is insufficient and engineers need to collaborate in groups. This work [65] presents a collaboration framework for collaborating groups capable of providing synchronous and asynchronous mode of collaboration. Additionally, our approach enables engineers to mix these collaboration modes to meet the preferences of individual group members. We evaluate the scalability of this framework using four real life large collaboration projects. These projects were found from GitHub and they were under active development by the time of evaluation. We have tested our approach creating groups of different sizes for each project. The results showed that our approach scales to support every case for the groups created. Additionally, we scouted the literature and discovered studies that support the usefulness of different groups with collaboration styles.

6.4.4. Software diversification

6.4.4.1. The Maven Dependency Graph: a Temporal Graph-based Representation of Maven Central

The Maven Central Repository provides an extraordinary source of data to understand complex architecture and evolution phenomena among Java applications. As of September 6, 2018, this repository includes 2.8M artifacts (compiled piece of code implemented in a JVM-based language), each of which is characterized with metadata such as exact version, date of upload and list of dependencies towards other artifacts. Today, one who wants to analyze the complete ecosystem of Maven artifacts and their dependencies faces two key challenges: (i) this is a huge data set; and (ii) dependency relationships among artifacts are not modeled explicitly and cannot be queried. In this work [55], we present the Maven Dependency Graph. This open source data set provides two contributions: a snapshot of the whole Maven Central taken on September 6, 2018, stored in a graph database in which we explicitly model all dependencies; an open source infrastructure to query this huge dataset.

6.4.4.2. The Emergence of Software Diversity in Maven Central

Maven artifacts are immutable: an artifact that is uploaded on Maven Central cannot be removed nor modified. The only way for developers to upgrade their library is to release a new version. Consequently, Maven Central accumulates all the versions of all the libraries that are published there, and applications that declare a dependency towards a library can pick any version. In this work [59], we hypothesize that the immutability of Maven artifacts and the ability to choose any version naturally support the emergence of software diversity within Maven Central. We analyze 1,487,956 artifacts that represent all the versions of 73,653 libraries. We observe that more than 30% of libraries have multiple versions that are actively used by latest artifacts. In the case of popular libraries, more than 50% of their versions are used. We also observe that more than 17% of libraries have several versions that are significantly more used than the other versions. Our results indicate that the immutability of artifacts in Maven Central does support a sustained level of diversity among versions of libraries in the repository.

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 a chaos engineering principles to IoT Services frameworks to improve the robustness of the software-defined network services using this approach and to explore the concept of equivalence for software-defined network services and propose an approach to constantly evolve the attack surface of the network services.

7.1.2. BCOM

- Coordinator: UR1
- Dates: 2018-2024
- Abstract: The aim 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 at working 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. GLOSE Demonstrator

- Partners: Inria/Safran
- Dates: 2019-2020
- Abstract: Demonstrator illustrating the technologies involved in the WP5 off the GLOSE project. The use case chosen for the demonstrator is the high-level description of a remote control drone system, whose the main objective is to illustrate the design and simulation of the main functional chains, the possible interactivity with the model in order to raise the level of understanding over the models built, and possibly the exploration of the design space.

7.1.5. 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 5 points of a software construction pipeline, which, once combined yield up to billions of variants. We also evaluate the support of diversification as a first class property in DevOps.

7.1.6. Kereval

- Partners: INSA Rennes/Kereval
- Dates: 2019-2022
- Abstract: Front-ends testing in a DevOps context, Romain Lebouc's PhD Cifre project.

7.1.7. Obeo

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

7.1.8. OKWind

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

7.1.9. Orange

- Partners: UR1/Orange
- Dates: 2016-2019
- Abstract: Modelling and evaluating security of authentication paths, Youssou Ndiaye's PhD Cifre project.

7.1.10. 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.

7.1.11. 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. 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 run time, 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. DGA

8.2.2.1. LangComponent (CYBERDEFENSE)

- Coordinator: DGA
- Partners: DGA MI, Inria
- Dates: 2019-2022
- Abstract: in the context of this project, DGA-MI and the Inria team DiverSE explore the existing approaches to ease the development of formal specifications of domain-Specific Languages (DSLs) dedicated to paquet filtering, while guaranteeing expressiveness, precision and safety. In the long term, this work is part of the trend to provide to DGA-MI and its partners a tooling to design and develop formal DSLs which ease the use while ensuring a high level of reasoning.

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

8.3.2. Collaborations with Major European Organizations

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

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

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

McGill University (Canada): language reuse, model composition, and models for sustainability.

CWI (The Netherlands): language engineering.

JKU Linz (Austria): model analysis and Model-Based DevOps.

RWTH Aachen (Germany): models for industry 4.0

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)
- University of Lancaster (UK)
- University of Namur (Belgium)
- Universita degli Studi di Cagliari (Italy)
- Università degli Studi dell'Aquila (Italy)
- JKU Linz (Austria)
- TU Wien (Austria)
- Michigan State University (MSU)
- RWTH 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 coordinate research partners worldwide to develop breakthrough software language engineering (SLE) approaches that support global software engineering through the use of multiple domain-specific languages. GEMOC members aim to provide effective SLE solutions to problems associated with the design and implementation of collaborative, interoperable and composable modeling languages.

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

The governance of the GEMOC initiative is provided 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 a GEMOC 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, July and October 2019.
- Nelly Bencomo, Lecturer in Computer Science Aston University, UK, visited the team from October 2019 to June 2020.
- Martin Montperrus, Professor at KTH, Sweden, visited the team in December 2019.
- Nicolas Harrand, PhD Student at KTH, Sweden, visited the team in December 2019.
- Paul Temple, postdoc at University de Namur, visited the team in February 2019.
- Thomas Degueule, postdoc at CWI, visited the team in December 2019
- Alfonso Pierantonio, Associate Professor at Università degli Studi dell'Aquila, visited the team in June 2019
- Mark van den Brand, Professor at Eindhoven University of Technology, visited the team in June 2019

8.5.2. Visits to International Teams

- Pierre JeanJean visited CWI for 1 week in December 2019 in the context of the Associated Team ALE.
- Benoit Combemale made several short visits at CWI in the context of the Associated Team ALE, visited McGill University in June 2019, and visited TU Eindhoven in November 2019.
- Olivier Barais made several short visits at KTH in the context of a collaboration with Prof Monperrus and Prof Baudry.
- Djamel E. Khelladi made a one week research visit in December 2019 to the DIRO laboratory at University of Montreal, Canada.

9. Dissemination

9.1. Promoting Scientific Activities

9.1.1. Scientific Events: Organisation

9.1.1.1. General Chair, Scientific Chair

- Mathieu Acher was the program committee co-chair of the 14th International Working Conference on Variability Modelling of Software-Intensive Systems (VaMoS)

9.1.1.2. Member of the Organizing Committees

Olivier Barais was the financial chair of the IEEE Mascots conference.

9.1.2. Scientific Events: Selection

9.1.2.1. Chair of Conference Program Committees

- Mathieu Acher organized the Fourth international workshop on software product line teaching (SPLTea 2019).
- Mathieu Acher organized the First international workshop on languages for modelling variability (MODEVAR 2019).
- Mathieu Acher organized the Seventh international workshop on reverse variability engineering (REVE 2019).
- Mathieu Acher organized the 3rd ACM SIGSOFT International Workshop on Machine Learning Techniques for Software Quality Evaluation, MaLTeSQuE@ESEC/SIGSOFT FSE 2019.

9.1.2.2. Member of the Conference Program Committees

Jean-Marc Jézéquel:

- PC member for SPLC 2019 Industry Track
- PC member for SEAMS 2019
- PC member for ICSE 2020

Olivier Barais:

- PC member for the 17th International Conference on Software Engineering and Formal Methods (SEFM 2019)
- PC Member for CIEL 2019
- PC Member for VaMoS 2019
- PC Member for VoSE 2019
- PC Member for SAC 2019, SATTA track - Software Architecture: Theory, Technology, and Applications
- PC Member for CCgrid 2020

Djamel E. Khelladi:

- PC member for Poster track at MODELS 2019
- PC member for REVE workshop 2019
- PC member for Models and Evolution (ME) Workshop 2019
- PC member for WETICE 2019 Validating Software for Critical Systems Track

Arnaud Blouin:

- PC member for SAC'2020
- PC member for PACM EICS Q4 2019
- PC member for Workshop on Human Factors in Modeling at MODELS'2019 (HuFaMo), 2019

Mathieu Acher:

- PC member for SPLC 2019 Research Track
- PC member for ICSE SEIP 2020
- PC member for VaMoS 2019

Benoit Combemale:

- PC chair (Foundation track) for ECMFA 2019
- PC member for ICMT'19
- Selection Committee member for the Doctoral Symposium at MODELS'19
- PC member for the MLE'19 workshop at MODELS'19
- PC member for the DevOps@MODELS'19 workshop at MODELS'19
- PC member for the FlexMDE'19 workshop at MODELS'19
- PC member for the PNSE'19 workshop

9.1.2.3. Reviewer

Olivier Barais: MASCOTS 2019, ICSE 2019, ASE 2019
Arnaud Blouin: ICSE 2019, ISSTA 2019
Johann Bourcier: SEAA 2019
Mathieu Acher: TSE 2019, JSS 2019, ICSE 2019

9.1.3. Journal

9.1.3.1. Member of the Editorial Boards

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 Combemale:

- Deputy Editor-in-Chief of JOT
- Member of the Editorial Board of the Springer Journal on Software and Systems Modeling (SoSyM)
- Member of the Editorial Board of the Elsevier Journal of Computer Languages (COLA)
- Member of the Editorial Board of the Springer Journal of Software Quality
- Member of the Advisory Board of the Elsevier Journal on Science of Computer Programming (SCP, Software Section)

9.1.3.2. Reviewer - Reviewing Activities

Olivier Barais:

- Journal of Software and System Modeling (SoSyM)

Djamel E. Khelladi:

- Journal of Software and System Modeling (SoSyM)

Arnaud Blouin:

- Journal of Software and System Modeling (SoSyM)
- Journal of Visual Languages and Computing (JVLC)
- Proceedings of the ACM for EICS 2019 and 2020 (PACM-EICS)

Mathieu Acher:

- Transactions on Software Engineering (TSE)
- Journal of Systems and Software (JSS)
- Editor with Myra B. Cohen of the Special issue on systems and software product line engineering. Journal of Systems and Software (JSS) 154: 110-111 (2019).

Benoit Combemale:

- Journal of Software and System Modeling (SoSyM)
- Journal of Empirical Software Engineering (ESE)

9.1.4. Invited Talks

Jean-Marc Jézéquel:

- *Keynote speech Models@Runtime workshop associated to MODELS2019*

Djamel E. Khelladi:

- Talk at the *IPA Fall Days* in Wageningen, Netherlands.

Johann Bourcier:

- Talk at the UBS university in Vannes, France.

Benoit Combemale:

- *Smart Modeling: On the Convergence of Scientific and Engineering Models*, Invited talk at Lancaster university, UK (14/11/19).
- *Bringing Intelligence to Sociotechnical IoT Systems: Modeling Opportunities and Challenges*, Keynote at MDDE4IoT'19, DE (15/09/19).
- *Breathe Life Into Your IDE*, Talk at LangDev'19, NL (22/03/19).

9.1.5. Leadership within the Scientific Community

Jean-Marc Jézéquel is an elected member of the Board of *InformaticsEurope*.

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.

Mathieu Acher:

- Member of the Steering committee of SPLC conference

Arnaud Blouin: founding member and co-organiser of the French GDR-GPL research action on Software Engineering and Human-Computer Interaction (GL-IHM).

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, and Benoit Combemale teaching about 100h, for a grand total of about 1300 hours, including several courses at ENSTB, Supelec, ENS Rennes 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 the head of the Information Technology department and member of the management board at the ESIR engineering school in Rennes. Arnaud Blouin is in charge of industrial relationships for the computer science department at INSA Rennes and elected member of this CS department council.

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 http://ejcp2019.icube.unistra.fr/

9.2.2. Supervision

- PhD in progress: Alejandro Gomez Boix, *Distributed counter-measure against browser fingerprinting*, 2016, B. Baudry, D. Bromberg.
- PhD in progress: Jean-Émile Dartois, *Efficient resources management for hybrid cloud computing*, 2016, O. Barais, Jalil Boukhobza.
- PhD in progress: Alexandre Rio, *Demand Side Management A model driven approach to promote energy self-consumption* , 2016, O. Barais, Y. Morel
- 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: 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: Alif Akbar-pranata, *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 in progress: Luc Lesoil, *Deep variability in large-scaled systems*, 2019, M. Acher. A. Blouin, JM Jézéquel.
- PhD in progress: Emmanuel Chebbi , *Domain-Specific Language Reuse*, 2019, B. Combemale, Olivier Barais, G. Leguernic
- PhD in progress: Mohamed Handaoui, *Scheduling Big Data applications in the cloud: the case of machine learning algorithms*, 2019, Jalil Boukhobza. Olivier Barais.
- PhD in progress: Romain Lebouc, *Testing front-ends in a DevOps context*, 2019, Arnaud Blouin, Noël Plouzeau, Alain Ribault.
- PhD : Manuel Leduc, *Formal and Executable Specification of domain-specific language families*, dec 2019, O. Barais, B. Combemale, G. Leguernic
- PhD : Youssou NDiaye, *Modelling and evaluating security of authentication paths*, dec 2019, N. Aillery, O. Barais, A. Blouin, A. Bouabdallah
- PhD: Ludovic Mouline, *Omniscient fault localization in IOT systems using model-driver data analytics*, nov 2019, O. Barais, Y. Le-Traon, J. Bourcier [23]
- PhD: Oscar Luis, *Automatic test amplification*, dec 2019, B. Baudry

- PhD: Manuel Leduc, *On modularity and performances of external domain-specific language implementations*, dec 2019, B. Combemale, O. Barais
- HdR : Arnaud Blouin, Contribution to the Engineering of User Interfaces, Univ Rennes, 30/08/2019 [22]

9.2.3. Juries

9.2.3.1. Jean-Marc Jézéquel

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

- Yoann Blein, April 2019, Univ Grenobles Alpes (Examiner)
- Jorge Roda, December 2019, Univ Sevilla (Examiner)
- Emilien Lavigne, December 2019, Univ Bretagne Occidentale (President)

9.2.3.2. Olivier Barais

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

- Benjamen Benni, December 2019, Univ Nice (Reviewer)
- Ludovic Mouline, November 2019, Univ Luxembourg (Supervisor)
- Leduc Manuel, December 2019, Univ Rennes (Supervisor)
- Ndiaye Youssou, December 2019, Univ Rennes (Supervisor)
- Davide Frey, December 2019, Univ Rennes (President)
- Hanyang CAO, February 2019, Univ Bordeaux (Reviewer)

9.2.3.3. Arnaud Blouin

was in the examination committee of the following PhD thesis:

- Youssou Ndiaye, December 2019, Univ Rennes (Supervisor)

9.2.3.4. Mathieu Acher

was in the examination committee of the following PhD thesis:

- Frederic Verdier, December 2019, Univ Montpellier (Examiner)

9.2.3.5. Johann Bourcier

was in the examination committee of the following PhD thesis:

- Ludovic Mouline, November 2019, Univ Luxembourg (Supervisor)

9.2.3.6. Benoit Combemale

was in the examination committee of the following PhD thesis:

- Théo Zimmermann, Univ Paris (December 2019), (Reviewer)
- Manuel Leduc, Univ Rennes 1 (December 2019), (Supervisor)

9.3. Popularization

9.3.1. Interventions

- STAMP - Orange Test and Dev Day, Grenoble, France - Nov 2019
- STAMP - Eastern Paris Dev Meetup, Lunatech, Chessy, France - Jul 2019
- STAMP Breakout Session & Talks - OW2con'19, Orange Gardens, Paris Châtillon - Jun 2019
- STAMP Workshop for EC-DGIT (European Commission DGIT) - Brussels - May 2019
- STAMP - Devoxx Paris - Apr 2019
- STAMP & CI testing automation - BreizhCamp 2019 - Mar 2019
- STAMP Talk and Workshop - Station-F, Paris - Feb 2019

- Tech talk at Alten - Rennes - Dec 2019
- Tech talk at Harmonic - Cesson-Sévigné - Oct 2019
- Tech talk at Veonum - Rennes - Sep 2019
- Tech talk at Nokia - Rennes - Jun 2019
- Tech talk at OrangeLabs : Hands-on Workshop - Cesson-Sévigné - May 2019
- Tech talk at Solocal and Kereval - Rennes - Mar 2019
- Tech talk at FaberNovel - Test amplification - Mar 2019
- Sciences de l'Ingénieur au féminin - Lycée Sévigné, Cesson-Sévigné - Nov 2019
- *J'peux pas j'ai informatique* - Inria, Rennes - Apr 2019
- *Petit-déjeuenr des métiers* - Lycée Sévigné, Cesson-Sévigné - Mar 2019 (je ne suis pas sure de la date)
- Devoxx 2019 - Les APIs hypermedia expliquées simplement (17-19 avril 2019) ("While web pages are full of hyperlinks and hypermedia, web APIs are not. However, augmenting web APIs with links is at the core of the REST architectural style, but only 5% of APIs actually do it. That's right, only 5%, despite most people saying that they provide RESTful APIs. In this talk, we discuss what an hypermedia-powered API looks like, what problems of distributed systems integration and composition it can solve and what it would change in frontend code to have such APIs.
- Embedded Linux conference Europe [54] – Given a configuration, can humans know in advance the size, the compilation time, or the boot time of a Linux kernel? Owing to the huge complexity of Linux (there are more than 15000 options with hard constraints and subtle interactions), machines should rather assist contributors and integrators in mastering the configuration space of the kernel. In this talk, we introduce TuxML an OSS tool based on Docker/Python to massively gather data about thousands of kernel configurations. Mathieu will describe how 200K+ configurations have been automatically built and how machine learning can exploit this information to predict properties of unseen Linux configurations, with different use cases (identification of influential/buggy options, finding of small kernels, etc.) The vision is that a continuous understanding of the configuration space is undoubtedly beneficial for the Linux community, yet several technical challenges remain in terms of infrastructure and automation.
- API Days - Backend is the new frontend (9-11 décembre 2019)

10. Bibliography

Major publications by the team in recent years

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

[2] 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

[3] 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

[4] M. BOUSSAA, O. BARAIS, G. SUNYÉ, B. BAUDRY. *Leveraging metamorphic testing to automatically detect inconsistencies in code generator families*, in "Software Testing, Verification and Reliability", December 2019 [*DOI :* 10.1002/STVR.1721], https://hal.inria.fr/hal-02422437

[5] 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

[6] 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

[7] 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

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

[9] J.-E. DARTOIS, J. BOUKHOBZA, A. KNEFATI, O. BARAIS. *Investigating Machine Learning Algorithms for Modeling SSD I/O Performance for Container-based Virtualization*, in "IEEE transactions on cloud computing", 2019, vol. 14, pp. 1-14 [*DOI :* 10.1109/TCC.2019.2898192], https://hal.inria.fr/hal-02013421

[10] 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

[11] 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

[12] 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

[13] 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

[14] 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 [*DOI :* 10.1007/S10664-018-9635-4], https://hal.inria.fr/hal-01829928

[15] 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° 2, pp. 905-920, https://hal.inria.fr/hal-00829839

[16] 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

[17] M. LEDUC, T. DEGUEULE, E. VAN WYK, B. COMBEMALE. *The Software Language Extension Problem*, in "Software and Systems Modeling", 2019, pp. 1-4, https://hal.inria.fr/hal-02399166

[18] 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

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

[20] P. TEMPLE, M. ACHER, J.-M. JÉZÉQUEL. *Empirical Assessment of Multimorphic Testing*, in "IEEE Transactions on Software Engineering", July 2019, pp. 1-21 [*DOI :* 10.1109/TSE.2019.2926971], https://hal.inria.fr/hal-02177158

[21] 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

Publications of the year

Doctoral Dissertations and Habilitation Theses

[22] A. BLOUIN. *Contribution to the Engineering of User Interfaces*, Université de Rennes 1 [UR1], August 2019, Habilitation à diriger des recherches, https://tel.archives-ouvertes.fr/tel-02354530

[23] L. MOULINE. *Towards a Modelling Framework with Temporal and Uncertain Data for Adaptive Systems*, Université Rennes 1 ; Université du Luxembourg, November 2019, https://hal.inria.fr/tel-02404304

Articles in International Peer-Reviewed Journals

[24] 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 2019, vol. 27, n° 1, pp. 307-347 [*DOI :* 10.1007/s11219-017-9400-8], https://hal.inria.fr/hal-01688247

[25] M. BOUSSAA, O. BARAIS, G. SUNYÉ, B. BAUDRY. *Leveraging metamorphic testing to automatically detect inconsistencies in code generator families*, in "Software Testing, Verification and Reliability", December 2019 [*DOI :* 10.1002/STVR.1721], https://hal.inria.fr/hal-02422437

[26] E. BOUSSE, T. MAYERHOFER, B. COMBEMALE, B. BAUDRY. *Advanced and efficient execution trace management for executable domain-specific modeling languages*, in "Software and Systems Modeling", February 2019, pp. 1–37 [*DOI :* 10.1007/s10270-017-0598-5], https://hal.inria.fr/hal-01614377

[27] J.-M. BRUEL, B. COMBEMALE, E. M. GUERRA, J.-M. JÉZÉQUEL, J. KIENZLE, J. DE LARA, G. MUSSBACHER, E. SYRIANI, H. VANGHELUWE. *Comparing and Classifying Model Transformation Reuse*

Approaches across Metamodels, in "Software and Systems Modeling", 2019, pp. 1-22, https://hal.inria.fr/hal-02317864

[28] B. DANGLOT, O. L. VERA-PÉREZ, B. BAUDRY, M. MONPERRUS. *Automatic Test Improvement with DSpot: a Study with Ten Mature Open-Source Projects*, in "Empirical Software Engineering", 2019, pp. 1-35 [*DOI :* 10.1007/S10664-019-09692-Y], https://hal.inria.fr/hal-01923575

[29] B. DANGLOT, O. L. VERA-PÉREZ, Z. YU, A. ZAIDMAN, M. MONPERRUS, B. BAUDRY. *A Snowballing Literature Study on Test Amplification*, in "Journal of Systems and Software", August 2019, vol. 157, pp. 1-16 [*DOI :* 10.1016/J.JSS.2019.110398], https://hal.inria.fr/hal-02290742

[30] J.-E. DARTOIS, J. BOUKHOBZA, A. KNEFATI, O. BARAIS. *Investigating Machine Learning Algorithms for Modeling SSD I/O Performance for Container-based Virtualization*, in "IEEE transactions on cloud computing", 2019, vol. 14, pp. 1-14 [*DOI :* 10.1109/TCC.2019.2898192], https://hal.inria.fr/hal-02013421

[31] G. KANAKIS, D. E. KHELLADI, S. FISCHER, M. TRÖLS, A. EGYED. *An Empirical Study on the Impact of Inconsistency Feedback during Model and Code Co-changing*, in "The Journal of Object Technology", 2019, vol. 18, n° 2, pp. 10:1-21 [*DOI :* 10.5381/JOT.2019.18.2.A10], https://hal.inria.fr/hal-02192486

[32] J. KIENZLE, G. MUSSBACHER, B. COMBEMALE, L. BASTIN, N. BENCOMO, J.-M. BRUEL, C. BECKER, S. BETZ, R. CHITCHYAN, B. CHENG, S. KLINGERT, R. PAIGE, B. PENZENSTADLER, N. SEYFF, E. SYRIANI, C. C. VENTERS. *Towards Model-Driven Sustainability Evaluation*, in "Communications of the ACM", 2019, pp. 1-10, https://hal.inria.fr/hal-02146543

[33] J. KIENZLE, G. MUSSBACHER, B. COMBEMALE, J. DEANTONI. *A Unifying Framework for Homogeneous Model Composition*, in "Software and Systems Modeling", January 2019, pp. 1-19 [*DOI :* 10.1007/S10270-018-00707-8], https://hal.inria.fr/hal-01949050

[34] M. LEDUC, T. DEGUEULE, E. VAN WYK, B. COMBEMALE. *The Software Language Extension Problem*, in "Software and Systems Modeling", 2019, pp. 1-4, https://hal.inria.fr/hal-02399166

[35] B. LELANDAIS, M.-P. OUDOT, B. COMBEMALE. *Applying Model-Driven Engineering to High-Performance Computing: Experience Report, Lessons Learned, and Remaining Challenges*, in "Computer Languages, Systems and Structures", 2019, pp. 1-19, https://hal.inria.fr/hal-02296030

[36] A. PIERANTONIO, M. VAN DEN BRAND, B. COMBEMALE. *The JOT Journal: Towards a Rising Generation*, in "Journal of Object Technology (JOT)", 2019, vol. 18, n° 1, pp. 1-3 [*DOI :* 10.5381/JOT.2019.18.1.E1], https://hal.inria.fr/hal-02408017

[37] P. TEMPLE, M. ACHER, J.-M. JÉZÉQUEL. *Empirical Assessment of Multimorphic Testing*, in "IEEE Transactions on Software Engineering", July 2019, pp. 1-21, forthcoming [*DOI :* 10.1109/TSE.2019.2926971], https://hal.inria.fr/hal-02177158

[38] A. WORTMANN, O. BARAIS, B. COMBEMALE, M. WIMMER. *Modeling Languages in Industry 4.0: An Extended Systematic Mapping Study*, in "Software and Systems Modeling", 2019, pp. 1-28 [*DOI :* 10.1007/S10270-019-00757-6], https://hal.inria.fr/hal-02282028

Invited Conferences

[39] B. COMBEMALE. *Bringing Intelligence to Sociotechnical IoT Systems: Modeling Opportunities and Challenges*, in "MDE4IoT 2019 - 3nd International Workshop on Model-Driven Engineering for the Internet-of-Things", Munich, Germany, September 2019, pp. 1-2, https://hal.inria.fr/hal-02285737

International Conferences with Proceedings

[40] M. ACHER, T. ZIADI, R. E. LOPEZ-HERREJON, J. MARTINEZ. *Seventh international workshop on reverse variability engineering (REVE 2019)*, in "SPLC 2019 - 23rd International Systems and Software Product Line Conference", Paris, France, ACM Press, September 2019, 1 p. , https://hal.archives-ouvertes.fr/hal-02268373

[41] B. AMAND, M. CORDY, P. HEYMANS, M. ACHER, P. TEMPLE, J.-M. JÉZÉQUEL. *Towards Learning-Aided Configuration in 3D Printing: Feasibility Study and Application to Defect Prediction*, in "VAMOS 2019 - 13th International Workshop on Variability Modelling of Software-Intensive Systems", Leuven, Belgium, ACM, February 2019, pp. 1-9 [*DOI :* 10.1145/3302333.3302338], https://hal.inria.fr/hal-01990767

[42] M. BAGHERZADEH, K. JAHED, B. COMBEMALE, J. DINGEL. *Live-UMLRT: A Tool for Live Modeling of UML-RT Models*, in "MODELS 2019 - ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems", Munich, Germany, IEEE, September 2019, pp. 743-747 [*DOI :* 10.1109/MODELS-C.2019.00115], https://hal.inria.fr/hal-02407932

[43] J.-E. DARTOIS, J. BOUKHOBZA, V. FRANCOISE, O. BARAIS. *Tracking Application Fingerprint in a Trustless Cloud Environment for Sabotage Detection*, in "MASCOTS 2019 - 27th IEEE International Symposium on the Modeling, Analysis, and Simulation of Computer and Telecommunication Systems", Rennes, France, IEEE, October 2019, pp. 74-82 [*DOI :* 10.1109/MASCOTS.2019.00018], https://hal.archives-ouvertes.fr/hal-02303153

[44] J.-E. DARTOIS, I. MERIAU, M. HANDAOUI, J. BOUKHOBZA, O. BARAIS. *Leveraging cloud unused resources for Big data application while achieving SLA*, in "MASCOTS 2019 - 27th IEEE International Symposium on the Modeling, Analysis, and Simulation of Computer and Telecommunication Systems", Rennes, France, IEEE, October 2019, pp. 1-2, https://hal.inria.fr/hal-02362257

[45] A. GÓMEZ-BOIX, D. FREY, Y.-D. BROMBERG, B. BAUDRY. *A Collaborative Strategy for mitigating Tracking through Browser Fingerprinting*, in "MTD 2019 - 6th ACM Workshop on Moving Target Defense", London, United Kingdom, November 2019, pp. 1-12 [*DOI :* 10.1145/3338468.3356828], https://hal.inria.fr/hal-02282591

[46] M. HEINZ, R. LÄMMEL, M. ACHER. *Discovering Indicators for Classifying Wikipedia Articles in a Domain: A Case Study on Software Languages*, in "SEKE 2019 - The 31st International Conference on Software Engineering and Knowledge Engineering", Lisbonne, Portugal, July 2019, pp. 1-6, https://hal.inria.fr/hal-02129131

[47] P. JEANJEAN, B. COMBEMALE, O. BARAIS. *From DSL Specification to Interactive Computer Programming Environment*, in "SLE 2019 - 12th ACM SIGPLAN International Conference on Software Language Engineering", Athènes, Greece, ACM, October 2019, pp. 167-178 [*DOI :* 10.1145/3357766.3359540], https://hal.inria.fr/hal-02307953

[48] G. LE GUERNIC. *Experience Report on the Development of a Specialized Multi-view Multi-stakeholder Model-Based Engineering Framework*, in "DSM 2019 - 17th ACM SIGPLAN International Workshop on Domain-Specific Modeling", Athens, Greece, ACM Press, October 2019, pp. 50-59 [*DOI :* 10.1145/3358501.3361237], https://hal.inria.fr/hal-02398053

[49] H. MARTIN, J. A. PEREIRA, P. TEMPLE, M. ACHER. *Machine Learning and Configurable Systems: A Gentle Introduction*, in "SPLC 2019 - 23rd International Systems and Software Product Line Conference", Paris, France, ACM, September 2019, pp. 83-88 [*DOI :* 10.1145/3336294.3342383], https://hal.inria.fr/hal-02287459

[50] Y. NDIAYE, O. BARAIS, A. BLOUIN, A. BOUABDALLAH, N. AILLERY. *Requirements for preventing logic flaws in the authentication procedure of web applications*, in "SAC 2019 - 34th ACM/SIGAPP Symposium On Applied Computing", Limassol, Cyprus, April 2019, pp. 1-9 [*DOI :* 10.1145/3297280.3297438], https://hal.inria.fr/hal-02087663

[51] Q. PLAZAR, M. ACHER, G. PERROUIN, X. DEVROEY, M. CORDY. *Uniform Sampling of SAT Solutions for Configurable Systems: Are We There Yet?*, in "ICST 2019 - 12th IEEE International Conference on Software Testing, Verification, and Validation", Xian, China, IEEE, April 2019, pp. 240-251 [*DOI :* 10.1109/ICST.2019.00032], https://hal.inria.fr/hal-01991857

[52] M. RODRIGUEZ-CANCIO, B. COMBEMALE, B. BAUDRY. *Approximate Loop Unrolling*, in "CF 2019 - ACM International Conference on Computing Frontiers", Alghero, Sardinia, Italy, ACM, 2019, pp. 94-105 [*DOI :* 10.1145/3310273.3323841], https://hal.inria.fr/hal-02407868

[53] P. TEMPLE, M. ACHER, G. PERROUIN, B. BIGGIO, J.-M. JÉZÉQUEL, F. ROLI. *Towards Quality Assurance of Software Product Lines with Adversarial Configurations*, in "SPLC 2019 - 23rd International Systems and Software Product Line Conference", Paris, France, ACM, September 2019, pp. 277-288 [*DOI :* 10.1145/3336294.3336309], https://hal.inria.fr/hal-02287616

Conferences without Proceedings

[54] M. ACHER. *Learning the Linux Kernel Configuration Space: Results and Challenges*, in "ELC Europe 2019 - Embedded Linux Conference Europe 2019", Lyon, France, October 2019, pp. 1-49, https://hal.inria.fr/hal-02342130

[55] *Best Paper*
A. BENELALLAM, N. HARRAND, C. SOTO-VALERO, B. BAUDRY, O. BARAIS. *The Maven Dependency Graph: a Temporal Graph-based Representation of Maven Central*, in "MSR 2019 - 16th International Conference on Mining Software Repositories", Montreal, Canada, ACM, May 2019, pp. 344-348 [*DOI :* 10.1109/MSR.2019.00060], https://hal.archives-ouvertes.fr/hal-02080243.

[56] A. CHERON, J. BOURCIER, O. BARAIS, A. MICHEL. *Comparison Matrices of Semantic RESTful APIs Technologies*, in "ICWE 2019 - 19th International Conference On Web Engineering", Daejeon, South Korea, LNCS, Springer, June 2019, vol. 11496, pp. 425-440 [*DOI :* 10.1007/978-3-030-19274-7_30], https://hal.archives-ouvertes.fr/hal-02114296

[57] J.-E. DARTOIS, H. B. RIBEIRO, J. BOUKHOBZA, O. BARAIS. *Cuckoo: Opportunistic MapReduce on Ephemeral and Heterogeneous Cloud Resources*, in "CLOUD 2019 - IEEE 12th International Conference on Cloud Computing", Milan, Italy, IEEE, July 2019, pp. 1-8 [*DOI :* 10.1109/CLOUD.2019.00070], https://hal.archives-ouvertes.fr/hal-02179453

[58] A. RIO, Y. MAUREL, Y. BUGNI, O. BARAIS. *Benefits of Energy Management Systems on local energy efficiency, an agricultural case study*, in "SmartGridComm 2019 - IEEE International Conference on Commu-

nications, Control, and Computing Technologies for Smart Grids", Beijing, China, IEEE, October 2019, pp. 1-7, https://hal.archives-ouvertes.fr/hal-02315327

[59] C. SOTO-VALERO, A. BENELALLAM, N. HARRAND, O. BARAIS, B. BAUDRY. *The Emergence of Software Diversity in Maven Central*, in "MSR 2019 - 16th International Conference on Mining Software Repositories", Montreal, Canada, ACM, May 2019, pp. 333-343 [*DOI :* 10.1109/MSR.2019.00059], https://hal.archives-ouvertes.fr/hal-02080248

Scientific Books (or Scientific Book chapters)

[60] B. COMBEMALE, M. WIMMER. *Towards a Model-Based DevOps for Cyber-Physical Systems*, in "Software Engineering Aspects of Continuous Development", Springer-Verlag, 2019, pp. 1-11, https://hal.inria.fr/hal-02407886

Research Reports

[61] M. ACHER, H. MARTIN, J. ALVES PEREIRA, A. BLOUIN, D. EDDINE KHELLADI, J.-M. JÉZÉQUEL. *Learning From Thousands of Build Failures of Linux Kernel Configurations*, Inria ; IRISA, June 2019, pp. 1-12, https://hal.inria.fr/hal-02147012

[62] M. ACHER, H. MARTIN, J. A. PEREIRA, A. BLOUIN, J.-M. JÉZÉQUEL, D. E. KHELLADI, L. LESOIL, O. BARAIS. *Learning Very Large Configuration Spaces: What Matters for Linux Kernel Sizes*, Inria Rennes - Bretagne Atlantique, October 2019, https://hal.inria.fr/hal-02314830

[63] J. ALVES PEREIRA, H. MARTIN, M. ACHER, J.-M. JÉZÉQUEL, G. BOTTERWECK, A. VENTRESQUE. *Learning Software Configuration Spaces: A Systematic Literature Review*, Univ Rennes, Inria, CNRS, IRISA, June 2019, n^{o} 1-44, https://arxiv.org/abs/1906.03018 , https://hal.inria.fr/hal-02148791

[64] G. LE GUERNIC. *Experience Report on the Development of a Specialized Multi-view Multi-stakeholder Model-Based Engineering Framework (extended version)*, Inria Rennes - Bretagne Atlantique ; IRISA, December 2019, n^{o} RR-9283, https://hal.inria.fr/hal-02398051

Scientific Popularization

[65] G. KANAKIS, S. FISCHER, D. E. KHELLADI, A. EGYED. *Supporting A Flexible Grouping Mechanism for Collaborating Engineering Teams*, in "ICGSE 2019 - 14th ACM/IEEE International Conference on Global Software Engineering", Montreal, QC, Canada, IEEE, May 2019, pp. 129-138 [*DOI :* 10.1109/ICGSE.2019.00033], https://hal.inria.fr/hal-02192482

[66] D. E. KHELLADI, R. KRETSCHMER, A. EGYED. *Detecting and Exploring Side Effects when Repairing Model Inconsistencies*, in "SLE 2019 - 12th ACM SIGPLAN International Conference on Software Language Engineering", Athènes, Greece, ACM, October 2019, pp. 103-126 [*DOI :* 10.1145/3357766.3359546], https://hal.inria.fr/hal-02326034

Other Publications

[67] J. ALVES PEREIRA, M. ACHER, H. MARTIN, J.-M. JÉZÉQUEL. *Sampling Effect on Performance Prediction of Configurable Systems: A Case Study*, November 2019, working paper or preprint, https://hal.inria.fr/hal-02356290

[68] M. LEDUC, G. JOUNEAUX, T. DEGUEULE, G. LE GUERNIC, O. BARAIS, B. COMBEMALE. *Automatic generation of Truffle-based interpreters for Domain-Specific Languages*, December 2019, working paper or preprint, https://hal.inria.fr/hal-02395867

References in notes

[69] 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

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

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

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

[73] 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

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

[75] 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

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

[77] 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

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

[79] 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

[80] 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

[81] 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

[82] 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

[83] 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

[84] 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

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

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

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

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

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

[90] 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

[91] 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

[92] 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

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

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

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

[96] 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

[97] 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

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

[99] 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

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

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

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

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

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

[105] 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

[106] 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

[107] 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.

[108] 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

[109] 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

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

[111] 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

[112] 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° 2, pp. 128–145

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

[114] 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° 4, pp. 252–268

[115] 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

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

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

[118] 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

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

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

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

[122] 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

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

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

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

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

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

[128] 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

[129] 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

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