RESEARCH CENTRE

**Inria Centre at Rennes University**

**IN PARTNERSHIP WITH:**

**Institut national des sciences appliquées de Rennes, CNRS, Université de Rennes**

# 2023 ACTIVITY REPORT

# Project-Team DIVERSE

## Diversity-centric Software Engineering

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

**DOMAIN**

**Networks, Systems and Services, Distributed Computing**

**THEME**

**Distributed programming and Software engineering**

Inria

# Contents

# Project-Team DIVERSE

*Creation of the Project-Team: 2014 July 01*

## Keywords

### Computer sciences and digital sciences

A1.2.1. – Dynamic reconfiguration

A1.3.1. – Web

A1.3.5. – Cloud

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

A2.6.4. – Ressource management

A4.1.1. – Malware analysis

A4.4. – Security of equipment and software

A4.6. – Authentication

A4.7. – Access control

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 members, visitors, external collaborators

## Research Scientists

- Djamel Khelladi [CNRS, Researcher]
- Gunter Mussbacher [UNIV MCGILL, Advanced Research Position]
- Olivier Zendra [INRIA, Researcher, HDR]

## Faculty Members

- Olivier Barais [Team leader, UNIV RENNES, Professor, HDR]
- Mathieu Acher [INSA RENNES, Professor, HDR]
- Aymeric Blot [UNIV RENNES, Associate Professor, from Sep 2023]
- Arnaud Blouin [INSA RENNES, Associate Professor, HDR]
- Johann Bourcier [UNIV RENNES, Associate Professor, HDR]
- Stéphanie Challita [UNIV RENNES, Associate Professor]
- Benoît Combemale [UNIV RENNES, Professor, HDR]
- Jean-Marc Jezequel [UNIV RENNES, Professor, HDR]
- Quentin Perez [INSA RENNES, Associate Professor, from Sep 2023]
- Noël Plouzeau [UNIV RENNES, Associate Professor]
- Walter Rudametkin Ivey [UNIV RENNES, Professor, HDR]
- Paul Temple [UNIV RENNES, Associate Professor]

## Post-Doctoral Fellows

- Gwendal Jouneaux [UNIV RENNES, from Sep 2023]
- Faezeh Khorram [CNRS, Post-Doctoral Fellow, until Mar 2023]
- Quentin Perez [UNIV RENNES, until Aug 2023]
- Xhevahire Ternava [UNIV RENNES, Post-Doctoral Fellow, until Aug 2023]

## PhD Students

- Lina Bilal [UNIV RENNES, from Oct 2023]
- Ewen Brune [INRIA, from Oct 2023]
- Anne Bumiller [ORANGE, until Sep 2023]
- Theo Giraudet [OBEO, CIFRE]
- Philemon Houdaille [CNRS, from Sep 2023]
- Gwendal Jouneaux [UNIV RENNES, until Aug 2023]
- Zohra Kebaili [CNRS, from Jan. 2022]
- N'Guessan Hermann Kouadio [CGI, CIFRE, from Dec 2023]

- Piergiorgio Ladisa [SAP, CIFRE]
- Clement Lahoche [INRIA, from Dec 2023]
- Quentin Le Dilavrec [UNIV RENNES, until Oct 2023]
- Romain Lefeuvre [UNIV RENNES, from Nov 2023]
- Georges Aaron Randrianaina [UNIV RENNES ]
- Chiara Relevat [UNIV RENNES, from Sep 2023]
- Sterenn Roux [UNIV RENNES, from Oct 2023]

### Technical Staff

- Florian Badie [INRIA, Engineer, until Apr 2023]
- Romain Belafia [UNIV RENNES, Engineer, until Aug 2023]
- Emmanuel Chebbi [INRIA, Engineer]
- Guy De Spiegeleer [UNIV RENNES, Engineer, until Aug 2023]
- Quentin Le Dilavrec [INRIA, Engineer, from Nov 2023]
- Romain Lefeuvre [INRIA, Engineer, until Oct 2023]
- Charly Reux [INRIA, Engineer, from Oct 2023]
- Didier Vojtisek [INRIA, Engineer]

### Interns and Apprentices

- Paul Adam [ENS RENNES, Intern, from May 2023 until Jul 2023]
- Arthur Allain [UNIV RENNES, Intern, from May 2023 until Aug 2023]
- Yazid Benjamaa [UNIV RENNES, Intern, from Jun 2023 until Sep 2023]
- Jeremy Bindel [UNIV RENNES, Intern, from Jun 2023 until Sep 2023]
- Jean-Baptiste Doderlein [ENS RENNES, Intern, from Mar 2023 until May 2023]
- Philemon Houdaille [INRIA, Intern, until Jul 2023]
- Margaux Millour [UNIV RENNES, Intern, from Jun 2023 until Jul 2023]
- Yawa Germaine Nyatsikor [UNIV RENNES, Intern, from Sep 2023]
- Benjamin Ramone [Univ Rennes, from May 2023 until Aug 2023]
- Bastien Sauvat [UNIV RENNES, Intern, from Jun 2023 until Sep 2023]
- Abdullah Sen [UNIV RENNES, Intern, from May 2023 until Jul 2023]
- Cyriaque Tossou [UNIV RENNES, Intern, from May 2023 until Aug 2023]

### Administrative Assistant

- Sophie Maupile [CNRS]

### External Collaborator

- Gurvan Le Guernic [DGA]

## 2 Overall objectives

DIVERSE's research agenda targets core values of software engineering. In this fundamental domain we focus on 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 borne 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 services). Today these systems seem to be all radically different, but we envision a strong convergence of the scientific principles that underpin their construction and validation, bringing forwards sane and reliable methods for the design of **flexible and open yet dependable systems**. Flexibility and openness are 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**, where software has to run and adapted; **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 as well as in its environment. Conversely, explicitly handling diversity is a great opportunity to allow software to spontaneously explore alternative design solutions, and to mitigate security risks.

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 correctness envelopes in an opportunistic manner);
- to dynamically synthesize software inside a given envelope.

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 Context

Applications are becoming more complex and the demand for faster development is increasing. In order to better adapt to the unbridled evolution of requirements in markets where software plays an essential role, companies are changing the way they design, develop, secure and deploy applications, by relying on:

- A massive use of reusable libraries from a rich but fragmented eco-system;

- An increasing configurability of most of the produced software;
- A strongly increase in evolution frequency;
- Cloud-native architectures based on containers, naturally leading to a diversity of programming languages used, and to the emergence of infrastructure, dependency, project and deployment descriptors (models);
- Implementations of fully automated software supply chains;
- The use of lowcode/nocode platforms;
- The use of ever richer integrated development environments (IDEs), more and more deployed in SaaS mode;
- The massive use of data and artificial intelligence techniques in software production chains.

These trends are set to continue, all the while with a strong concern about the security properties of the produced and distributed software.
The numbers in the examples below help to understand why this evolution of modern software engineering brings a **change of dimension**:

- When designing a simple kitchen sink (*hello world*) with the `angular` framework, more than 1600 dependencies of JavaScript libraries are pulled.
- The numbers revealed by Google in 2018 showed that over 500 million tests are run *per day* inside Google's systems, leading to over 4 millions daily builds.
- Also at Google, they reported 86 TB of data, including two billion lines of code in nine million source files [127]. Their software also rapidly evolves both in terms of frequency and in terms of size. Again, at Google, 25,000 developers typically commit 16,000 changes to the codebase on a single workday. This is also the case for most of software code, including open source software.
- x264, a highly popular and configurable video encoder, provides 100+ options that can take boolean, integer or string values. There are different ways of compiling x264, and it is well-known that the compiler options (e.g., -O1 –O2 –O3 of gcc) can influence the performance of a software; the widely used gcc compiler, for example, offers more than 200 options. The x264 encoder can be executed on different configurations of the Linux operating system, whose options may in turn influence x264 execution time; in recent versions (> 5), there are 16000+ options to the Linux kernel. Last but not least, x264 should be able to encode many different videos, in different formats and with different visual properties, implying a huge variability of the input space. Overall, the variability space is enormous, and ideally x264 should be run and tested in all these settings. But a rough estimation shows that the number of possible configurations, resulting from the combination of the different variability layers, is $10^{6000}$.

The DIVERSE research project is working and evolving in the context of this acceleration. We are active at all stages of the **software supply chain**. Software supply chain covers all the activities and all the stakeholders that relate to software production and delivery. All these activities and stakeholders have to be smartly managed together as part of an overall strategy. The goal of supply chain management (SCM) is to meet customer demands with the most efficient use of resources possible.

In this context, DIVERSE is particularly interested in the following research questions:

- How to engineer tool-based abstractions for a given set of experts in order to foster their socio-technical collaboration;
- How to generate and exploit useful data for the optimization of this supply chain, in particular for the control of variability and the management of the co-evolution of the various software artifacts;
- How to increase the confidence in the produced software, by working on the resilience and security of the artifacts produced throughout this supply chain.

## 3.2 Scientific background

### 3.2.1 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) [131]. It provides DIVERSE with solid foundations to specify, analyze and reason about the different forms of diversity that occur throughout the development life cycle. 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 [103]. 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 [86]. 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*, including a unified typing theory that integrates models as first class entities [133].

Incorporating domain-specific concepts and a 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 cyber-physical systems. A study performed by Hutchinson et al. [108] 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 the 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 the 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 [117].

### 3.2.2 Variability modeling

While the basic vision underlying *Software Product Lines* (SPL) can probably be traced back to David Parnas' seminal article [124] on the Design and Development of Program Families, it is only quite recently that SPLs have started emerging as a paradigm shift towards modeling and developing software system families rather than individual systems [121]. 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* [122]. Bosch provides a different definition [92]: *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 [98]. Instead of describing a single software system, a SPL model describes a

set of products in the same domain. This is accomplished by distinguishing between elements common to all SPL members, and those that may vary from one product to another. Reuse of core assets, which form the basis of the product line, is key to productivity and quality gains. These core assets extend beyond simple code reuse and may include the architecture, software components, domain models, requirements statements, documentation, test plans or test cases.

The SPL engineering process consists of two major steps:

1. **Domain Engineering**, or *development for reuse*, focuses on core assets development.

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

Central to both processes is the management of **variability** across the product line [105]. 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 [93]. Variability management is thus increasingly seen as the cornerstone of SPL development, covering the entire development life cycle, from requirements elicitation [135] to product derivation [139] to product testing [120, 119].

Halmans *et al.* [105] distinguish between *essential* and *technical* variability, especially at the 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.* [126]: 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.* [114] (feature, hardware platform, performance and attributes variability) or Bass *et al.* [84] 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* [110]. 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 [93]. 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 [88]. Model Driven Engineering (MDE) makes it possible to automate this weaving process [109]. 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 fact an executable meta-model [118]) manipulating these models to produce for instance a detailed design that can ultimately be transformed to code, or to test suites [125], or other software artifacts.

### 3.2.3 Component-based software development

Component-based software development [134] 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 [99]: description of requirements on services required from the other components; indirect connections between components thanks to ports and connectors constructs [112]; hierarchical definition of components (assemblies of components can define new component types); connectors supporting various communication semantics [96]; quantitative properties on the services [91].

In recent years component-based architectures have evolved from static designs to dynamic, adaptive designs (e.g., SOFA [96], Palladio [89], Frascati [115]). 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 processes support *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 [111]. 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 [97, 137]. 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 [90], 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 [129]". 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 systems. 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 envelope at run-time. 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 [104]. Multi Objectives Search based techniques [100] 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.

#### 3.2.4 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 envelope. 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 envelope 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 [136] mainly relies on model analysis, constraint solving [101] and search-based reasoning [113]. DIVERSE leverages in particular the applications of model-based testing in the context of highly-configurable systems and [138] interactive systems [116] as well as recent advances based on diversity for test cases selection [107].

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 [123], or Xholon. 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 [95] 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 [85] provides modeling elements amenable to semantically equivalent mathematical models. Metropolis offers a precise semantics flexible enough to support different models of computation. ModHel'X [106] 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 [83], recovery blocks [128] and code randomization [87], 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.

#### 3.2.5 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 [132, 130]. Such methods have been used for example to understand the impact of new software development paradigms [94]. 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 [82].

### 3.3 Research axis

DIVERSE explore *Software Diversity*. Leveraging our strong background on Model-Driven Engineering, and our large expertise on several related fields (programming languages, distributed systems, GUI, machine learning, security...), *we explore tools and methods to embrace the inherent diversity in software engineering*, from the stakeholders and underlying tool-supported languages involved in the software system life cycle, to the configuration and evolution space of the modern software systems, and the heterogeneity of the targeted execution platforms. Hence, we organize our research directions according to three axes (cf. Fig. 1):

- **Axis #1: Software Language Engineering.** We explore the future engineering and scientific environments to support the socio-technical coordination among the various stakeholders involved across

modern software system life cycles.

- **Axis #2: Spatio-temporal Variability in Software and Systems.** We explore systematic and automatic approaches to cope with software variability, both in space (software variants) and time (software maintenance and evolution).
- **Axis #3: DevSecOps and Resilience Engineering for Software and Systems.** We explore smart continuous integration and deployment pipelines to ensure the delivery of secure and resilient software systems on heterogeneous execution platforms (cloud, IoT...).

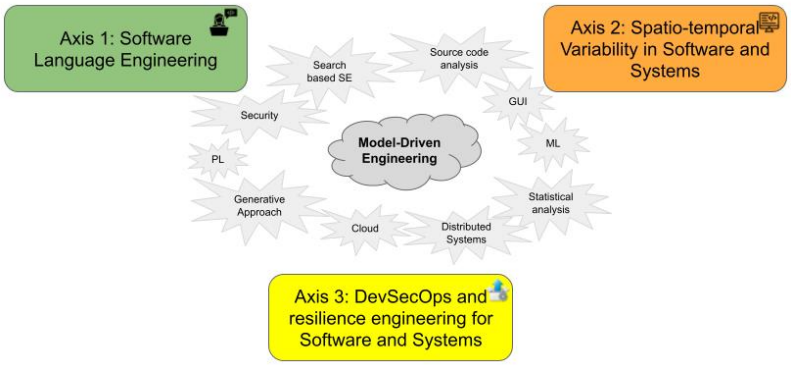


Figure 1: The three research axes of DIVERSE, relying on model driven engineering scientific background and leveraging several related fields

### 3.3.1 Axis #1: Software Language Engineering

**Overall objective.** The disruptive design of new, complex systems requires a high degree of flexibility in the communication between many stakeholders, often limited by the silo-like structure of the organization itself (cf. Conway's law). To overcome this constraint, modern engineering environments aim to: (i) better manage the necessary exchanges between the different stakeholders; (ii) provide a unique and usable place for information sharing; and (iii) ensure the consistency of the many points of view. Software languages are the key pivot between the *diverse* stakeholders involved, and the software systems they have to implement. Domain-Specific (Modeling) Languages enable stakeholders to address the *diverse* concerns through specific points of view, and their coordinated use is essential to support the socio-technical coordination across the overall software system life cycle.

Our perspectives on Software Language Engineering over the next period is presented in Figure 2 and detailed in the following paragraphs.

**DSL Executability.** Providing rich and adequate environments is key to the adoption of domain-specific languages. In particular, we focus on tools that support model and program execution. We explore the foundations to define the required concerns in language specification, and systematic approaches to derive environments (*e.g.*, IDE, notebook, design labs) including debuggers, animators, simulators, loggers, monitors, trade-off analysis, etc.

**Modular & Distributed IDE.** IDEs are indispensable companions to software languages. They are increasingly turning towards Web-based platforms, heavily relying on cloud infrastructures and forges. Since all language services require different computing capacities and response times (to guarantee a user-friendly experience within the IDE) and use shared resources (*e.g.*, the program), we explore new architectures for their modularization and systematic approaches for their individual deployment and dynamic adaptation within an IDE. To cope with the ever-growing number of programming languages, manufacturers of Integrated Development Environments (IDE) have recently defined protocols as a way to use and share multiple language services in language-agnostic environments. These protocols rely on

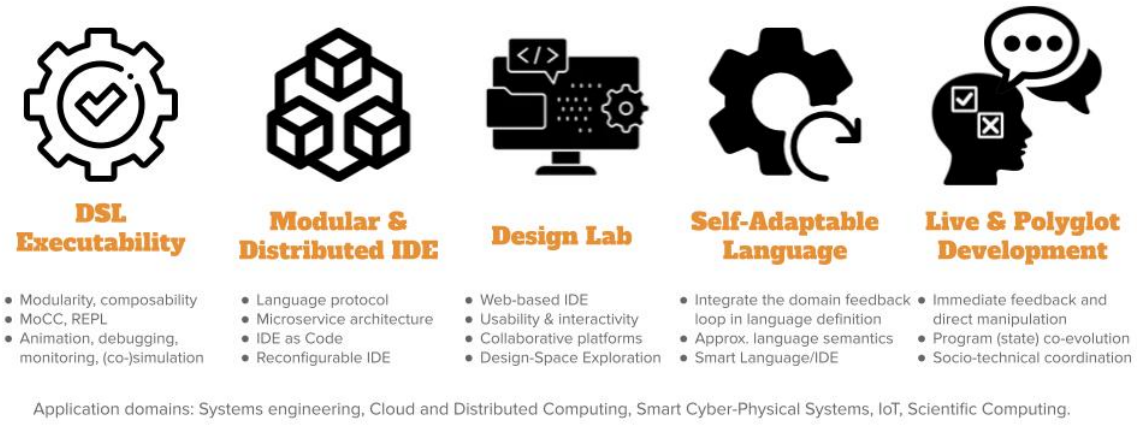


Figure 2: Perspectives on Software Language Engineering (axis #1)

a proper specification of the services that are commonly found in the tool support of general-purpose languages, and define a fixed set of capabilities to offer in the IDE. However, new languages regularly appear offering unique constructs (e.g., DSLs), and which are supported by dedicated services to be offered as new capabilities in IDEs. This trend leads to the multiplication of new protocols, hard to combine and possibly incompatible (e.g., overlap, different technological stacks). Beyond the proposition of specific protocols, we will explore an original approach to be able to specify language protocols and to offer IDEs to be configured with such protocol specifications. IDEs went from directly supporting languages to protocols, and we envision the next step: *IDE as code*, where language protocols are created or inferred on demand and serve as support of an adaptation loop taking in charge of the (re)configuration of the IDE.

**Design Lab.** Web-based and cloud-native IDEs open new opportunities to bridge the gap between the IDE and collaborative platforms, *e.g.,* forges. In the complex world of software systems, we explore new approaches to reduce the distance between the various stakeholders (e.g., systems engineers and all those involved in specialty engineering) and to improve the interactions between them through an adapted tool chain. We aim to improve the usability of development cycles with efficiency, affordance and satisfaction. We also explore new approaches to explore and interact with the design space or other concerns such as human values or security, and provide facilities for trade-off analysis and decision making in the the context of software and system designs.

**Live & Polyglot Development.** As of today, polyglot development is massively popular and virtually all software systems put multiple languages to use, which not only complexifies their development, but also their evolution and maintenance. Moreover, as software are more used in new application domains (e.g., data analytics, health or scientific computing), it is crucial to ease the participation of scientists, decision-makers, and more generally non-software experts. Live programming makes it possible to change a program while it is running, by propagating changes on a program code to its run-time state. This effectively bridges the gulf of evaluation between program writing and program execution: the effects a change has on the running system are immediately visible, and the developer can take immediate action. The challenges at the intersection of polyglot and live programming have received little attention so far, and we envision a language design and implementation approach to specify domain-specific languages and their coordination, and automatically provide interactive domain-specific environments for live and polyglot programming.

**Self-Adaptable Language.** Over recent years, self-adaptation has become a concern for many software systems that operate in complex and changing environments. At the core of self-adaptation lies a feedback loop and its associated trade-off reasoning, to decide on the best course of action. However, existing software languages do not abstract the development and execution of such feedback loops for self-

adaptable systems. Developers have to fall back to ad-hoc solutions to implement self-adaptable systems, often with wide-ranging design implications (e.g., explicit MAPE-K loop). Furthermore, existing software languages do not capitalize on monitored usage data of a language and its modeling environment. This hinders the continuous and automatic evolution of a software language based on feedback loops from the modeling environment and runtime software system. To address the aforementioned issues, we will explore the concept of Self-Adaptable Language (SAL) to abstract the feedback loops at both system and language levels.

### 3.3.2 Axis #2: Spatio-temporal Variability in Software and Systems

**Overall objective.** Leveraging our longstanding activity on variability management for software product lines and configurable systems covering *diverse* scenarios of use, we will investigate over the next period the impact of such a variability across the *diverse* layers, incl. source code, input/output data, compilation chain, operating systems and underlying execution platforms. We envision a better support and assistance for the configuration and optimisation (e.g., non-functional properties) of software systems according to this deep variability. Moreover, as software systems involve *diverse* artefacts (*e.g.,* APIs, tests, models, scripts, data, cloud services, documentation, deployment descriptors...), we will investigate their continuous co-evolution during the overall lifecycle, including maintenance and evolution. Our perspectives on spatio-temporal variability over the next period is presented in Figure 3 and is detailed in the following paragraphs.


Figure 3: Perspectives on Spatio-temporal Variability in Software and Systems (axis #2)

**Deep Software Variability.** Software systems can be configured to reach specific functional goals and non-functional performance, either statically at compile time or through the choice of command line options at runtime. We observed that considering the software layer only might be a naive approach to tune the performance of the system or to test its functional correctness. In fact, many layers (hardware, operating system, input data, etc.), which are themselves subject to variability, can alter the performance or functionalities of software configurations. We call *deep software variability* the interaction of all variability layers that could modify the behavior or non-functional properties of a software. Deep software variability calls to investigate how to systematically handle cross-layer configuration. The diversification of the different layers is also an opportunity to test the robustness and resilience of the software layer in multiple environments. Another interesting challenge is to tune the software for one specific executing environment. In essence, deep software variability questions the generalization of the configuration knowledge.

**Continuous Software Evolution.** Nowadays, software development has become more and more complex, involving various artefacts, such as APIs, tests, models, scripts, data, cloud services, documentation, etc., and embedding millions of lines of code (LOC). Recent evidence highlights continuous software

evolution based on thousands of commits, hundreds of releases, all done by thousands of developers. We focus on the following essential backbone dimensions in software engineering: languages, models, APIs, tests and deployment descriptors, all revolving around software code implementation. We will explore the foundations of a multidimensional and polyglot co-evolution platform, and will provide a better understanding with new empirical evidence and knowledge.

#### 3.3.3 Axis #3: DevSecOps and Resilience Engineering for Software and Systems

**Overall objective.** The production and delivery of modern software systems involves the integration of *diverse* dependencies and continuous deployment on *diverse* execution platforms in the form of large distributed socio-technical systems. This leads to new software architectures and programming models, as well as complex supply chains for final delivery to system users. In order to boost cybersecurity, we want to provide strong support to software engineers and IT teams in the development and delivery of secure and resilient software systems, ie. systems able to resist or recover from cyberattacks. Our perspectives on DevSecOps and Resilience Engineering over the next period are presented in Figure 4 and detailed in the following paragraphs.

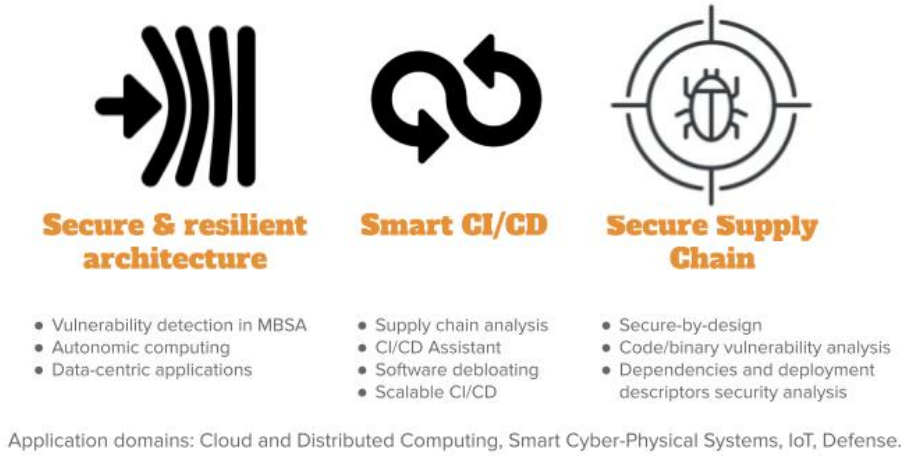


Figure 4: Perspectives on DevSecOps and Resilience Eng. for Software and Systems (axis #3)

**Secure & Resilient Architecture.** Continuous integration and deployment pipelines are processes implementing complex software supply chains. We envision an explicit and early consideration of security properties in such pipelines to help in detecting vulnerabilities. In particular, we integrate the security concern in Model-Based System Analysis (MBSA) approaches, and explore guidelines, tools and methods to drive the definition of secure and resilient architectures. We also investigate resilience at runtime through frameworks for autonomic computing and data-centric applications, both for the software systems and the associated deployment descriptors.

**Smart CI/CD.** Dependencies management, Infrastructure as Code (IaC) and DevOps practices open opportunities to analyze complex supply chains. We aim at providing relevant metrics to evaluate and ensure the security of such supply chains, advanced assistants to help in specifying corresponding pipelines, and new approaches to optimize them (*e.g.*, software debloating, scalability…). We study how supply chains can actively leverage software variability and diversity to increase cybersecurity and resilience.

**Secure Supply Chain.** In order to produce secure and resilient software systems, we explore new secure-by-design foundations that integrate security concerns as first class entities through a seamless continuum from the design to the continuous integration and deployment. We explore new models, architectures, inter-relations, and static and dynamic analyses that rely on explicitly expressed security

concerns to ensure a secure and resilient supply chain. We lead research on automatic vulnerability and malware detection in modern supply chains, considering the various artefacts either as white boxes enabling source code analysis (to avoid accidental vulnerabilities or intentional ones or code poisoning), or as black boxes requiring binary analysis (to find malware or vulnerabilities). We also conduct research activities in dependencies and deployment descriptors security analysis.

# 4 Application domains

Information technology affects all areas of society. The need to develop software systems is therefore present in a huge number of application domains. One of the goals of software engineering is to *apply a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software* whatever the application domain.

As a result, the team covers a wide range of application domains and never refrains from exploring a particular field of application. Our primary expertise is in complex, heterogeneous and distributed systems. While we historically collaborated with partners in the field of systems engineering, it should be noted that for several years now, we have investigated several new areas in depth:

- the field of web applications, with the associated design principles and architectures, for applications ranging from cloud-native applications to the design of modern web front-ends.
- the field of scientific computing in connection with the CEA DAM, Safran and scientists from other disciplines such as the ecologists of the University of Rennes. In this field where the writing of complex software is common, we explore how we could help scientists to use software engineering approach, in particular, the use of SLE and approximate computing techniques.
- the field of large software systems such as the Linux kernel or other open-source projects. In this field, we explore, in particular, the variability management, the support of co-evolution and the use of polyglot approaches.

# 5 Social and environmental responsibility

## 5.1 Footprint of research activities

We share the vision that reducing the environmental footprint of research activities is crucial for promoting sustainability within academic and scientific communities. Here are some examples of actions that we promote within the team:

We encourage virtual seminars (e.g., the creation of the EDT Community (cf. https://edt.community) on the engineering of digital twins) and meetings (not conferences) to reduce the need for long-distance travel. When travel is necessary, we try to opt for modes of transportation with lower carbon footprints, such as trains. We want to share that INRIA has to improve the booking system that do not offer trains that go to London for example, as well as reasonable per diem reimbursements that cover the actual costs (e.g., Amsterdam where even the travel agency is incapable of proposing hotels within the budget) so that as people can stay longer working with colleagues when they have to travel.

We try to engage students of the field through educational outreach: We raise awareness about the importance of environmental sustainability within research communities through educational programs and seminars (We organise ICT4S this year as a joint event with the GDRGPL days). We encourage students to incorporate sustainable practices into their work. We have also started to create scientific results on the impact of software development practices on environmental sustainability. Quentin Perez has been hired as a new faculty member on this research topic.

## 5.2 Impact of research results

The DiverSE project-team initiated several research activities at the crossroads of sustainability and software engineering. In particular, the research challenges are twofold: i) GreenIT, and more specifically how to measure the energy consumption of software all along the development life cycle and the DevOps

pipelines, and ii) IT for green, more specifically the engineering of digital twins either to optimize and reconfigure, or to support informed decisions in tradeoff analysis and design space exploration. In this context, the project-team organized in 2023 the international conference on Information and Communications Technology for Sustainability (ICT4S), with not only a research program, but also a so called OFF! Program which complements the research program with a set of satellite events bringing together researchers, practitioners, decision and policy makers, artists, students and the general public. It proposed various kinds of events on campus as well as in pubs downtown. In particular, the OFF! Program included general keynotes, panels, debates, art performances, etc.

Moreover, the DiverSE project-team is currently exploring several research axes related to social and environmental challenges, all in a pluri-disciplinary context. In particular, the team is involved in both: i) collaboration with environmental sciences and sociology on the use of climate change scientific models for decision-makers, and ii) collaboration with sociology on the privacy in web applications.

# 6 Highlights of the year

- Jean-Marc Jézéquel has been appointed as a fellow of the Institut Universitaire de France (IUF) in Sept. 2023.
- We organised the ICT4S conference and the GDR GPL days in parallel. We brought together over 400 students and researchers for a week on the subjects of software engineering and sustainability (e.g., greenIT and IT4Green).
- Olivier Zendra got his Habilitation à Dirigier des Recherches (HDR) on 20 September 2023.

## 6.1 Awards

- It was a great journey working with Airbus, leading to scientific contributions and concrete applications! We got a best paper award certificate of Models 2023 for our paper about software product lines and a combination of negative and positive variability [47]. Great collaboration with Airbus, McGill, and University of Rennes!
- CSAW'23 award for Piergiogio Ladisa. The CSAW event brought together the elite young researchers in cybersecurity who had published in the most prestigious conferences in the field over the previous year. During the final, the authors presented their research to a panel of experts and pitched their project for 3 minutes to the general public. The judges had the difficult task of deciding between the finalists, with each of the projects presented being particularly impressive and having a significant impact on future security techniques. **First place**: Piergiorgio Ladisa: SAP Security Research & Université de Rennes 1, Inria, IRISA (France) : SoK: Taxonomy of Attacks on Open-Source Software Supply Chains

# 7 New software, platforms, open data

## 7.1 New software

### 7.1.1 GEMOC Studio

**Name:** 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):

1) 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. 2) 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. 3) 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. 4) BCOoL, a tool-supported meta-language dedicated to the specification of language coordination patterns to automatically coordinates the execution of, possibly heterogeneous, models. 5) Monilog, an extension for monitoring and logging executable domain-specific models 6) 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 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 designers to 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 provides various generic services such as graphical animation, debugging tools, trace and event managers, timeline.

**URL:** http://gemoc.org/studio.html

**Publications:** hal-00850770, hal-01355391, hal-01609576, hal-01651801, hal-01152342, hal-03374955, hal-01614561, hal-01616154

**Contact:** Benoît Combemale

**Participants:** Didier Vojtisek, Erwan Bousse, Julien Deantoni

**Partners:** I3S, Université de Nantes

#### 7.1.2 Interacto

**Keywords:** GUI (Graphical User Interface), User Interfaces, HCI, Software engineering

**Functional Description:** Interacto is a framework for developing user interfaces and user interactions. It complements other general graphical framework by providing a fluent API specifically designed to process user interface event and develop complex user interactions. Interacto is currently developped in Java and TypeScript to target both Java desktop applications (JavaFX) and Web applications (Angular).

**URL:** https://interacto.github.io

**Publications:** hal-03231669, tel-02354530, inria-00590891, inria-00477627

**Contact:** Arnaud Blouin

**Participants:** Arnaud Blouin, Olivier Beaudoux

#### 7.1.3 HyperAST

**Keywords:** Code analysis, Git svn

**Functional Description:** The HyperAST is an AST structured as a Direct Acyclic Graph (DAG) (similar to MerkleDAG used in Git). An HyperAST is efficiently constructed by leveraging Git and TreeSitter.

It reimplements the Gumtree algorithm in Rust while using the HyperAST as the underlying AST structure.

It implements a use-def solver, that uses a context-free indexing of references present in subtrees (each subtree has a bloom filter of contained references).

**Author:** Quentin Le Dilavrec

**Contact:** Olivier Barais

#### 7.1.4 CorrectExam

**Name:** CorrectExam: GRADE YOUR ASSESSMENTS MORE EFFICIENTLY

**Keyword:** Digital pedagogy

**Functional Description:** The first objective of the correctexam project is pedagogical. The aim is to be able to send feedback to students as quickly as possible on the marking of their papers, to easily generate a standard answer key from answers marked as excellent by the marker, and to facilitate a constructive exchange between students and the teaching team. This helps to overcome a shortcoming at university where, as examinations generally take place partly at the end of the course, students are not strongly encouraged to look at their marked papers in order to understand their mistakes. The second objective is to seek to increase the efficiency of exam marking and the administrative aspects associated with an exam by using AI techniques to mark certain questions, and by factoring standard comments added to an exam paper, generating documents in the format expected by the school, and so on. Finally, the last notable element of the project that could be discussed concerns the choice of technical architecture. Even though an application server is used to store the students' results, all the processing of the scans (pdf), images and AI is carried out completely on the browser side, using the possibilities offered by modern browsers such as WASM or worker services. This is an opportunity to significantly limit the power required on the server side.

**Release Contributions:** See https://correctexam.github.io/#about

**Contact:** Olivier Barais

**Partner:** Université de Rennes 1

#### 7.1.5 PolyglotAST

**Name:** PolyglotAST

**Keywords:** Code analysis, Static analysis

**Functional Description:** Framework to facilitate the static analysis of multilingual programs on GraalVM, by providing a unified representation of the various sub-programs via a single AST

**Author:** Philemon Houdaille

**Contact:** Olivier Barais

#### 7.1.6 HydroPredictUI

**Name:** Jupyter graphical interface for HydroModPy

**Keywords:** GUI (Graphical User Interface), Jupyter, Simulator, Scientific computing, Distributed Applications

**Functional Description:** HydroModPy is a Python tool for running numerical simulations of groundwater flow. The aim of the HydroPredictUi software is to provide a graphical interface in the form of a Jupyter notebook to make it easier to learn and run simulations on a remote server.

**Contact:** Johann Bourcier

### 7.2 New platforms

**A platform for experimentation as part of the digital twins of Industry 4.0.**

**Participants:** Olivier Barais, Benoit Combemale, Jean-Marc Jézéquel, Quentin Perez, Didier Vojtisek.

As part of the ANR MBDO project in conjunction with our German partners, we are creating a platform to emulate the behaviour of a factory. On the hardware side, this platform consists of a FisherTechnik base. FisherTechnik The digital twins software layer is built using the GEMOC platform. In 2023, we worked mainly on the specification, equipment orders and initial experiments. This platform will be further developed in 2024.

### 7.3 Open data

- Piergiorgio Ladisa contributes to the Backstabbers-Knife-Collection dataset https://github.com/cybertier/Backstabbers-Knife-Collection/.
- Piergiorgio Ladisa created a public dataset of runnable examples for multiple ecosystems, explaining how a 3rd-party dependency can trigger execution in downstream projects, ultimately resulting in OSS supply chain attacks. https://github.com/SAP-samples/risk-explorer-execution-pocs
- In the context of a collaboration with Université de Montréal and Software Heritage, we proposed an approach for fingerprinting and building large reproducible datasets [74]. We show how it can help reduce the limitations researchers face when creating or reproducing datasets.

## 8 New results

### 8.1 Results for Axis #1: Software Language Engineering

**Participants:** Olivier Barais, Johann Bourcier, Benoît Combemale, Jean-Marc Jézéquel, Djamel Eddine Khelladi, Gurvan Leguernic, Gunter Mussbacher, Noël Plouzeau, Didier Vojtisek.

#### 8.1.1 Modeling: From CASE Tools to SLE and Machine Learning

Finding better ways to handle software complexity (both inherent and accidental) is the holy grail for a significant part of the software engineering community, and especially for the Model Driven Engineering (MDE) one. To that purpose, plenty of techniques have been proposed, leading to a succession of trends in model based software developments paradigms in the last decades. While these trends seem to pop out from nowhere, we claim in [65] that most of them actually stem from trying to get a better grasp on the variability of software. We revisit the history of MDE trying to identify the main aspect of variability they wanted to address when they were introduced. We conclude on what are the variability challenges of our time, including variability of data leading to machine learning of models.

#### 8.1.2 A Generic Framework for Representing and Analysing Model Concurrency

Recent results in language engineering simplify the development of tool-supported executable domain-specific modelling languages (xDSMLs), including editing (e.g., completion and error checking) and execution analysis tools (e.g., debugging, monitoring and live modelling). However, such frameworks are currently limited to sequential execution traces, and cannot handle execution traces resulting from an execution semantics with a concurrency model supporting parallelism or interleaving. This prevents the development of concurrency analysis tools, like debuggers supporting the exploration of model executions resulting from different interleavings. In [41], we present a generic framework to integrate execution semantics with either implicit or explicit concurrency models, to explore the possible execution traces of conforming models, and to define strategies for helping in the exploration of the possible

executions. This framework is complemented with a protocol to interact with the resulting executions and hence to build advanced concurrency analysis tools. The approach has been implemented within the GEMOC Studio. We demonstrate how to integrate two representative concurrent meta-programming approaches (MoCCML/Java and Henshin), which use different paradigms and underlying foundations to define an xDSML's concurrency model. We also demonstrate the ability to define an advanced concurrent omniscient debugger with the proposed protocol. The work, thus, contributes key abstractions and an associated protocol for integrating concurrent meta-programming approaches in a language workbench, and dynamically exploring the possible executions of a model in the modelling workbench.

#### 8.1.3 Adaptive Structural Operational Semantics

Software systems evolve more and more in complex and changing environments, often requiring runtime adaptation to best deliver their services. When self-adaptation is the main concern of the system, a manual implementation of the underlying feedback loop and trade-off analysis may be desirable. However, the required expertise and substantial development effort make such implementations prohibitively difficult when it is only a secondary concern for the given domain. In [49], we present ASOS, a metalanguage abstracting the runtime adaptation concern of a given domain in the behavioral semantics of a domain-specific language (DSL), freeing the language user from implementing it from scratch for each system in the domain. We demonstrate our approach on RobLANG, a procedural DSL for robotics, where we abstract a recurrent energy-saving behavior depending on the context. We provide formal semantics for ASOS and pave the way for checking properties such as determinism, completeness, and termination of the resulting self-adaptable language. We provide first results on the performance of our approach compared to a manual implementation of this selfadaptable behavior. We demonstrate, for RobLANG, that our approach provides suitable abstractions for specifying sound adaptive operational semantics while being more efficient.

#### 8.1.4 Testing Metamodel and Code Co-evolution

Models play a significant role in Model-Driven Engineering (MDE) and metamodels are commonly transformed into code. Developers intensively rely on the generated code to build language services and tooling, such as editors and views which are also tested to ensure their behavior. The metamodel evolution between releases updates the generated code, and this may impact the developers' additional, client code. Accordingly, the impacted code must be co-evolved too, but there is no guarantee of preserving its behavior correctness. In [50], we envision an automatic approach for ensuring code co-evolution correctness. It first aims to trace the tests impacted by the metamodel evolution before and after the code co-evolution, and then compares them to analyze the behavior of the code. Preliminary evaluation on two implementations of OCL and Modisco Eclipse projects showed that we can successfully trace the impacted tests automatically by selecting 738 and 412 tests, before and after co-evolution respectively, based on 303 metamodel changes. By running these impacted tests, we observed both behaviorally correct and incorrect code co-evolution.

#### 8.1.5 Practical Runtime Instrumentation of Software Languages: The Case of SciHook

Software languages have pros and cons, and are usually chosen accordingly. In this context, it is common to involve different languages in the development of complex systems, each one specifically tailored for a given concern. However, these languages create de facto silos, and offer little support for interoperability with other languages, be it statically or at runtime. In [56], we report on our experiment on extracting a relevant behavioral interface from an existing language, and using it to enable interoperability at runtime. In particular, we present a systematic approach to define the behavioral interface and we discuss the expertise required to define it. We illustrate our work on the case study of SciHook, a C++ library enabling the runtime instrumentation of scientific software in Python. We present how the proposed approach, combined with SciHook, enables interoperability between Python and a domain-specific language dedicated to numerical analysis, namely NabLab, and discuss overhead at runtime.

### 8.1.6 Polyglot Software Development and Code Analysis

The notion of polyglot software development refers to the fact that most software projects nowadays rely on multiple languages to deal with widely different concerns, from core business concerns to user interface, security, and deployment concerns among many others. Many different wordings around this notion have been proposed in the literature, with little understanding of their differences. In [39], we propose a concise and unambiguous definition of polyglot software development including a conceptual model and its illustration on a well-known, open-source project. We further characterize the techniques used for the specification and operationalization of polyglot software development with a feature model, concentrating on polyglot programming. Finally, we outline the many challenges and perspectives raised by polyglot software development.

In this contexts, GraalVM and PolyNote are examples of runtimes allowing polyglot programming. However, there is a striking lack of support at design time for building and analyzing polyglot code. To the best of our knowledge, there is no uniform language-agnostic way of reasoning over multiple languages to provide seamless code analysis, since each language comes with its own form of Abstract Syntax Trees (AST). In [48], we present an approach to build a uniform yet polyglot AST over polyglot code, so that it is easier to perform global analysis. We first motivate this challenge and identify the main requirements for building a polyglot AST. We then propose a proof of concept implementation of our solutions on GraalVM's polyglot API. On top of the polyglot AST, we demonstrate the ability to implement several polyglot-specific analysis services, namely auto-completion, consistency checking, type inference, and rename refactoring. Our evaluation on three polyglot projects taken from GitHub, and involving JavaScript and Python code, shows that we can build a polyglot AST without significant overhead. We also demonstrate the usefulness of the polyglot analysis services through the provided automation, as well as their scalability.

### 8.1.7 Pull Requests Integration Process Optimization: An Empirical Study

Pull-based Development (PbD) is widely used in collaborative development to integrate changes into a project codebase. In this model, contributions are notified through Pull Request (PR) submissions. Project administrators are responsible for reviewing and integrating PRs. In the integration process, conflicts occur when PRs are concurrently opened on a given target branch and propose different modifications for a same code part. In a previous work, we proposed an approach, called IP Optimizer, to improve the Integration Process Efficiency (IPE) by prioritizing PRs. In this work [67], we conduct an empirical study on 260 open-source projects hosted by GitHub that use PRs intensively in order to quantify the frequency of conflicts in software projects and analyze how much the integration process can be improved. Our results indicate that regarding the frequency of conflicts in software projects, half of the projects have a moderate and high number of pairwise conflicts and half have a low number of pairwise conflicts or none. Futhermore, on average 18.82% of the time windows have conflicts. On the other hand, regarding how much the integration process can be improved, IP Optimizer improves the IPE in 94.16% of the time windows and the average improvement percentage is 146.15%. In addition, it improves the number of conflict resolutions in 67.16% of the time windows and the average improvement percentage is 134.28%.

## 8.2 Results for Axis #2: Spatio-temporal Variability in Software and Systems

**Participants:** Mathieu Acher, Olivier barais, Arnaud Blouin, Benoît Combemale, Jean-Marc Jézéquel, Djamel Eddine Khelladi, Olivier Zendra, Paul Temple
.

### 8.2.1 Generative AI and Large Language Models for Variability

*LLM for programming variability* Programming variability is central to the design and implementation of software systems that can adapt to a variety of contexts and requirements, providing increased flexibility and customization. Managing the complexity that arises from having multiple features, variations, and

possible configurations is known to be highly challenging for software developers. In this work, we explore how large language model (LLM)-based assistants can support the programming of variability. In [43] we report on new approaches made possible with LLM-based assistants, like: features and variations can be implemented as prompts; augmentation of variability out of LLM-based domain knowledge; seamless implementation of variability in different kinds of artefacts, programming languages, and frameworks, at different binding times (compile-time or run-time).

*LLM for re-engineering variants* We are interested in the following problem: given a set of variants (Java, C, SVG, UML, state charts, etc.) how to build a configurable program (a software product line aka SPL) that allows you to retrieve/derive them? For instance let us say you have three variants written in Java. What would be the Java program that can be configured to retrieve them? You can do it manually but it is error-prone and time-consuming. In [45] we explore the use of LLM and ChatGPT for this problem. We revisit four illustrative cases of the literature where the challenge is to migrate variants written in a different formalism (UML class diagrams, Java, GraphML, statecharts). We systematically report on our experience with ChatGPT-4, describing our strategy to prompt LLMs and documenting positive aspects but also failures. We compare the use of LLMs with a state-of-the-art approach, BUT4Reuse. While LLMs offer potential in assisting domain analysts and developers in transitioning software variants into SPLs, their intrinsic stochastic nature and restricted ability to manage large variants or complex structures necessitate a semiautomatic approach, complete with careful review, to counteract inaccuracies.

*End-user customization with generative AI* Producing a variant of code is highly challenging, particularly for individuals unfamiliar with programming. In [42], we introduce a novel use of generative AI to aid end-users in customizing code. We first describe how generative AI can be used to customize code through prompts and instructions, and further demonstrate its potential in building end-user tools for configuring code. We showcase how to transform an undocumented, technical, low-level TikZ into a user-friendly, configurable, Web-based customization tool written in Python, HTML, CSS, and JavaScript and itself configurable. We discuss how generative AI can support this transformation process and traditional variability engineering tasks, such as identification and implementation of features, synthesis of a template code generator, and development of end-user configurators. We believe it is a first step towards democratizing variability programming, opening a path for end-users to adapt code to their needs.

### 8.2.2 Reverse Engineering Variability

Software Product Lines (SPLs) are families of systems that share common assets allowing disciplined software reuse. The adoption of SPLs practices has been shown to enable significant technical and economic benefits for the companies that employ them. However, successful SPLs rarely start from scratch. Instead, they usually start from a set of existing legacy systems that must undergo a well-defined re-engineering process.

Many approaches to conduct such re-engineering processes have been proposed and documented in the literature. This handbook is the result of the collective community expertise and knowledge acquired in conducting theoretical and empirical research also in partnership with industry. The topic discussed in this handbook is a recurrent and challenging problem faced by many companies. Conducting a reengineering process could unlock new levels of productivity and competitiveness. The chapter authors are all experts in different topics of the re-engineering process, which brings valuable contributions to the content of this handbook. Additionally, organizing the international workshop on REverse Variability Engineering (REVE) has contributed to this topic during the last decade. REVE has fostered research collaborations between Software Re-engineering and SPL Engineering (SPLE) communities. Thus, this handbook is also a result of our expertise and knowledge acquired from the fruitful discussions with the attendants of REVE. Our handbook aims to bring together into a single, comprehensive, and cohesive reference the wealth of experience and expertise in the area of re-engineering software intensive systems into SPLs. We cover the entire re-engineering life-cycle, from requirements gathering to maintenance and evolution tasks. Also, we provide future directions and perspectives.

We released the book "Handbook of Re-Engineering Software Intensive Systems into Software Product Lines". It is the result of a collective effort over the last 3 years. It underwent a rigorous and careful selection and edition process. The selected contributors are worldwide experts in their field, and all chapters were peer reviewed.

We also contributed with a chapter "Machine Learning for Feature Constraints Discovery" that provides an overview of methods and applications of automatically extracting unspecified constraints out of a software system (e.g., Linux, 3D printing models, video generator).

#### 8.2.3 A Specialized Language to Realize Variability at Airbus

In software product line (SPL) engineering, feature models are the de facto standard for modeling variability. A user can derive products out of a base model by selecting features of interest. Doing it automatically, however, requires a realization model, which is a description of how a base model should be modified when a given feature is selected/unselected. A realization model then necessarily depends on the base metamodel, asking for ad hoc solutions that have flourished in recent years. In [47], we propose Greal, a generic solution to this problem in the form of (1) a generic declarative realization language that can be automatically composed with one or more base metamodels to yield a domain-specific realization language and (2) a product derivation algorithm applying a realization model to a base model and a resolved model to yield a derived product. We describe how, on top of Greal, we specialized a realization language to support both positive and negative variability, fit the syntax and semantics of the targeted language (BPMN) and take into account modeling practices at Airbus. We report on lessons learned of applying this approach on Program Development Plans based on business process models and discuss open problems.

We won a best paper at the ACM/IEEE 26th International Conference on Model-Driven Engineering Languages and Systems. link

#### 8.2.4 Debloating Variability

*A call to remove variability.* Software variability is largely accepted and explored in software engineering and seems to have become a norm and a must, if only in the context of product lines. Yet, the removal of superfluous or unneeded software artefacts and functionalities is an inevitable trend. It is frequently investigated in relation to software bloat. In [44] we call the community on software variability to devise methods and tools that will facilitate the removal of unneeded variability from software systems. The advantages are expected to be numerous in terms of functional and non-functional properties, such as maintainability (lower complexity), security (smaller attack surface), reliability, and performance (smaller binaries).

*Specializing configuration space through debloating.* Numerous software systems are highly configurable through runtime options (e.g., command-line parameters). Users can tune some of the options to meet various functional and non-functional requirements such as footprint, security, or execution time. However, some options are never set for a given system instance, and their values remain the same whatever the use cases of the system. In [62], we design a controlled experiment in which the system's run-time configuration space can be specialized at compile-time and combinations of options can be removed on demand. We perform an in-depth study of the well-known x264 video encoder and quantify the effects of its specialization to its non-functional properties, namely on binary size, attack surface, and performance while ensuring its validity. Our exploratory study suggests that the configurable specialization of a system has statistically significant benefits on most of its analysed non-functional properties, which benefits depend on the number of the debloated options. While our empirical results and insights show the importance of removing code related to unused run-time options to improve software systems, an open challenge is to further automate the specialization process.

#### 8.2.5 Software Build Variability

Software engineers are acutely aware that the building of software is an essential but resource-intensive step in any software development process. This is especially true when building large systems or highly configurable systems whose vast number of configuration options results in a space explosion in the number of versions that should ideally be built and evaluated. Linux is precisely one such large and highly configurable system with thousands of options that can be combined. A previous study showed the benefit of incremental build, however, only on small-sized configurable software systems, unlike Linux. In [78], we show preliminary results of our ongoing work on enabling efficient exploration of the Linux configuration space with incremental builds. Although incremental compilation for post-commit is

used in Linux, we show that the build of large numbers of random Linux configurations does not benefit from incremental build. Thus, we introduce and detail PyroBuildS, our new approach to efficiently explore, with incremental builds, the very large configuration space of Linux. Very much like fireworks, PyroBuildS starts from several base configurations ("rockets") and generates mutated configurations ("sparks") derived from each of the base ones. This enables exploring the configuration space with an efficient incremental build of the mutants, while keeping a good amount of diversity. We show on a total of 2520 builds that our PyroBuildS approach does trigger synergies with the caching capabilities of Make, hence significantly decreasing builds time with gains up to 85%, while having a diversity of 33% of options and 15 out of 17 subsystems. Overall, individual contributors and continuous integration services can leverage PyroBuildS to efficiently augment their configuration builds, or reduce the cost of building numerous configurations.

#### 8.2.6 Deep Variability

Deep software variability refers to the interaction of all external layers (hardware, operating system, compiler, versions, etc.) modifying the behavior of software. Configuring software is a powerful means to reach functional and performance goals of a system, but many layers of variability can make this difficult.

One dimension of the problem is of course that performance depends on the input data: e.g., a video as input to an encoder like x264 or a file fed to a tool like xz . To achieve good performance, users should therefore take into account both dimensions of (1) software variability and (2) input data. In [37] we detail a large study over 8 configurable systems that quantifies the existing interactions between input data and configurations of software systems. The results exhibit that (1) inputs fed to software systems can interact with their configuration options in non-monotonous ways, significantly impacting their performance properties (2) input sensitivity can challenge our knowledge of software variability and question the relevance of performance predictive models for a field deployment. Given the results of our study, we call researchers to address the problem of input sensitivity when tuning, predicting, understanding, and benchmarking configurable systems.

Owing to the significance of the input-configuration interplay, we propose solutions and methods to address the problem. In [38], we empirically evaluate how supervised and transfer learning methods can be leveraged to efficiently learn performance models based on configuration options and input data. Our study over 1,941,075 data points empirically shows that measuring the performance of configurations on multiple inputs allows one to reuse this knowledge and train performance models robust to the change of input data. To the best of our knowledge, this is the first domain-agnostic empirical evaluation of machine learning methods addressing the input-aware performance prediction problem.

This line of work is a fruitful collaboration between Simula and Inria through RESIST-EA associate team RESIST

#### 8.2.7 Variability-Aware debugging

Business processes have to manage variability in their execution, e.g., to deliver the correct building permit in different municipalities. This variability is visible in event logs, where sequences of events are shared by the core process (building permit authorisation) but may also be specific to each municipality. To rationalise resources (e.g., derive a configurable business process capturing all municipalities' permit variants) or to debug anomalous behaviour, it is mandatory to identify to which variant a given trace belongs. Manually providing this whole mapping is labour-intensive. [102] experimented with variant-based mapping using supervised machine learning (ML) to identify the variants responsible of the production of a given execution trace, and demonstrated that recurrent neural networks (RNNs) work well ($\geq$ 80% accuracy) when trained on datasets in which we label execution traces with variants. However, this mapping (i) may not scale to large VIS because of combinatorial explosion and (ii) makes the internal ML representation hard to understand. In [77], we discuss the design of a novel approach: feature-based mapping learning; where contrarily to our previous work, we do not want to map execution traces to configurations directly but to features composing the configurations.

#### 8.2.8 Representation of variability

Building on this idea of changing the perspective of the addressed problems and representation in variability; [60] discusses the differences in practices regarding feature engineering in the ML community and in the software variability community. While initiatives in applying ML models to software variability has increased, we noticed that the representation space in which ML models work and the ones used in software variability differ. ML models like their representation spaces to be continuous and differentiable while software variability practitioners usually work on the features used to describe software configurations. These features can be heterogeneous (i.e., some may be numerical values like integers or float values, while other may be Boolean) preventing the space from being continuous and requiring being extra-careful when using ML models since they may have trouble coping with heterogeneity. [60] discusses the fact that to be able to use deep learning models in the world of software variability, we need to think differently to create a representation space that is continuous and derivable but at the cost of interpretability; or we stick to machine learning models that are less efficient but on which we can have a better control and better understanding.

#### 8.2.9 Scaling Diff computation in temporal variability

With the advent of fast software evolution and multistage releases, temporal code analysis is becoming useful for various purposes, such as bug cause identification, bug prediction or code evolution analysis. Temporal code analyses can consist in analyzing multiple Abstract Syntax Trees (ASTs) extracted from code evolutions, e.g. one AST for each commit or release. Core feature to temporal analysis is code differencing: the computation of the so-called Diff or edit script between two given versions of the code. However, jointly analyzing and computing the difference on thousands versions of code faces scalability issues. Mainly because of the cost of: 1) parsing the original and evolved code in two source and target ASTs; 2) wasting resources by not reusing intermediate computation results that can be shared between versions. In [55], we detail a novel approach based on time-oriented data structures that makes code differencing scale up to large software codebases. In particular, we leverage on the HyperAST, a novel representation of code histories, to propose an incremental and memory efficient approach by lazifying the well known GumTree diffing algorithms, a mainstream code differencing algorithm and tool. We evaluated our approach on a curated list of 19 large software projects and compared it to GumTree. Our approach outperforms it in scalability both in time and memory. We observed an order-of-magnitude difference: 1) in CPU time from x1.2 to x12.7 for the total time of diff computation and up to x226 in intermediate phases of the diff computation, and 2) in memory footprint of x4.5 per AST node. The approach produced 99.3% of identical diffs with respect to GumTree.

#### 8.2.10 Benchmarking platform for uniform random sampling

In [26], we present BURST, a benchmarking platform for uniform random sampling techniques. With BURST, researchers have a flexible, controlled environment in which they can evaluate the scalability and uniformity of their sampling. BURST comes with an extensive — and extensible — benchmark dataset comprising 128 feature models, including challenging, real-world models of the Linux kernel. BURST takes as inputs a sampling tool, a set of feature models and a sampling budget. It automatically translates any feature model of the set in DIMACS and invokes the sampling tool to generate the budgeted number of samples. To evaluate the scalability of the sampling tool, BURST measures the time the tool needs to produce the requested sample. To evaluate the uniformity of the produced sample, BURST integrates the state-of-the-art and proven statistical test Barbarik. We envision BURST to become the starting point of a standardisation initiative of sampling tool evaluation. Given the huge interest of research for sampling algorithms and tools, this initiative would have the potential to reach and crosscut multiple research communities including AI, ML, SAT and SPL.

### 8.3 Results for Axis #3: DevSecOps and Resilience Engineering for Software and Systems

**Participants:** Mathieu Acher, Olivier Barais, Arnaud Blouin, Stephanie Challita, Benoît Combemale, Jean-Marc Jézéquel, Olivier Zendra.

### 8.3.1 Fingerprinting and Building Large Reproducible Datasets

Obtaining a relevant dataset is central to conducting empirical studies in software engineering. However, in the context of mining software repositories, the lack of appropriate tooling for large scale mining tasks hinders the creation of new datasets. Moreover, limitations related to data sources that change over time (e.g., code bases) and the lack of documentation of extraction processes make it difficult to reproduce datasets over time. This threatens the quality and reproducibility of empirical studies. In [74], we propose a tool-supported approach facilitating the creation of large tailored datasets while ensuring their reproducibility. We leveraged all the sources feeding the Software Heritage append-only archive which are accessible through a unified programming interface to outline a reproducible and generic extraction process. We propose a way to define a unique fingerprint to characterize a dataset which, when provided to the extraction process, ensures that the same dataset will be extracted. We demonstrate the feasibility of our approach by implementing a prototype. We show how it can help reduce the limitations researchers face when creating or reproducing datasets.

### 8.3.2 Caught in the Game: On the History and Evolution of Web Browser Gaming

Web browsers have come a long way since their inception, evolving from a simple means of displaying text documents over the network to complex software stacks with advanced graphics and network capabilities. As personal computers grew in popularity, developers jumped at the opportunity to deploy cross-platform games with centralized management and a low barrier to entry. Simply going to the right address is now enough to start a game. From text-based to GPU-powered 3D games, browser gaming has evolved to become a strong alternative to traditional console and mobile-based gaming, targeting both casual and advanced gamers. Browser technology has also evolved to accommodate more demanding applications, sometimes even supplanting functions typically left to the operating system. Today, websites display rich, computationally intensive, hardware-accelerated graphics, allowing developers to build ever-more impressive applications and games. In this work [57], we present the evolution of browser gaming and the technologies that enabled it, from the release of the first text-based games in the early 1990s to current open-world and game-engine-powered browser games. We discuss the societal impact of browser gaming and how it has allowed a new target audience to access digital gaming. Finally, we review the potential future evolution of the browser gaming industry.

### 8.3.3 On Understanding Context Modelling for Adaptive Authentication Systems

In many situations, it is of interest for authentication systems to adapt to context ( e.g., when the user's behavior differs from the previous behavior). Hence, representing the context with appropriate and well-designed models is crucial. In [27], we provide a comprehensive overview and analysis of research work on Context Modelling for Adaptive Authentication systems (CM4AA) . To this end, we pursue three goals based on the Systematic Mapping Study (SMS) and Systematic Literature Review (SLR) research methodologies. We first present a SMS to structure the research area of CM4AA ( goal 1 ). We complement the SMS with a SLR to gather and synthesise evidence about context information and its modelling for adaptive authentication systems ( goal 2 ). From the knowledge gained from goal 2, we determine the desired properties of the context information model and its use for adaptive authentication systems ( goal 3 ). Motivated to find out how to model context information for adaptive authentication, we provide a structured survey of the literature to date on CM4AA and a classification of existing proposals according to several analysis metrics. We demonstrate the ability of capturing a common set of contextual features that are relevant for adaptive authentication systems independent from the application domain. We emphasise that despite the possibility of a unified framework, no standard for CM4AA exists.

### 8.3.4 Uncertainty-aware Simulation of Adaptive Systems

Adaptive systems manage and regulate the behavior of devices or other systems using control loops to automatically adjust the value of some measured variables to equal the value of a desired set-point. These systems normally interact with physical parts or operate in physical environments, where uncertainty is unavoidable. Traditional approaches to manage that uncertainty use either robust control algorithms that consider bounded variations of the uncertain variables and worst-case scenarios, or adaptive control methods that estimate the parameters and change the control laws accordingly. In this work [35] we propose to include the sources of uncertainty in the system models as first-class entities using random variables, in order to simulate adaptive and control systems more faithfully, including not only the use of random variables to represent and operate with uncertain values, but also to represent decisions based on their comparisons. Two exemplar systems are used to illustrate and validate our proposal.

### 8.3.5 Open-source software supply chain security

Open-source software supply chain attacks aim at infecting downstream users by poisoning open-source packages. The common way of consuming such artifacts is through package repositories and the development of vetting strategies to detect such attacks is ongoing research. Despite its popularity, the Java ecosystem is the less explored one in the context of supply chain attacks. In this work [36], we study simple-yet-effective indicators of malicious behavior that can be observed statically through the analysis of Java bytecode. Then we evaluate how such indicators and their combinations perform when detecting malicious code injections. We do so by injecting three malicious payloads taken from real-world examples into the Top-10 most popular Java libraries from libraries.io. We found that the analysis of strings in the constant pool and of sensitive APIs in the bytecode instructions aids in the task of detecting malicious Java packages by significantly reducing the information, thus, making also manual triage possible.

In this context of Supply chain attacks on open-source projects, recent work systematized the knowledge about such attacks and proposed a taxonomy in the form of an attack tree [51]. We propose a visualization tool called Risk Explorer [36] for Software Supply Chains, which allows inspecting the taxonomy of attack vectors, their descriptions, references to real-world incidents and other literature, as well as information about associated safeguards. Being open-source itself, the community can easily reference new attacks, accommodate for entirely new attack vectors or reflect the development of new safeguards. This tool is also available online [1]

Current software supply chains heavily rely on open-source packages hosted in public repositories. Given the popularity of ecosystems like npm and PyPI, malicious users started to spread malware by publishing open-source packages containing malicious code. Recent works apply machine learning techniques to detect malicious packages in the npm ecosystem. However, the scarcity of samples poses a challenge to the application of machine learning techniques in other ecosystems. Despite the differences between JavaScript and Python, the open-source software supply chain attacks targeting such languages show noticeable similarities (e.g., use of installation scripts, obfuscated strings, URLs). In this work [52], we present a novel approach that involves a set of language-independent features and the training of models capable of detecting malicious packages in npm and PyPI by capturing their commonalities. This methodology allows us to train models on a diverse dataset encompassing multiple languages, thereby overcoming the challenge of limited sample availability. We evaluate the models both in a controlled experiment (where labels of data are known) and in the wild by scanning newly uploaded packages for both npm and PyPI for 10 days. We find that our approach successfully detects malicious packages for both npm and PyPI. Over an analysis of 31,292 packages, we reported 58 previously unknown malicious packages (38 for npm and 20 for PyPI), which were consequently removed from the respective repositories.

The increasing popularity of certain programming languages has spurred the creation of ecosystem-specific package repositories and package managers. Such repositories (e.g., npm, PyPI) serve as public databases that users can query to retrieve packages for various functionalities, whereas package managers automatically handle dependency resolution and package installation on the client side. These mechanisms enhance software modularization and accelerate implementation. However, they have become a target for malicious actors seeking to propagate malware on a large scale. In this work [53], we show how attackers can leverage capabilities of popular package managers and languages to achieve arbitrary

[1] Risk explorer web site

code execution on victim machines, thereby realizing open-source software supply chain attacks. Based on the analysis of 7 ecosystems, we identify 3 install-time and 4 runtime techniques, and we provide recommendations describing how to reduce the risk when consuming third-party dependencies. We will provide proof-of-concepts that demonstrate the identified techniques. Furthermore, we describe evasion strategies employed by attackers to circumvent detection mechanisms.

#### 8.3.6 Efficient Resource Management for Adaptive Software

Serverless is a trending service model for cloud computing. It shifts a lot of the complexity from customers to service providers. However, current serverless platforms mostly consider the provider's infrastructure as homogeneous, as well as the users' requests. This limits possibilities for the provider to leverage heterogeneity in their infrastructure to improve function response time and reduce energy consumption. We propose a heterogeneity-aware serverless orchestrator for private clouds that consists of two components: the autoscaler allocates heterogeneous hardware resources (CPUs, GPUs, FPGAs) for function replicas, while the scheduler maps function executions to these replicas. Our objective is to guarantee function response time, while enabling the provider to reduce resource usage and energy consumption. This work [54] considers a case study for a deepfake detection application relying on CNN inference. We devised a simulation environment that implements our model and a baseline Knative orchestrator, and evaluated both policies with regard to consolidation of tasks, energy consumption and SLA penalties. Experimental results show that our platform yields substantial gains for all those metrics, with an average of 35% less energy consumed for function executions while consolidating tasks on less than 40% of the infrastructure's nodes, and more than 60% less SLA violations.

We also demonstrated that by proactively caching functions from the same application using adequate storage on the same nodes, we seek to minimize cold starts and data movement to improve total response times. We evaluate our platform in a simulation environment using workload traces derived from Microsoft's Azure Functions, enriched with measurements from a deepfake detection project at the B<>com Institute of Research and Technology [79].

#### 8.3.7 GDPR Enforcement By The Operating System

Currently, it is very hard for companies driven by personal data to make their applications GDPR-compliant, especially if those applications were developed before the GDPR was established. In [59], we present rgpdOS, a GDPR-aware operating system that aims to bring GDPR-compliance to every application, while requiring minimal changes to application code.

#### 8.3.8 Model-Based DevOps: Foundations and Challenges

Time-to-market and continuous improvement are key success indicators to deliver for Industry 4.0 Cyber-Physical Systems (CPSs). There is thus a growing interest in adapting DevOps approaches coming from software systems to CPSs. However, CPSs are made not only of software but also of physical parts that need to be monitored at runtime. In [46], we claim that Model-Driven Engineering can facilitate DevOps for CPSs by automatically connecting a CPS design model to its runtime monitoring, in the form of a digital twin.

# 9 Bilateral contracts and grants with industry

## 9.1 Bilateral contracts with industry

**BCOM**

**Participants:** Olivier Barais.

- 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.

**Test4Science**

**Participants:** Benoît Combemale, Arnaud Blouin.

- Partners: Inria/CEA DAM
- Dates: 2023-2026
- Abstract: Test4Science aims to propose a disciplined and tool-supported approach for scientific software testing. Test4Science is a bilateral collaboration (2023-2026), between the CEA DAM/DIF and the DiverSE team at Inria (follow-up of the previous collaboration, aka. Debug4Science, from 2020 to 2022).

**Orange**

**Participants:** Olivier Barais, Benoît Combemale, Stéphanie Chalita.

- Partners: Inria/Orange
- Dates: 2020-2023
- Abstract: We aim at working on the context of adaptive authentification for trying to go beyond risk Scores : Context-Aware Adaptive Authentication

**Obeo**

**Participants:** Benoît Combemale, Arnaud Blouin.

- Partners: UR1/Obéo
- Dates: 2022-2025
- Abstract: Low-code language workbench, Theo Giraudet's PhD Cifre project.

**SAP**

**Participants:** Olivier Barais.

- Partners: UR1/SAP
- Dates: 2021-2024
- Abstract: Research focusing on Open-source software Supply Chain security. Piergiorgio Ladisa's PhD Cifre project.

**CGI**

**Participants:** Olivier Barais, Mathieu Acher, Jean-Marc Jézéquel.

- Partners: UR1/CGI
- Dates: 2023-2026
- Abstract: Research focusing on legacy source code reeingineering using LLM.

# 10 Partnerships and cooperations

## 10.1 International initiatives

### 10.1.1 Associate Teams in the framework of an Inria International Lab or in the framework of an Inria International Program

**ALE**

**Title:** Agile Language Engineering

**Duration:** 2017 -> 2024

**Coordinator:** Tijs van der Storm

**Partners:**

- CWI (The Netherlands)

**Inria contact:** Benoit Combemale

**Summary:** The team contributes to the field of Software Language Engineering, aiming to provide more agility to both language designers and language users. We explore extreme modularity constructs for the development of Domain-Specific Languages, and advanced mechanisms for live modeling.

### 10.1.2 Inria associate team not involved in an IIL or an international program

**RESIST_EA**

**Title:** Resilient Software Science

**Duration:** 2021 ->

**Coordinator:** Arnaud Gotlieb (arnaud@simula.no)

**Partners:**

- SIMULA (Norvège)

**Inria contact:** Mathieu Acher

**Summary:** The Science of Resilient Software (RESIST_EA) intends to create software-systems which can resist failures without significantly degrading their functionality. For several years, creating resilient software-systems has become extremely important in various application domains. For example, in robotics, the deployment of advanced collaborative robots which have to cope with uncertainty and unexpected behaviors while being able to recover from their failures has led to new research challenges. A recent area where these challenges have become pregnant is industrial robotics for car manufacturing where major issues faced by an “excessive automation” have surfaced. For instance, Tesla has struggled with painting, welding, assembling industrial robots in its advanced California car factory since 2018. Generally speaking, Autonomous Software-Systems (AS) such

as self-driving cars, autonomous ships or industrial robots require the development of resilient software-systems as they have to manage unexpected events, such as faults or hazards. The goal of the Associate Team “Resilient Software Science” (and the main innovation of this project) is to explore the Science of resilient software by laying the ground to foundational work on advanced a priori testing methods such as metamorphic testing and a posteriori continuous improvements through digital twins.

## 10.2 International research visitors

### 10.2.1 Visits of international scientists

**Inria International Chair** Gunter Mussbacher has an Inria International Chair, and he is visiting the DiverSE team 4 months per year.

**Other international visits to the team**

**Joerg Kienzle**

**Status** (Professor)

**Institution of origin:** McGill University

**Country:** Canada

**Dates:** July 2024

**Context of the visit:** Collaboration with the team on different papers regarding SLE and Variability.

**Mobility program/type of mobility:** research stay

**Lola Burgueno**

**Status** (Associate Professor)

**Institution of origin:** Malaga University

**Country:** Spain

**Dates:** July 2024

**Context of the visit:** Collaboration with the team on different papers regarding SLE and Variability.

**Mobility program/type of mobility:** research stay

### 10.2.2 Visits to international teams

**Research stays abroad**

Paul Temple visited University of NAMUR in April, July, September, October, December 2023.
Mathieu Acher visited KTH in SWEDEN in June 2023.
Jean-Marc Jézéquel visited University of Montreal in september 2023.

## 10.3 European initiatives

### 10.3.1 Horizon Europe

### HiPEAC

**Participants:** Olivier Zendra, Jean-Marc Jézéquel.

HiPEAC project on cordis.europa.eu

**Title:** High Performance, Edge And Cloud computing

**Duration:** From December 1, 2022 to May 31, 2025

**Partners:**

- INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET AUTOMATIQUE (INRIA), France
- ECLIPSE FOUNDATION EUROPE GMBH (EFE GMBH), Germany
- INSIDE, Netherlands
- UNIVERSITEIT GENT (UGent), Belgium
- RHEINISCH-WESTFAELISCHE TECHNISCHE HOCHSCHULE AACHEN (RWTH AACHEN), Germany
- COMMISSARIAT A L ENERGIE ATOMIQUE ET AUX ENERGIES ALTERNATIVES (CEA), France
- SINTEF AS (SINTEF), Norway
- IDC ITALIA SRL, Italy
- THALES (THALES), France
- CLOUDFERRO SA, Poland
- BARCELONA SUPERCOMPUTING CENTER CENTRO NACIONAL DE SUPERCOMPUTACION (BSC CNS), Spain

**Inria contact:** Olivier Zendra

**Coordinator:** Ghent University

**Summary:** The objective of HiPEAC is to stimulate and reinforce the development of the dynamic European computing ecosystem that supports the digital transformation of Europe. It does so by guiding the future research and innovation of key digital, enabling, and emerging technologies, sectors, and value chains. The longer term goal is to strengthen European leadership in the global data economy and to accelerate and steer the digital and green transitions through human-centred technologies and innovations. This will be achieved via mobilising and connecting European partnerships and stakeholders to be involved in the research, innovation and development of computing and systems technologies. They will provide roadmaps supporting the creation of next-generation computing technologies, infrastructures, and service platforms.

The key aim is to support and contribute to rapid technological development, market uptake and digital autonomy for Europe in advanced digital technology (hardware and software) and applications across the whole European digital value chain. HiPEAC will do this by connecting and upscaling existing initiatives and efforts, by involving the key stakeholders, and by improving the conditions for large-scale market deployment. The next-generation computing and systems technologies and applications developed will increase European autonomy in the data economy. This is required to support future hyper-distributed applications and provide new opportunities for further disruptive digital transformation of the economy and society, new business models, economic growth, and job creation.

The HiPEAC CSA proposal directly addresses the research, innovation, and development of next generation computing and systems technologies and applications. The overall goal is to support the European value chains and value networks in computing and systems technologies across the computing continuum from cloud to edge computing to the Internet of Things (IoT).

## 10.4 National initiatives

### 10.4.1 ANR

**MC-Evo2 ANR JCJC**

**Participants:** Djamel Eddine Khelladi.

- Coordinator: Djamel E. Khelladi
- DiverSE, CNRS/IRISA Rennes
- Dates: 2021-2025
- Abstract: Software maintenance represents 40% to 80% of the total cost of developing software. On 65 projects, an IT company reported a cost of several million dollars, with a 25% higher cost on complex projects. Nowadays, software evolves frequently with the philosophy "Release early, release often" embraced by IT giants like the GAFAM, thus making software maintenance difficult and costly. Developing complex software inevitably requires developers to handle multiple dimensions, such as APIs to use, tests to write, models to reason with, etc. When software evolves, a co-evolution is usually necessary as a follow-up, to resolve the impacts caused by the evolution changes. For example, when APIs evolve, code must be co-evolved, or when code evolves, its tests must be co-evolved. The goals of this project are to: 1) address these challenges from a novel perspective, namely a multidimensional co-evolution approach, 2) investigate empirically the multidimensional co-evolution in practice in GitHub, Maven, and Eclipse, 3) automate and propagate the multidimensional co-evolution between the software code, APIs, tests, and models.

**MBDO**

**Participants:** Jean-Marc Jezequel, Benoit Combemale, Quentin Perez, Didier Vojtisek.

- Coordinator: Jean-Marc Jezequel
- Coordinator: Univ. Rennes
- Partners: Aachen University, University of Stuttgart
- Dates: 2023-2026
- Abstract: Our goal in MBDO is to provide the foundations for a Model-Based DevOps framework unifying these different forms of models in the specific context of cloud-native and IoT systems. The proposed Model-Based DevOps framework would then allow engineers to smoothly go back and forth from Dev time to Ops time by leveraging semi-automatically generated digital twins of their systems.

### 10.4.2 DGA

**LangSpecialize**

**Participants:** Benoît Combemale, Olivier Barais.

- Coordinator: DGA
- Partners: DGA MI, INRIA

- Dates: 2023-2026
- 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 packet 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.

#### 10.4.3 DGAC

**MIP 4.0**

**Participants:** Benoît Combemale, Didier Vojtisek, Olivier Barais.

- Coordinator: Safran
- Partners: Safran, Akka, Inria.
- Dates: 2022-2024
- Abstract: The MIP 4.0 project aims at investigating integrated methods for efficient and shared propulsion systems. Inria explores new techniques for collaborative modeling over the time.

#### 10.4.4 PEPR

**PEPR Cloud Taranis**

**Participants:** Olivier Barais, Paul Temple, Stéphanie Chalitta.

- Coordinator: INRIA
- Partners: INRIA, CNRS, UR.
- Dates: 2023-2030
- Abstract: In order to efficiently exploit new infrastructures, we propose a strategy based on a significant abstraction of the application structure description to further automate application and infrastructure management. Thus, it will be possible to globally optimize the resources used with respect to multi-criteria objectives (price, deadline, performance, energy, etc.) on both the user side (applications) and the provider side (infrastructures). This abstraction also includes the challenges related to the abstraction of application reconfiguration and to automatically adapt the use of resources.

**PEPR Numpex Exasoft**

**Participants:** Benoit Combemale, Olivier Barais.

- Coordinator: INRIA
- Partners: INRIA, CNRS, UR.
- Dates: 2023-2030
- Abstract: The ExaSoft project will study the software stack for future exascale machines (compilers, programming and execution model, monitoring and optimisation tools and energy management.

### 10.4.5 Campus Cyber

**Software Heritage Sec**

**Participants:** Olivier Barais, Mathieu Acher, Djamel Eddine Khelladi, Olivier Zendra.

- Coordinator: INRIA
- Partners: INRIA, IMT, CEA, Université Sorbone.
- Dates: 2023-2027
- Abstract: By analyzing the evolution of software source code over time, researchers and practitioners will be able to gain a better understanding of the vulnerabilities and threats that may exist in software systems, identify potential security risks early on and take proactive measures to mitigate them.

## 10.5 Regional initiatives

**IPSCo**

**Participants:** Benoît Combemale, Didier Vojtisek, Olivier Barais.

- Coordinator: Jamespot
- Partners: Jamespot, UR1, Logpickr.
- Dates: 2022-2023
- Abstract: The IPSCo project aims at investigating new tools and methods to bring intelligence into processes and communities.

**SAD CoEvoMP**

**Participants:** Djamel Eddine Khelladi, Benoît Combemale, Arnaud Blouin.

- Coordinator: Djamel E. Khelladi
- Partners: CNRS, UR1.
- Dates: 12/2022-12/2024
- Abstract: The CoEvoMP project aims at investigating polyglot co-evolution in parallel of the MC-Evo2 project.

**Privacy Bugs**

**Participants:** Johann Bourcier, Walter Rudametkin.

- Coordinator: Johann Bourcier
- Partners: UR1, DGA.

- Dates: 2023-2026
- Creach labs funding
- Abstract: The Privacy Bugs project aims at investigating new tools and methods to detect and quantify privacy bugs in frontend web applications.

# 11 Dissemination

**Participants:** Djamel Khelladi, Gunter Mussbacher, Olivier Zendra, Olivier Barais, Mathieu Acher, Aymeric Blot, Arnaud Blouin, Johann Bourcier, Stéphanie Challita, Benoît Combemale, Jean-Marc Jezequel, Quentin Perez, Noël Plouzeau, Walter Rudametkin Ivey, Paul Temple, Gwendal Jouneaux, Faezeh Khorram, Quentin Perez, Xhevahire Ternava, Lina Bilal, Ewen Brune, Anne Bumiller, Theo Giraudet, Philemon Houdaille, Gwendal Jouneaux, Zohra Kebaili, N'Guessan Hermann Kouadio, Piergiorgio Ladisa, Clement Lahoche, Quentin Le Dilavrec, Romain Lefeuvre, Georges Aaron Randrianaina, Chiara Relevat, Sterenn Roux, Florian Badie, Romain Belafia, Emmanuel Chebbi, Guy De Spiegeleer, Quentin Le Dilavrec, Romain Lefeuvre, Charly Reux, Didier Vojtisek.

## 11.1 Promoting scientific activities

### 11.1.1 Scientific events: organisation

**General chair, scientific chair** Benoit Combemale and Gunter Mussbacher have been General co-chairs of ICT4S 2023 [70, 71], organized at the University of Rennes, France.

**Member of the organizing committees** Olivier Barais and Djamel Eddine Khelladi have co-organized "*Les Journées du GDR GPL 2023*".

Arnaud Blouin has co-organized the:

- HuFaMo Workshop co-organizer at MODELS 2023
- 2023; MLE Workshop co-organizer at MODELS 2023
- GL-IHM Workshop co-organizer at IHM 2023 (French-Speaking conference)

Djamel Eddine Khelladi has co-organized the Models and Evolution (ME) workshop at MODELS.
Stéphanie Challita has been Student Volunteers Co-Chair at ICT4S'2023.
Quentin Perez has been Student Virtual Chair at ICT4S'2023.
Johann Bourcier has been Publicity chair at ICT4S'2023

### 11.1.2 Scientific events: selection

**Member of the conference program committees** Arnaud Blouin has been a member of the following PCs:

- 15th ACM SIGCHI symposium on Engineering interactive computing systems (EICS 2023), 2023
- The 38th ACM/SIGAPP Symposium on Applied Computing (SAC), software engineering track, 2023

Olivier Barais has been a member of the following PCs:

- MODELS 2023 ACM/IEEE 26th International Conference on Model Driven Engineering Languages and Systems, Västerås, Sweden, Oct 1 - Oct 6, 2023

Benoit Combemale has been a member of the following PCs:

- MODELS 2023 ACM/IEEE 26th International Conference on Model Driven Engineering Languages and Systems (MODELS 2023, Program Board)
- ECMFA 2023 The 19th European Conference on Modelling Foundations and Applications (ECMFA 2023)
- ModDiT'23 workshop at MODELS'23
- 3rd Eclipse Security, AI, Architecture and Modelling Conference on Cloud to Edge Continuum

Olivier Zendra has been a member of the PC of the 30th CE&SAR conference, CE&SAR 2023 by DGA, at ECW 2023.

Jean-Marc Jézéquel has been a member of the following PCs:

- SEAMS 2023 18th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, May 23-24, 2023, Melbourne, Australia, co-located event with ICSE.
- SPLC 2023 The 26th International Software Product Line Conference (Industry Track), August 27-31, 2023, Tokyo, Japan
- MODELS 2023 ACM/IEEE 26th International Conference on Model Driven Engineering Languages and Systems, Västerås, Sweden, Oct 1 - Oct 6, 2023

Djamel Eddine Khelladi has been a member of the following PCs:

- MoDDiT'23, ME'23, MLE'23 at MODELS
- ESEC/FSE'23 Artifacts
- VariVolution'23 at SPLC

**Reviewer** Arnaud Blouin has served as an external reviewer for Interact 2023.

### 11.1.3 Journal

**Member of the editorial boards** Benoit Combemale is Editor-in-Chief of the Springer-Nature International Journal on Software and Systems Modeling (SoSyM) [33, 34, 28, 31, 30, 32, 29]. He is also member of the Editorial Boards of the Springer Software Quality Journal (SQJ), the platinum open access JOT journal (former deputy editor-in-chief of the journal, 2020-2023), and the Elsevier Journal of Computer Languages (COLA).

Jean-Marc Jézéquel has been Associate Editor in Chief of IEEE Computer and of SoSYM, as well as a member of the EB of JSS.

Stéphanie Challita has been Assistant Editor of the Journal of Software and Systems Modeling, Springer (SoSyM).

Djamel Eddine Khelladi is the guest editor in the special issue on Model Driven Engineering for Digital Twins in the the Journal of Software and Systems Modeling(SoSyM).

**Reviewer - reviewing activities** Arnaud Blouin has served as an external reviewer for IEEE Transactions of Software Engineering.

Stéphanie Challita has been a reviewer at Annals of Telecommunications and SoSyM.

Olivier Barais has been a reviewer at SoSyM.

Benoit Combemale has served as an external reviewer for IEEE Transactions of Software Engineering, and ACM Transactions on Software Engineering and Methodology.

Djamel Eddine Khelladi has served as an external reviewer for the Journal on Software and Systems Modeling (SoSyM) and IEEE Transactions of Software Engineering, and ACM Transactions on Software Engineering and Methodology.

#### 11.1.4 Invited talks

Benoit Combemale gave a talk entitled "Expériences et défis scientifiques des jumeaux numériques." at the network Aristote (21/09/23).

#### 11.1.5 Leadership within the scientific community

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

Jean-Marc Jézéquel has been Vice-President of Informatics Europe, and elected as the new President starting 2024.

Olivier Zendra is:

- Founder and a member of the Steering Committee of ICOOOLPS (International Workshop on Implementation, Compilation, Optimization of OO Languages, Programs and Systems).
- Member of the EU HiPEAC CSA project Steering Committee
- Member of the HiPEAC Vision Editorial Board

Benoit Combemale is a founding member of the GEMOC initiative, an international effort to develop techniques, frameworks, and environments to facilitate the creation, integration, and automated processing of heterogeneous modeling languages. He is currently the scientific leader of the Research Consortium GEMOC at the Eclipse Foundation.

Benoit Combemale is also a member of the steering committees of the ACM/IEEE Intl. Conference on Model-Driven Engineering Languages and Systems (member since 2023), the ACM SIGPLAN Intl. Conference on Software Language Engineering (member since 2014, and chair of the steering committee from 2018 to 2022), the Intl. Conference on Information and Communications Technology for Sustainability (member since 2022), the Modeling Language Engineering and Execution (MLE) workshop (founding member, since 2019) and the Model-Driven Engineering of Digital Twins (ModDiT) workshop (founding member, since 2021).

#### 11.1.6 Scientific expertise

Arnaud Blouin: expert for the CIR agency (research tax credit, "crédit d'impôt recherche").

Olivier Barais: expert for the following call for projects:

- PHC FRANCE - GRECE 2023
- STIC AmSud 2023
- CONTRAT DOCTORAL UFA 2023
- VINCI 2023 Chapitre 2 – Aides à la mobilité pour thèses en cotutelle
- Bourses Irlande 2023
- SAMUEL DE CHAMPLAIN - Formation 2023
- PHC ALLIANCE GRANDE BRETAGNE 2023
- PHC CARLOS J. FINLAY CUBA 2023
- PHC ALLIANCE GRANDE BRETAGNE 2023
- Workshops Maupertuis Finlande 2023
- PHC FASIC ECPD AUSTRALIE 2023
- PHC BOSPHORE TURQUIE 2023
- PHC CEDRE LIBAN 2023

- PHC GALILEE ITALIE 2023
- PHC UTIQUE TUNISIE 2023
- PHC BOSPHORE TURQUIE 2023
- PHC TOUBKAL MAROC 2023
- PHC UTIQUE TUNISIE 2023

Olivier Barais: member of the scientific board of Pole de compétitivité Image et Réseau

Stéphanie Challita has been a member of the Conference Activities Committee (CAC) at IEEE Computer Society.

Olivier Zendra: scientific CIR/JEI expert for the MESR.

Johann Bourcier: expert for the CIR agency (research tax credit, "crédit d'impôt recherche") and reviewer for an ANR JCJC.

#### 11.1.7 Research administration

Olivier Barais is a new member of the CNU 27.

Olivier Zendra was a member of Inria Evaluation Committee (CE) till September 2023.

### 11.2 Teaching - Supervision - Juries

#### 11.2.1 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, Sustainable Software Engineering) and for the second year of the MSc in software engineering (Model driven Engineering, DevOps, DevSecOps, Validation & Verification, *etc.*).

Each of Jean-Marc Jézéquel, Noël Plouzeau, Olivier Barais, Benoît Combemale, Johann Bourcier, Arnaud Blouin, Aymeric Blot, Quentin Perez, Stéphanie Challita and Mathieu Acher teaches about 250h in these domains for a grand total of about 2000 hours, including several courses at IMT, 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 Master in Computer Science at the University of Rennes 1.

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.

#### 11.2.2 Supervision

- Piergiorgio Ladisa, CIFRE with SAP (defense in 2024). Olivier Barais is the supervisor of this thesis.
- Anne Bumiller, CIFRE with Orange (defense in 2023). Benoît Combemale, Stéphanie Chalita and Olivier Barais are co-supervisors of this thesis.
- Theo Giraudet, CIFRE with Obéo (defense in 2025). Benoît Combemale and Arnaud Blouin are co-supervisors of this thesis.
- Georges Aaron Randrianaina, (defense in 2024). Mathieu Acher, Djamel Eddine Khelladi and Olivier Zendra are co-supervisors of this thesis.
- Quentin Le Dilavrec, (defense in 2024). Djamel Eddine Khelladi and Aranaud Blouin are co-supervisors of this thesis.

- Gwendal Jouneaux, (defense in 2024). Benoît Combemale and Olivier Barais are co-supervisors of this thesis.
- Lina Bilal (defense in 2026). Benoît Combemale and Jean-Marc Jézéquel are co-supervisors of this thesis.
- Romain Lefeuvre (defense in 2026). Benoît Combemale and Quentin Perez are co-supervisors of this thesis.
- Houdaille Philémon (defense in 2026). Benoît Combemale and Djamel Eddine Khelladi are co-supervisors of this thesis.
- Sterenn Roux (defense in 2026). Johann Bourcier and Walter Rudametkin are co-supervisors of this thesis.
- Kaouter Zohra Kebaili (defense in 2025). Djamel Eddine Khelladi and Mathieu Acher and Olivier Barais are co-supervisors of this thesis.
- Ewen Brune (defense in 2026). Benoît Combemale and Arnaud Blouin are co-supervisors of this thesis.
- Clément Lahoche (defense in 2026). Olivier Barais and Olivier Zendra are co-supervisors of this thesis.
- N'Guessan Hermann Kouadio (defense in 2026). Olivier Barais and Mathieu Acher are co-supervisors of this thesis.
- Chiara Relevat (defense in 2026). Benoit Combemale and Gurvan Le Guernic are co-supervisors of this thesis.

### 11.2.3 Juries

Benoit Combemale was in the PhD jury (reviewer) of Hamza Bourbouh (ISAE-Supaéro), "Static analyses and model checking of mixed data-flow/control-flow models for critical systems."

Arnaud Blouin was in the PhD jury (reviewer) of Philippe Schmid (University of Lille, Inria Lille, France), "Développement d'historiques de commandes avancés pour améliorer le processus d'édition numérique"

Olivier Barais was in the PhD jury (reviewer) of Romain Fouquet (University of Lille, Inria Lille, France), "Improving Web User Privacy Through Content Blocking"

Olivier Barais was in the PhD jury (reviewer) of Santiago Bragagnolo (University of Lille, Inria Lille, France), "An Holistic Approach to Migrate Industrial Legacy Systems"

Walter Rudametkin was in the PhD jury (reviewer) of Vero Sosnovik (University of Grenoble, France), "Detection and analysis of online issue and political ads".

Walter Rudametkin was in the PhD jury (reviewer) of Anne Josiane Kouam (Institut polytechnique de Paris) Bypass frauds in cellular networks : Understanding and Mitigation

Mathieu Acher was in the PhD jury (president/examiner) of Adrien GOUGEON (Université de Rennes) "Optimizing a Dynamic and Energy Efficient Network Piloting the Electrical Grid".

Mathieu Acher was invited in the PhD jury (invited) of César Soto Valero (KTH, Sweden) "Debloating Java Dependencies".

## 11.3 Popularization

### 11.3.1 Internal or external Inria responsibilities

Olivier Zendra, as a member of the HiPEAC Vision Editorial Board, contributed to the writing of the overall **HiPEAC Vision 2023** [63], also leading the writing of the cybersecurity chapter [69].

### 11.3.2 Articles and contents

**HiPEAC** Olivier Zendra, as a member of the HiPEAC Vision Editorial Board, contributed to the writing of the overall **HiPEAC Vision 2023** [63], the focus of which is that we are in a race, both against time and with the rest of the world. Indeed, technology never stands still. The last few years have once again seen rapid, profound changes across the world, both from the technological point of view – with impressive advances in artificial intelligence – and from the geopolitical point of view, where technology is increasingly seen as a strategic asset. Different world regions are competing for leadership in several areas. Competition between the United States (US) and China in the technology and artificial intelligence (AI) domains is particularly fierce, and it is becoming more intense. This creates a threat to Europe, but at the same time an opportunity. The recent change of ownership and leadership at Twitter is also a wake-up call for Europe. Many of the essential services the European society depends on run on platforms that are not controlled by Europe. This creates vulnerabilities in the event of conflict, comparable to European dependency on Russian energy. These are just the evolutions of the last year. Change is taking place so rapidly that it is also having an impact on the HiPEAC Vision: updating it every two years is no longer sufficient to keep up with the speed of the evolution of computing systems. Therefore, from now on, there will be a HiPEAC Vision every year. The speed of the evolution has also inspired the editorial board to present the challenges of our community as six leadership races: for the “next web”, for AI, for innovative hardware solutions, for cybersecurity, for digital sovereignty, and for sustainability solutions. Structurally, the HiPEAC Vision 2023 has two parts: First, a set of recommendations for the HiPEAC community at large. Second, a set of articles written by experts and grouped into six chapters each describing one “global leadership race”.

In the the HiPEAC Vision 2023, Olivier Zendra was also specifically in charge of the **Cybersecurity chapter** [69], that addresses "The race for cybersecurity", in which we explain that, after decades of digitalization spreading into every area of our lives, with very little attention given to the aspects linked to cybersecurity, information technology (IT) had essentially become an “open bar” for cybercriminals. For a few years, with a marked degradation during the peak of the COVID-19 pandemic, the news has been rife with reports of privacy breaches and cyberattacks (mainly ransomware) on companies and institutions, especially local governments and hospitals. In addition, cyberwarfare has been making the news too, especially in relation to the conflict in Ukraine. Thus, the era of blissful ignorance and naiveté has ended. Although the wake-up call was abrupt, knowledge of these issues has expanded, and governments and to some extent businesses have taken first moves to enhance the cybersecurity frontline. However, cybersecurity is a highly competitive race between nations, between defenders and attackers, with enormous stakes. The pervasiveness of IT provides a broad attack surface, and attacks can be economically devastating, but they can also have tangible or even lethal repercussions on the physical world. Despite several highly acclaimed advancements (e.g. the General Data Protection Regulation-GDPR), the EU still has a great deal of work to do in this regard, particularly to maintain its sovereignty and become a leader in the global competition. Cybersecurity is indeed a matter of both economic leadership and national sovereignty. This chapter contains two contributions. In article “From cybercrime to cyberwarfare, nobody can overlook cybersecurity any more” [68], we describe the current state of IT system cybersecurity, showing how vulnerable systems are to the numerous dangers and challenges posed by cybercrime and cyberwarfare. It goes on to present a few concrete ways to remedy the issue, whether by technical, legal, sociological, or political means. Indeed, although the EU has weaknesses, linked to its extremely high reliance on IT systems, it also has the potential to become a world leader in cybersecurity, owing to both its strong technical culture and its regulatory capabilities. In article “Is privacy possible in a digital world?” [64], we explain that over the last few years, privacy has become a hot topic. However, this is in large part due to the fact that ever more data is being collected, not only by governments, but also by companies. It is often unclear for which purposes this data ends up being used; worse, it can even be leaked to third parties by attackers. Furthermore, even if this collected data would appear not to be sensitive in and of itself, sometimes sensitive information can be deduced from it. In this article, we present a summary of some of the ways in which data is gathered; how additional information can be inferred from it and how this is problematic; and how we can try to protect our privacy.

**Data Protection: GDPR Enforcement By The Operating System** The challenges of effective data protection cannot be addressed solely by law. The demonstrated need for an alliance with technology has

led to the project of building an operating system incorporating data protection rules because it serves as the intermediary between processing and data. Three central ideas structure the technical project (the creation of active personal data, the focus of the OS on data rather than the process, and access at the level of the data itself), suggesting an implementation of data protection rules at the level of each personal data [40]. Such innovations do not fail to pose challenges, both in terms of the choices made and the translation of legal rules. However, the project involves more fundamental issues at the micro level – the modularity of personal data characteristics – as well as at the macro level – the correlative modification of the data processing ecosystem.

#### 11.3.3 Education

In the field of education, the team is applying some of its research results to develop a test scoring platform for Universities: CorrectExam. This platform is gaining users, with more than 150 exams marked on the platform. Discussions are underway to incubate the platform within the esup-portail association.

Olivier Barais gave several talks on the use of ChatGPT for education in front of a set of high-school professors and the dean.

#### 11.3.4 Interventions

As Professors, we introduce into our course a set of shared slides raising awareness of the academic community and associated discipline. We might regret that a laboratory like the one in Rennes is not easily open to our Masters students. This does not help in reducing the gap between academia and industry. Indeed, many students can follow university or engineering degrees without going to the laboratory.

## 12 Scientific production

### 12.1 Major publications


[1] M. Acher, R. E. Lopez-Herrejon and R. Rabiser. 'Teaching Software Product Lines: A Snapshot of Current Practices and Challenges'. In: *ACM Transactions of Computing Education* (May 2017). URL: https://hal.inria.fr/hal-01522779.

[2] A. Blouin, V. Lelli, B. Baudry and F. Coulon. 'User Interface Design Smell: Automatic Detection and Refactoring of Blob Listeners'. In: *Information and Software Technology* 102 (May 2018), pp. 49–64. DOI: 10.1016/j.infsof.2018.05.005. URL: https://hal.inria.fr/hal-01499106.

[3] M. Boussaa, O. Barais, G. Sunyé and B. Baudry. 'Leveraging metamorphic testing to automatically detect inconsistencies in code generator families'. In: *Software Testing, Verification and Reliability* (Dec. 2019). DOI: 10.1002/stvr.1721. URL: https://hal.inria.fr/hal-02422437.

[4] E. Bousse, D. Leroy, B. Combemale, M. Wimmer and B. Baudry. 'Omniscient Debugging for Executable DSLs'. In: *Journal of Systems and Software* 137 (Mar. 2018), pp. 261–288. DOI: 10.1016/j.jss.2017.11.025. URL: https://hal.inria.fr/hal-01662336.

[5] B. Combemale, J. Deantoni, B. Baudry, R. B. France, J.-M. Jézéquel and J. Gray. 'Globalizing Modeling Languages'. In: *IEEE Computer* (June 2014), pp. 10–13. URL: https://hal.inria.fr/hal-00994551.

[6] K. Corre, O. Barais, G. Sunyé, V. Frey and J.-M. Crom. 'Why can't users choose their identity providers on the web?' In: *Proceedings on Privacy Enhancing Technologies* 2017.3 (Jan. 2017), pp. 72–86. DOI: 10.1515/popets-2017-0029. URL: https://hal.archives-ouvertes.fr/hal-01611048.

[7] J.-E. Dartois, J. Boukhobza, A. Knefati and O. Barais. 'Investigating Machine Learning Algorithms for Modeling SSD I/O Performance for Container-based Virtualization'. In: *IEEE transactions on cloud computing* 14 (2019), pp. 1–14. DOI: 10.1109/TCC.2019.2898192. URL: https://hal.inria.fr/hal-02013421.

[8] J.-M. Davril, E. Delfosse, N. Hariri, M. Acher, J. Clelang-Huang and 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)*. Sept. 2013, pp. 290–300. DOI: 10.1145/2491411.2491455. URL: https://hal.inria.fr/hal-00859475.

[9] T. Degueule, B. Combemale, A. Blouin, O. Barais and 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)*. Oct. 2015. URL: https://hal.inria.fr/hal-01197038.

[10] D. Foures, M. Acher, O. Barais, B. Combemale, J.-M. Jézéquel and J. Kienzle. 'Experience in Specializing a Generic Realization Language for SPL Engineering at Airbus'. In: MODELS 2023 - 26th International Conference on Model-Driven Engineering Languages and Systems. Västerås, Sweden: IEEE, 2023, pp. 1–12. URL: https://inria.hal.science/hal-04216627.

[11] J. A. Galindo Duarte, M. Alférez, M. Acher, B. Baudry and 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. URL: https://hal.inria.fr/hal-01003148.

[12] I. Gonzalez-Herrera, J. Bourcier, E. Daubert, W. Rudametkin, O. Barais, F. Fouquet, J.-M. Jézéquel and 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. URL: https://hal.inria.fr/hal-01354999.

[13] A. Halin, A. Nuttinck, M. Acher, X. Devroey, G. Perrouin and 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. URL: https://hal.inria.fr/hal-01829928.

[14] J.-M. Jézéquel, B. Combemale, O. Barais, M. Monperrus and F. Fouquet. 'Mashup of Meta-Languages and its Implementation in the Kermeta Language Workbench'. In: *Software and Systems Modeling* 14.2 (2015), pp. 905–920. URL: https://hal.inria.fr/hal-00829839.

[15] D. E. Khelladi, B. Combemale, M. Acher and O. Barais. 'On the Power of Abstraction: a Model-Driven Co-evolution Approach of Software Code'. In: *42nd International Conference on Software Engineering, New Ideas and Emerging Results*. Séoul, South Korea, May 2020. URL: https://hal.inria.fr/hal-03029426.

[16] D. E. Khelladi, B. Combemale, M. Acher, O. Barais and J.-M. Jézéquel. 'Co-Evolving Code with Evolving Metamodels'. In: ICSE 2020 - 42nd International Conference on Software Engineering. Séoul, South Korea, 6th July 2020, pp. 1–13. URL: https://hal.inria.fr/hal-03029429.

[17] P. Laperdrix, W. Rudametkin and 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. URL: https://hal.inria.fr/hal-01285470.

[18] Q. Le Dilavrec, D. E. Khelladi, A. Blouin and J.-M. Jézéquel. 'HyperAST: Enabling Efficient Analysis of Software Histories at Scale'. In: ASE 2022 - 37th IEEE/ACM International Conference on Automated Software Engineering. Oakland, United States: IEEE, 10th Oct. 2022, pp. 1–12. URL: https://hal.inria.fr/hal-03764541.

[19] M. Leduc, T. Degueule, E. Van Wyk and B. Combemale. 'The Software Language Extension Problem'. In: *Software and Systems Modeling* (2019), pp. 1–4. URL: https://hal.inria.fr/hal-02399166.

[20] H. Martin, M. Acher, J. A. Pereira, L. Lesoil, J.-M. Jézéquel and D. E. Khelladi. 'Transfer Learning Across Variants and Versions: The Case of Linux Kernel Size'. In: *IEEE Transactions on Software Engineering* 48.11 (1st Nov. 2022), pp. 4274–4290. DOI: 10.1109/TSE.2021.3116768. URL: https://hal.inria.fr/hal-03358817.

[21] G. A. Randrianaina, X. Tërnava, D. E. Khelladi and M. Acher. 'On the Benefits and Limits of Incremental Build of Software Configurations: An Exploratory Study'. In: ICSE 2022 - 44th International Conference on Software Engineering. Pittsburgh, Pennsylvania / Virtual, United States, 8th May 2022, pp. 1–12. URL: https://hal.science/hal-03547219.

[22] M. Rodriguez-Cancio, B. Combemale and B. Baudry. 'Automatic Microbenchmark Generation to Prevent Dead Code Elimination and Constant Folding'. In: *Proc. of the Int. Conf. on Automated Software Engineering (ASE)*. Sept. 2016. URL: https://hal.inria.fr/hal-01343818.

[23] P. Temple, M. Acher, J.-M. Jezequel and O. Barais. 'Learning-Contextual Variability Models'. In: *IEEE Software* 34.6 (Nov. 2017), pp. 64–70. DOI: 10.1109/MS.2017.4121211. URL: https://hal.inria.fr/hal-01659137.

[24] P. Temple, M. Acher and 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. URL: https://hal.inria.fr/hal-02177158.

[25] P. Temple, G. Perrouin, M. Acher, B. Biggio, J.-M. Jézéquel and F. Roli. 'Empirical Assessment of Generating Adversarial Configurations for Software Product Lines'. In: *Empirical Software Engineering* (Dec. 2020), pp. 1–57. URL: https://hal.inria.fr/hal-03045797.

## 12.2 Publications of the year

**International journals**

[26] M. Acher, G. Perrouin and M. Cordy. 'BURST: Benchmarking Uniform Random Sampling Techniques'. In: *Science of Computer Programming* 226 (3rd Jan. 2023), pp. 1–10. DOI: 10.1016/j.scico.2022.102914. URL: https://inria.hal.science/hal-03897639.

[27] A. Bumiller, S. Challita, B. Combemale, O. Barais, N. Aillery and G. Le Lan. 'On Understanding Context Modelling for Adaptive Authentication Systems'. In: *ACM Transactions on Autonomous and Adaptive Systems* (1st Jan. 2023), pp. 1–36. DOI: 10.1145/3582696. URL: https://hal.science/hal-04037520.

[28] S. Challita, B. Combemale, H. Ergin, J. Gray, B. Rumpe and M. Schindler. 'Report on the State of the SoSyM Journal end of 2022'. In: *Software and Systems Modeling* 22.1 (Feb. 2023), pp. 1–7. DOI: 10.1007/s10270-023-01085-6. URL: https://inria.hal.science/hal-04216688.

[29] B. Combemale, J. Gray and B. Rumpe. 'Adopting the concept of a function as an underlying semantic paradigm for modeling languages'. In: *Software and Systems Modeling* 22.6 (30th Nov. 2023), pp. 1733–1735. DOI: 10.1007/s10270-023-01140-2. URL: https://inria.hal.science/hal-04425740.

[30] B. Combemale, J. Gray and B. Rumpe. 'ChatGPT in software modeling'. In: *Software and Systems Modeling* 22.3 (11th May 2023), pp. 777–779. DOI: 10.1007/s10270-023-01106-4. URL: https://inria.hal.science/hal-04425731.

[31] B. Combemale, J. Gray and B. Rumpe. 'How to define modeling languages?' In: *Software and Systems Modeling* 22.2 (2023), pp. 449–451. DOI: 10.1007/s10270-023-01098-1. URL: https://hal.science/hal-04128248.

[32] B. Combemale, J. Gray and B. Rumpe. 'Large language models as an "operating" system for software and systems modeling'. In: *Software and Systems Modeling* 22.5 (16th Sept. 2023), pp. 1391–1392. DOI: 10.1007/s10270-023-01126-0. URL: https://inria.hal.science/hal-04425734.

[33] B. Combemale, J. Gray and B. Rumpe. 'Research software engineering and the importance of scientific models'. In: *Software and Systems Modeling* 22.4 (Aug. 2023), pp. 1081–1083. DOI: 10.1007/s10270-023-01119-z. URL: https://inria.hal.science/hal-04216671.

[34] B. Combemale, R. Eramo and J. de Lara. 'Guest editorial for the theme section on modeling language engineering'. In: *Software and Systems Modeling* 22.3 (Mar. 2023), pp. 795–796. DOI: 10.1007/s10270-023-01097-2. URL: https://inria.hal.science/hal-04216676.

[35] J.-M. Jézéquel and A. Vallecillo. 'Uncertainty-aware Simulation of Adaptive Systems'. In: *ACM Transactions on Modeling and Computer Simulation* (28th Mar. 2023), pp. 1–18. DOI: 10.1145/3589517. URL: https://inria.hal.science/hal-04064771.

[36] P. Ladisa, S. E. Ponta, A. Sabetta, M. Martinez and O. Barais. 'Journey to the Center of Software Supply Chain Attacks'. In: *IEEE Security and Privacy Magazine* 21.6 (Nov. 2023), pp. 34–49. DOI: 10.1109/MSEC.2023.3302066. URL: https://inria.hal.science/hal-04423786.

[37] L. Lesoil, M. Acher, A. Blouin and J.-M. Jézéquel. 'Input Sensitivity on the Performance of Configurable Systems: An Empirical Study'. In: *Journal of Systems and Software* (2023), pp. 1–18. DOI: 10.1016/j.jss.2023.111671. URL: https://inria.hal.science/hal-03476464.

[38] L. Lesoil, H. Spieker, A. Gotlieb, M. Acher, P. Temple, A. Blouin and J.-M. Jézéquel. 'Learning input-aware performance models of configurable systems: An empirical evaluation'. In: *Journal of Systems and Software* (Nov. 2023), p. 111883. DOI: 10.1016/j.jss.2023.111883. URL: https://hal.science/hal-04271476.

[39] G. Mussbacher, B. Combemale, J. Kienzle, L. Burgueño, A. Garcia-Dominguez, J.-M. Jézéquel, G. Jouneaux, D.-E. Khelladi, S. Mosser, C. Pulgar, H. Sahraoui, M. Schiedermeier and T. van der Storm. 'Polyglot Software Development: Wait, What?' In: *IEEE Software* (2024), pp. 1–8. DOI: 10.1109/MS.2023.3347875. URL: https://inria.hal.science/hal-04383286.

[40] L. Pailler, A. Tchana and B. Combemale. 'Protéger les données jusqu'à l'OS'. In: *Dalloz IP/IT : droit de la propriété intellectuelle et du numérique* (2023). URL: https://inria.hal.science/hal-04216866.

[41] S. Zschaler, E. Bousse, J. Deantoni and B. Combemale. 'A Generic Framework for Representing and Analysing Model Concurrency'. In: *Software and Systems Modeling* 22 (2023), pp. 1319–1340. DOI: 10.1007/s10270-022-01073-2. URL: https://inria.hal.science/hal-03921704.

**International peer-reviewed conferences**

[42] M. Acher. 'A Demonstration of End-User Code Customization Using Generative AI'. In: 18th International Working Conference on Variability Modelling of Software-Intensive Systems. Bern, Switzerland, 7th Feb. 2024. DOI: 10.1145/3634713.3634732. URL: https://hal.science/hal-04312909.

[43] M. Acher, J. G. Duarte and J.-M. Jézéquel. 'On Programming Variability with Large Language Model-based Assistant'. In: SPLC 2023 - 27th ACM International Systems and Software Product Lines Conference. Tokyo, Japan: ACM, 28th Aug. 2023, pp. 1–7. DOI: 10.1145/nnnnnnn.nnnnnnn. URL: https://inria.hal.science/hal-04153310.

[44] M. Acher, L. Lesoil, G. A. Randrianaina, X. Tërnava and O. Zendra. 'A Call for Removing Variability'. In: VaMoS 2023 - 17th International Working Conference on Variability Modelling of Software-Intensive Systems. Odense, Denmark, 25th Jan. 2023, p. 3. DOI: 10.1145/3571788.3571801. URL: https://hal.science/hal-03882594.

[45] M. Acher and J. Martinez. 'Generative AI for Reengineering Variants into Software Product Lines: An Experience Report'. In: SPLC 2023 - 27th ACM International Systems and Software Product Lines Conference. Vol. B. B. Tokyo, Japan: ACM, 25th Aug. 2023, pp. 1–9. DOI: 10.1145/3579028.3609016. URL: https://inria.hal.science/hal-04160693.

[46] B. Combemale, J.-M. Jézéquel, Q. Perez, D. Vojtisek, N. Jansen, J. Michael, F. Rademacher, B. Rumpe, A. Wortmann and J. Zhang. 'Model-Based DevOps: Foundations and Challenges'. In: 2023 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C). Västerås, France: IEEE, 1st Oct. 2023, pp. 429–433. DOI: 10.1109/MODELS-C59198.2023.00076. URL: https://inria.hal.science/hal-04425802.

[47] D. Foures, M. Acher, O. Barais, B. Combemale, J.-M. Jézéquel and J. Kienzle. 'Experience in Specializing a Generic Realization Language for SPL Engineering at Airbus'. In: MODELS 2023 - 26th International Conference on Model-Driven Engineering Languages and Systems. Västerås, Sweden: IEEE, 2023, pp. 1–12. URL: https://inria.hal.science/hal-04216627.

[48] P. Houdaille, D. E. Khelladi, R. Briend, R. Jongeling and B. Combemale. 'Polyglot AST: Towards Enabling Polyglot Code Analysis'. In: ICECCS 2023 - 27th International Conference on Engineering of Complex Computer Systems. Toulouse, France, 2023, pp. 1–10. URL: https://inria.hal.science/hal-04077663.

[49] G. Jouneaux, D. Frölich, O. Barais, B. Combemale, G. Le Guernic, G. Mussbacher and L. T. van Binsbergen. 'Adaptive Structural Operational Semantics'. In: *SLE 2023: Proceedings of the 16th ACM SIGPLAN International Conference on Software Language Engineering*. SLE 2023 - 16th ACM SIGPLAN International Conference on Software Language Engineering. Cascais, Portugal: ACM, 23rd Oct. 2023, pp. 29–42. DOI: 10.1145/3623476.3623517. URL: https://inria.hal.science/hal-04252577.

[50] Z. K. Kebaili, D. E. Khelladi, M. Acher and O. Barais. 'Towards Leveraging Tests to Identify Impacts of Metamodel and Code Co-evolution'. In: CAiSE 2023 - 35th International Conference on Advanced Information Systems Engineering. Vol. 477. Lecture Notes in Business Information Processing. Zaragoza, Spain: Springer International Publishing; Springer International Publishing, 8th June 2023, pp. 129–137. DOI: 10.1007/978-3-031-34674-3_16. URL: https://inria.hal.science/hal-04126496.

[51] P. Ladisa, H. Plate, M. Martinez and O. Barais. 'Taxonomy of Attacks on Open-Source Software Supply Chains'. In: 2023 IEEE Symposium on Security and Privacy (SP). San Francisco, France: IEEE, 21st May 2023, pp. 1509–1526. DOI: 10.1109/SP46215.2023.10179304. URL: https://inria.hal.science/hal-04423761.

[52] P. Ladisa, S. E. Ponta, N. Ronzoni, M. Martinez and O. Barais. 'On the Feasibility of Cross-Language Detection of Malicious Packages in npm and PyPI'. In: ACSAC '23: Annual Computer Security Applications Conference. Austin TX USA, France: ACM, 4th Dec. 2023, pp. 71–82. DOI: 10.1145/3627106.3627138. URL: https://inria.hal.science/hal-04423806.

[53] P. Ladisa, M. Sahin, S. E. Ponta, M. Rosa, M. Martinez and O. Barais. 'The Hitchhiker's Guide to Malicious Third-Party Dependencies'. In: CCS '23: ACM SIGSAC Conference on Computer and Communications Security. Copenhagen Denmark, France: ACM, 26th Nov. 2023, pp. 65–74. DOI: 10.1145/3605770.3625212. URL: https://inria.hal.science/hal-04423802.

[54] V. Lannurien, L. d'Orazio, O. Barais, E. Bernard, O. Weppe, L. Beaulieu, A. Kacete, S. Paquelet and J. Boukhobza. 'HeROfake: Heterogeneous Resources Orchestration in a Serverless Cloud – An Application to Deepfake Detection'. In: CCGrid 2023 - IEEE/ACM 23rd International Symposium on Cluster, Cloud and Internet Computing. Bangalore, India: IEEE, 1st May 2023, pp. 154–165. DOI: 10.1109/CCGrid57682.2023.00024. URL: https://inria.hal.science/hal-04165179.

[55] Q. Le Dilavrec, D. E. Khelladi, A. Blouin and J.-M. Jézéquel. 'HyperDiff: Computing Source Code Diffs at Scale'. In: ESEC/FSE 2023 - 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering. San Francisco (CA, USA), United States: ACM, 2023, pp. 1–12. DOI: 10.1145/3611643.3616312. URL: https://inria.hal.science/hal-04189855.

[56] D. Leroy, B. Combemale, B. Lelandais and M.-P. Oudot. 'Practical Runtime Instrumentation of Software Languages: The Case of SciHook'. In: SLE 2023 - 16th ACM SIGPLAN International Conference on Software Language Engineering. Cascais, Lisbon, Portugal: ACM, 2023, pp. 1–6. URL: https://inria.hal.science/hal-04249049.

[57] N. Mehanna and W. Rudametkin. 'Caught in the Game: On the History and Evolution of Web Browser Gaming'. In: *Companion Proceedings of the ACM Web Conference 2023 (WWW '23 Companion)*, The Web Conference 2023. Austin (TX), United States, 2023, pp. 1–9. URL: https://hal.science/hal-04084097.

[58] B. Rouxel, C. Brown, E. Ebeid, K. Eder, H. Falk, C. Grelck, J. Holst, S. Jadhav, Y. Marquer, M. M. D. Alejandro, K. Nikov, A. Sahafi, U. P. S. Lundquist, A. Seewald, V. Vassalos, S. Wegener and O. Zendra. 'The TeamPlay Project: Analysing and Optimising Time, Energy, and Security for Cyber-Physical Systems'. In: DATE 2023 - Design, Automation and Test in Europe Conference (MPP - Multi-Partner Projects Track ). Antwerp, Belgium, 31st May 2023, pp. 1–6. URL: https://inria.hal.science/hal-04108237.

[59] A. Tchana, R. Colin, A. L. Berre, V. Berger, B. Combemale and L. Pailler. 'rgpdOS: GDPR Enforcement By The Operating System'. In: 2023 53rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks - Supplemental Volume (DSN-S). Porto, France: IEEE, 27th June 2023, pp. 100–104. DOI: 10.1109/DSN-S58398.2023.00032. URL: https://inria.hal.science/hal-04425768.

[60] P. Temple and G. Perrouin. 'Explicit or Implicit? On Feature Engineering for ML-based Variability-intensive Systems'. In: *VaMoS '23: Proceedings of the 17th International Working Conference on Variability Modelling of Software-Intensive Systems*. VaMoS 2023 - 17th International Working Conference on Variability Modelling of Software-Intensive Systems. Odense, Denmark, 2023, pp. 1–3. DOI: 10.1145/3571788.3571804. URL: https://hal.science/hal-03890876.

**National peer-reviewed Conferences**

[61] T. Giraudet and P.-C. David. 'Fonctions d'utilisabilité dans les studios de conception de langages dédiés graphiques'. In: IHM 2023 - 34e Conférence Francophone sur l'Interaction Homme-Machine. Troyes, France, 3rd Apr. 2023, pp. 1–2. DOI: 10.1145/nnnnnnn.nnnnnnn. URL: https://inria.hal.science/hal-04164888.

**Conferences without proceedings**

[62] X. Tërnava, M. Acher and B. Combemale. 'Specialization of Run-time Configuration Space at Compile-time: An Exploratory Study'. In: SAC 2023 - The 38th ACM/SIGAPP Symposium on Applied Computing. Tallinn, Estonia: ACM, 27th Mar. 2023, pp. 1–10. URL: https://hal.science/hal-03916459.

**Scientific books**

[63] M. Duranton, K. D. Bosschere, B. Coppens, C. Gamrat, M. Gray, T. Hoberg, H. Munk, C. Robinson, T. Vardanega and O. Zendra. *HiPEAC Vision 2023: High Performance Embedded Architecture And Compilation*. Jan. 2023, pp. 1–238. URL: https://inria.hal.science/hal-04023794.

**Scientific book chapters**

[64] B. Coppens and O. Zendra. 'Is privacy possible in a digital world?' In: *The HiPEAC Vision 2023*. Jan. 2023, pp. 145–162. DOI: 10.5281/zenodo.7461921. URL: https://inria.hal.science/hal-04113319.

[65] J.-M. Jézéquel. 'Modeling: From CASE Tools to SLE and Machine Learning'. In: *The French School of Programming*. Springer, 1st Sept. 2023, pp. 1–22. URL: https://hal.science/hal-04080311.

[66] H. Martin, P. Temple, M. Acher, J. A. Pereira and J.-M. Jézéquel. 'Machine Learning for Feature Constraints Discovery'. In: *Handbook of Re-Engineering Software Intensive Systems into Software Product Lines*. Springer International Publishing, 5th July 2023, pp. 175–196. DOI: 10.1007/978-3-031-11686-5_7. URL: https://inria.hal.science/hal-03921905.

[67] A. Olmedo, G. Arévalo, I. Cassol, Q. Perez, C. Urtado and S. Vauttier. 'Pull Requests Integration Process Optimization: An Empirical Study'. In: *Evaluation of Novel Approaches to Software Engineering*. Vol. 1829. Communications in Computer and Information Science. Springer, 8th July 2023, pp. 155–178. DOI: 10.1007/978-3-031-36597-3_8. URL: https://imt-mines-ales.hal.science/hal-04157804.

[68] O. Zendra and B. Coppens. 'From cybercrime to cyberwarfare, nobody can overlook cybersecurity any more'. In: *The HiPEAC Vision 2023*. Jan. 2023, pp. 130–144. DOI: 10.5281/zenodo.7461910. URL: https://inria.hal.science/hal-04113296.

[69] O. Zendra and B. Coppens. 'THE RACE FOR CYBERSECURITY'. In: *The HiPEAC Vision 2023*. Jan. 2023, pp. 127–129. URL: https://inria.hal.science/hal-04113336.

**Edition (books, proceedings, special issue of a journal)**

[70] T. Batista, B. Penzenstadler, B. Combemale and G. Mussbacher, eds. *Preface: ICT4S 2023*. 2023 International Conference on ICT for Sustainability (ICT4S). IEEE, 2023, pp. viii–x. DOI: 10.1109/ICT4S58814.2023.00005. URL: https://inria.hal.science/hal-04425642.

[71] B. Combemale, G. Mussbacher, S. Betz, A. Friday, I. Hadar, J. Sallou, I. Groher, H. Muccini, O. Le Meur, C. Herglotz, E. Eriksson, B. Penzenstadler, A.-K. Peters and C. C. Venters, eds. *Joint Proceedings of ICT4S 2023 Doctoral Symposium, Demonstrations & Posters Track and Workshopsco-located with 9th International Conference on Information and Communications Technology for Sustainability (ICT4S 2023)*. 9th International Conference on Information and Communications Technology for Sustainability (ICT4S 2023). 2023. URL: https://inria.hal.science/hal-04425662.

[72] G. Le Guernic, ed. *C&ESAR'22: Ensuring Trust in a Decentralized World*. Computer & Electronics Security Application Rendezvous 2022. Vol. 3329. CEUR-WS.org, 13th Jan. 2023. URL: https://inria.hal.science/hal-04408264.

[73] G. Le Guernic, ed. *C&ESAR'23: Cybersecurity of Smart Peripheral Devices (Mobiles / IoT / Edge)*. Computer & Electronics Security Application Rendezvous 2023. Vol. 3610. CEUR-WS.org, 8th Jan. 2024. URL: https://inria.hal.science/hal-04408267.

[74] R. Lefeuvre, J. Galasso, B. Combemale, H. Sahraoui and S. Zacchiroli, eds. *Fingerprinting and Building Large Reproducible Datasets*. ACM REP '23: Proceedings of the 2023 ACM Conference on Reproducibility and Replicability. 28th June 2023. DOI: 10.5281/zenodo.7989955. URL: https://hal.science/hal-04132604.

**Doctoral dissertations and habilitation theses**

[75] C. Genevey-Metat. 'Apprentissage automatique pour les attaques par canaux auxiliaires'. Université de Rennes, 28th Sept. 2023. URL: https://theses.hal.science/tel-04241537.

[76] L. Lesoil. 'Deep software variability for resilient performance models of configurable systems'. Université de Rennes, 17th Apr. 2023. URL: https://theses.hal.science/tel-04190983.

**Reports & preprints**

[77] S. Fortz, P. Temple, X. Devroey and G. Perrouin. *Towards Feature-based ML-enabled Behaviour Location*. 27th Nov. 2023. URL: https://inria.hal.science/hal-04309208.

[78] G. A. Randrianaina, D. E. Khelladi, O. Zendra and M. Acher. *PyroBuildS: Enabling Efficient Exploration of Linux Configuration Space with Incremental Build*. 15th June 2023. URL: https://hal.science/hal-04130361.

**Other scientific publications**

[79] V. Lannurien, L. D'orazio, O. Barais, S. Paquelet and J. Boukhobza. 'Distributed Function Cache for Heterogeneous Serverless Cloud'. In: Per3S - Performance and Scalability of Storage Systems. Paris, France, 2023, pp. 1–1. URL: https://hal.science/hal-04303898.

[80] Q. Le Dilavrec, D. E. Khelladi, A. Blouin and J.-M. Jézéquel. *Analyser efficacement de grands historiques de code avec HyperAST : une démonstration*. 17th July 2023. URL: https://inria.hal.science/hal-04163509.

## 12.3 Other

**Educational activities**

[81] M. Acher, P. Temple and O. Barais. 'Reproducible Science and Software Engineering'. Doctoral. France, 5th July 2023. URL: https://inria.hal.science/hal-04152637.

## 12.4 Cited publications


[82] A. Arcuri and L. C. Briand. 'A practical guide for using statistical tests to assess randomized algorithms in software engineering'. In: *ICSE*. 2011, pp. 1–10.

[83] A. Avizienis. 'The N-version approach to fault-tolerant software'. In: *Software Engineering, IEEE Transactions on* 12 (1985), pp. 1491–1501.

[84] F. Bachmann and L. Bass. 'Managing variability in software architectures'. In: *SIGSOFT Softw. Eng. Notes* 26 (3 May 2001), pp. 126–132. DOI: http://doi.acm.org/10.1145/379377.375274. URL: http://doi.acm.org/10.1145/379377.375274.

[85] F. Balarin, Y. Watanabe, H. Hsieh, L. Lavagno, C. Passerone and A. Sangiovanni-Vincentelli. 'Metropolis: An integrated electronic system design environment'. In: *Computer* 36.4 (2003), pp. 45–52.

[86] E. Baniassad and S. Clarke. 'Theme: an approach for aspect-oriented analysis and design'. In: *26th International Conference on Software Engineering (ICSE)*. 2004, pp. 158–167.

[87] E. G. Barrantes, D. H. Ackley, S. Forrest and D. Stefanović. 'Randomized instruction set emulation'. In: *ACM Transactions on Information and System Security (TISSEC)* 8.1 (2005), pp. 3–40.

[88] D. Batory, R. E. Lopez-Herrejon and J.-P. Martin. 'Generating Product-Lines of Product-Families'. In: *ASE '02: Automated software engineering*. IEEE, 2002, pp. 81–92.

[89] S. Becker, H. Koziolek and R. Reussner. 'The Palladio component model for model-driven performance prediction'. In: *Journal of Systems and Software* 82.1 (Jan. 2009), pp. 3–22.

[90] 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.

[91] A. Beugnard, J.-M. Jézéquel and N. Plouzeau. 'Contract Aware Components, 10 years after'. In: *WCSI*. 2010, pp. 1–11.

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

[93] J. Bosch, G. Florijn, D. Greefhorst, J. Kuusela, J. H. Obbink and 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.

[94] L. C. Briand, E. Arisholm, S. Counsell, F. Houdek and P. Thévenod–Fosse. 'Empirical studies of object-oriented artifacts, methods, and processes: state of the art and future directions'. In: *Empirical Software Engineering* 4.4 (1999), pp. 387–404.

[95] J. T. Buck, S. Ha, E. A. Lee and D. G. Messerschmitt. 'Ptolemy: A framework for simulating and prototyping heterogeneous systems'. In: *Int. Journal of Computer Simulation* (1994).

[96] T. Bures, P. Hnetynka and 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.

[97] 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 and J. Whittle. *Software Engineering for Self-Adaptive Systems: A Research Roadmap*. Ed. by 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 and J. Magee. Vol. 5525. Betty H. C. Cheng, Rogério de Lemos, Holger Giese, Paola Inverardi, and Jeff Magee. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009.

[98] J. Coplien, D. Hoffman and D. Weiss. 'Commonality and Variability in Software Engineering'. In: *IEEE Software* 15.6 (1998), pp. 37–45.

[99] I. Crnkovic, S. Sentilles, A. Vulgarakis and M. R. Chaudron. 'A classification framework for software component models'. In: *Software Engineering, IEEE Transactions on* 37.5 (2011), pp. 593–615.

[100] K. Deb, A. Pratap, S. Agarwal and T. Meyarivan. 'A fast and elitist multiobjective genetic algorithm: NSGA-II'. In: *Evolutionary Computation, IEEE Transactions on* 6.2 (2002), pp. 182–197.

[101] R. DeMilli and A. J. Offutt. 'Constraint-based automatic test data generation'. In: *Software Engineering, IEEE Transactions on* 17.9 (1991), pp. 900–910.

[102] S. Fortz, P. Temple, X. Devroey, P. Heymans and G. Perrouin. 'VaryMinions: leveraging RNNs to identify variants in event logs'. In: *International Workshop on Machine Learning Techniques for Software Quality Evolution (MaLTeSQuE)*. Athènes, Greece: ACM, Aug. 2021, pp. 13–18. DOI: 10.1145/3472674.3473980. URL: https://hal.science/hal-03840073.

[103] R. B. France and B. Rumpe. 'Model-driven Development of Complex Software: A Research Roadmap'. In: *Proceedings of the Future of Software Engineering Symposium (FOSE '07)*. Ed. by L. C. Briand and A. L. Wolf. IEEE, 2007, pp. 37–54.

[104] S. Frey, F. Fittkau and 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.

[105] G. Halmans and K. Pohl. 'Communicating the Variability of a Software-Product Family to Customers'. In: *Software and System Modeling* 2.1 (2003), pp. 15–36.

[106] C. Hardebolle and F. Boulanger. 'ModHel'X: A component-oriented approach to multi-formalism modeling'. In: *Models in Software Engineering*. Springer, 2008, pp. 247–258.

[107] H. Hemmati, L. C. Briand, A. Arcuri and S. Ali. 'An enhanced test case selection approach for model-based testing: an industrial case study'. In: *SIGSOFT FSE*. 2010, pp. 267–276.

[108] J. Hutchinson, J. Whittle, M. Rouncefield and S. Kristoffersen. 'Empirical assessment of MDE in industry'. In: *Proceedings of the 33rd International Conference on Software Engineering (ICSE '11)*. Ed. by R. N. Taylor, H. Gall and N. Medvidovic. ACM, 2011, pp. 471–480.

[109] J.-M. Jézéquel. 'Model Driven Design and Aspect Weaving'. In: *Journal of Software and Systems Modeling (SoSyM)* 7.2 (May 2008), pp. 209–218. URL: http://www.irisa.fr/triskell/publis/2008/Jezequel08a.pdf.

[110] K. C. Kang, S. G. Cohen, J. A. Hess, W. E. Novak and A. S. Peterson. *Feature-Oriented Domain Analysis (FODA) Feasibility Study*. Tech. rep. Carnegie-Mellon University Software Engineering Institute, Nov. 1990.

[111] J. Kramer and J. Magee. 'Self-Managed Systems: an Architectural Challenge'. In: *Future of Software Engineering*. IEEE, 2007, pp. 259–268.

[112] K.-K. Lau, P. V. Elizondo and Z. Wang. 'Exogenous connectors for software components'. In: *Component-Based Software Engineering*. Springer, 2005, pp. 90–106.

[113] P. McMinn. 'Search-based software test data generation: a survey'. In: *Software Testing, Verification and Reliability* 14.2 (2004), pp. 105–156.

[114] J. Meekel, T. B. Horton and C. Mellone. 'Architecting for Domain Variability'. In: *ESPRIT ARES Workshop*. 1998, pp. 205–213.

[115] R. Mélisson, P. Merle, D. Romero, R. Rouvoy and 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, p. 171.

[116] A. M. Memon. 'An event-flow model of GUI-based applications for testing'. In: *Software Testing, Verification and Reliability* 17.3 (2007), pp. 137–157.

[117] B. Morin, O. Barais, J.-M. Jézéquel, F. Fleurey and A. Solberg. 'Models at Runtime to Support Dynamic Adaptation'. In: *IEEE Computer* (Oct. 2009), pp. 46–53. URL: http://www.irisa.fr/triskell/publis/2009/Morin09f.pdf.

[118] P.-A. Muller, F. Fleurey and J.-M. Jézéquel. 'Weaving Executability into Object-Oriented Meta-Languages'. In: *Proc. of MODELS/UML'2005*. LNCS. Jamaica: Springer, 2005.

[119] C. Nebut, Y. Le Traon and J.-M. Jézéquel. 'System Testing of Product Families: from Requirements to Test Cases'. In: *Software Product Lines*. Springer Verlag, 2006, pp. 447–478. URL: http://www.irisa.fr/triskell/publis/2006/Nebut06b.pdf.

[120] C. Nebut, S. Pickin, Y. Le Traon and 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. URL: http://www.irisa.fr/triskell/publis/2003/nebut03b.pdf.

[121] L. M. Northrop. 'A Framework for Software Product Line Practice'. In: *Proceedings of the Workshop on Object-Oriented Technology*. London, UK: Springer-Verlag, 1999, pp. 365–366.

[122] L. M. Northrop. 'SEI's Software Product Line Tenets'. In: *IEEE Softw.* 19.4 (2002), pp. 32–40.

[123] I. Ober, S. Graf and I. Ober. 'Validating timed UML models by simulation and verification'. In: *International Journal on Software Tools for Technology Transfer* 8.2 (2006), pp. 128–145.

[124] D. L. Parnas. 'On the Design and Development of Program Families'. In: *IEEE Trans. Softw. Eng.* 2.1 (1976), pp. 1–9.

[125] S. Pickin, C. Jard, T. Jéron, J.-M. Jézéquel and Y. Le Traon. 'Test Synthesis from UML Models of Distributed Software'. In: *IEEE Transactions on Software Engineering* 33.4 (Apr. 2007), pp. 252–268.

[126] K. Pohl, G. Böckle and F. J. van der Linden. *Software Product Line Engineering: Foundations, Principles and Techniques*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2005.

[127] R. Potvin and J. Levenberg. 'Why Google stores billions of lines of code in a single repository'. In: *Communications of the ACM* 59.7 (2016), pp. 78–87.

[128] B. Randell. 'System structure for software fault tolerance'. In: *Software Engineering, IEEE Transactions on* 2 (1975), pp. 220–232.

[129] J. Rothenberg, L. E. Widman, K. A. Loparo and N. R. Nielsen. 'The Nature of Modeling'. In: *in Artificial Intelligence, Simulation and Modeling*. John Wiley & Sons, 1989, pp. 75–92.

[130] P. Runeson and M. Höst. 'Guidelines for conducting and reporting case study research in software engineering'. In: *Empirical Software Engineering* 14.2 (2009), pp. 131–164.

[131] D. Schmidt. 'Guest Editor's Introduction: Model-Driven Engineering'. In: *IEEE Computer* 39.2 (2006), pp. 25–31.

[132] F. Shull, J. Singer and D. I. Sjberg. *Guide to advanced empirical software engineering*. Springer, 2008.

[133] J. Steel and J.-M. Jézéquel. 'On Model Typing'. In: *Journal of Software and Systems Modeling (SoSyM)* 6.4 (Dec. 2007), pp. 401–414. URL: http://www.irisa.fr/triskell/publis/2007/Steel07a.pdf.

[134] C. Szyperski, D. Gruntz and S. Murer. *Component software: beyond object-oriented programming*. Addison-Wesley, 2002.

[135] J.-C. Trigaux and P. Heymans. *Modelling variability requirements in Software Product Lines: a comparative survey*. Tech. rep. FUNDP Namur, 2003.

[136] M. Utting and B. Legeard. *Practical model-based testing: a tools approach*. Morgan Kaufmann, 2010.

[137] P. Vromant, D. Weyns, S. Malek and J. Andersson. 'On interacting control loops in self-adaptive systems'. In: *SEAMS 2011*. ACM, 2011, pp. 202–207.

[138] C. Yilmaz, M. B. Cohen and A. A. Porter. 'Covering arrays for efficient fault characterization in complex configuration spaces'. In: *Software Engineering, IEEE Transactions on* 32.1 (2006), pp. 20–34.

[139] T. Ziadi and J.-M. Jézéquel. 'Product Line Engineering with the UML: Deriving Products'. In: Springer Verlag, 2006, pp. 557–586.