RESEARCH CENTRE

Inria Centre at Rennes University

IN PARTNERSHIP WITH:

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

2024

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, Senior Researcher, from May 2024 until Jul 2024]
- 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]
- 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]
- Gwendal Jouneaux [UNIV RENNES, ATER, until Aug 2024]
- Quentin Perez [INSA RENNES, Associate Professor]
- Noël Plouzeau [UNIV RENNES, Associate Professor]
- Walter Rudametkin Ivey [UNIV RENNES, Associate Professor, HDR]
- Paul Temple [UNIV RENNES, Associate Professor]

Post-Doctoral Fellows

- Faezeh Khorram [CNRS, Post-Doctoral Fellow]
- Gauthier Le Bartz Lyan [INRIA, Post-Doctoral Fellow]
- Samuel Pelissier [INRIA, Post-Doctoral Fellow, from Nov 2024]
- Jolan Philippe [UNIV RENNES, Post-Doctoral Fellow, from Aug 2024]

PhD Students

- Lina Bilal [UNIV RENNES]
- Ewen Brune [INRIA]
- Nicolo Cavalli [INRIA, from Dec 2024]
- Haitam El Hayani [UNIV RENNES, from Sep 2024]
- Theo Giraudet [OBEO, CIFRE]
- Philemon Houdaille [CNRS]
- Zohra Kebaili [CNRS]

- N'Guessan Hermann Kouadio [CGI , CIFRE]
- Clement Lahoche [INRIA]
- Romain Lefeuvre [UNIV RENNES]
- Camille Molinier [UNIV RENNES, from Oct 2024]
- Georges Aaron Randrianaina [UNIV RENNES]
- Chiara Relevat [UNIV RENNES]
- Charly Reux [INRIA, from Oct 2024]
- Sterenn Roux [UNIV RENNES]

Technical Staff

- Emmanuel Chebbi [INRIA, Engineer]
- Caroline Landry [INRIA, Engineer, from Dec 2024]
- Quentin Le Dilavrec [INRIA, Engineer, until Feb 2024]
- Baptiste Mehat [INRIA, Engineer, from Nov 2024]
- Sergiu Mocanu [INRIA, Engineer, from Dec 2024]
- Charly Reux [INRIA, Engineer, until Sep 2024]

Interns and Apprentices

- Elie Castang [UNIV RENNES, Intern, from Jun 2024 until Aug 2024]
- Matis Codjia [UNIV RENNES, Intern, from May 2024 until Jul 2024]
- Mathieu Cyr [INRIA, Intern, from Jun 2024 until Sep 2024]
- Haitam El Hayani [UNIV RENNES, Intern, from Mar 2024 until Aug 2024]
- Sofiane El Naggar [INRIA, Intern, from Jul 2024 until Aug 2024]
- Guillaume Freyermuth [INRIA, Intern, from Jun 2024 until Aug 2024]
- Ulysse-Neo Lartigaud [INSA RENNES, Intern]
- Axel Martin [INSA RENNES, Intern, until Mar 2024]
- Hywel Mathieu [UNIV RENNES, Intern, until Mar 2024]
- Sergiu Mocanu [INRIA, Intern, from Mar 2024 until Sep 2024]
- Duc Nam Nguyen [INRIA, Intern, from Jun 2024 until Aug 2024]
- Florentin Royer [INRIA, Intern, from May 2024 until Jun 2024]
- Aubry Tonnerre [UNIV RENNES, Intern, from Jun 2024 until Aug 2024]
- Ivann Vyslanko [INRIA, Intern, from Jun 2024 until Aug 2024]

Administrative Assistant

- Sophie Maupile [CNRS]

Visiting Scientists

- Jessie Galasso-Carbonnel [Mc Guill]
- Theo Matricon [CNRS, from Mar 2024 until Apr 2024]
- Sergio Queiroz De Medeiros [UFRGS, from Apr 2024]

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 [117]. 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) [121]. 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 [93]. 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 [77]. 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 [123].

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. [98] 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 [107].

3.2.2 Variability modeling

While the basic vision underlying *Software Product Lines* (SPL) can probably be traced back to David Parnas' seminal article [114] 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 [111]. SPL engineering embraces the ideas of mass customization and software reuse. It focuses on the means of efficiently producing and maintaining multiple related software products, exploiting what they have in common and managing what varies among them.

Several definitions of the *software product line* concept can be found in the research literature. Clements *et al.* define it as *a set of software-intensive systems sharing a common, managed set of features*

that satisfy the specific needs of a particular market segment or mission and are developed from a common set of core assets in a prescribed way [112]. Bosch provides a different definition [83]: *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 [89]. 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 [95]. 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 [84]. Variability management is thus increasingly seen as the cornerstone of SPL development, covering the entire development life cycle, from requirements elicitation [125] to product derivation [129] to product testing [110, 109].

Halmans *et al.* [95] 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.* [116]: 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.* [104] (feature, hardware platform, performance and attributes variability) or Bass *et al.* [75] 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* [100]. 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 [84]. 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 [79]. Model Driven Engineering (MDE) makes it possible to automate this weaving process [99]. 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 [108]) manipulating these models to produce for instance a detailed design that can ultimately be transformed to code, or to test suites [115], or other software artifacts.

3.2.3 Component-based software development

Component-based software development [124] aims at providing reliable software architectures with a low cost of design. Components are now used routinely in many domains of software system designs: distributed systems, user interaction, product lines, embedded systems, etc. With respect to more traditional software artifacts (e.g., object oriented architectures), modern component models have the following distinctive features [90]: description of requirements on services required from the other components; indirect connections between components thanks to ports and connectors constructs [102]; hierarchical definition of components (assemblies of components can define new component types); connectors supporting various communication semantics [87]; quantitative properties on the services [82].

In recent years component-based architectures have evolved from static designs to dynamic, adaptive designs (e.g., SOFA [87], Palladio [80], Frascati [105]). 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 [101]. 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 [88, 127]. Platform changes and more generally environmental changes trigger *imposed adaptation*, when the system can no longer use its design to provide the services it must support. In order to support an eternal system [81], 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 [119]". 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 [94]. Multi Objectives Search based techniques [91] 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 [126] mainly relies on model analysis, constraint solving [92] and search-based reasoning [103]. DIVERSE leverages in particular the applications of model-based testing in the context of highly-configurable systems and [128] interactive systems [106] as well as recent advances based on diversity for test cases selection [97].

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 [113], 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 [86] 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 [76] provides modeling elements amenable to semantically equivalent mathematical models. Metropolis offers a precise semantics flexible enough to support different models of computation. ModHel'X [96] 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 [74], recovery blocks [118] and code randomization [78], 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 [122, 120]. Such methods have been used for example to understand the impact of new software development paradigms [85]. 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 [73].

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...).

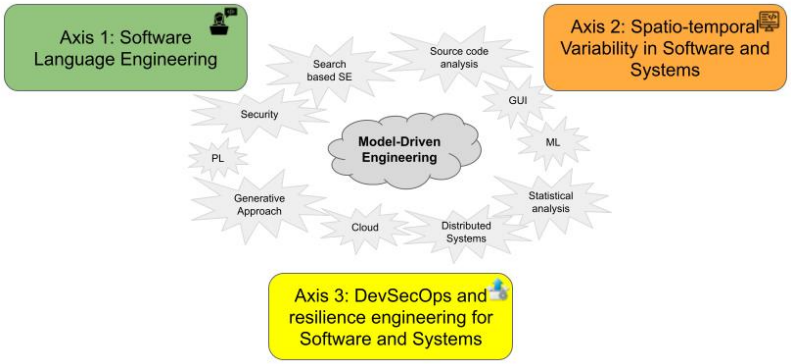

Figure 1: The three research axes of DIVERSE, relying on model driven engineering scientific background and leveraging several related fields

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.

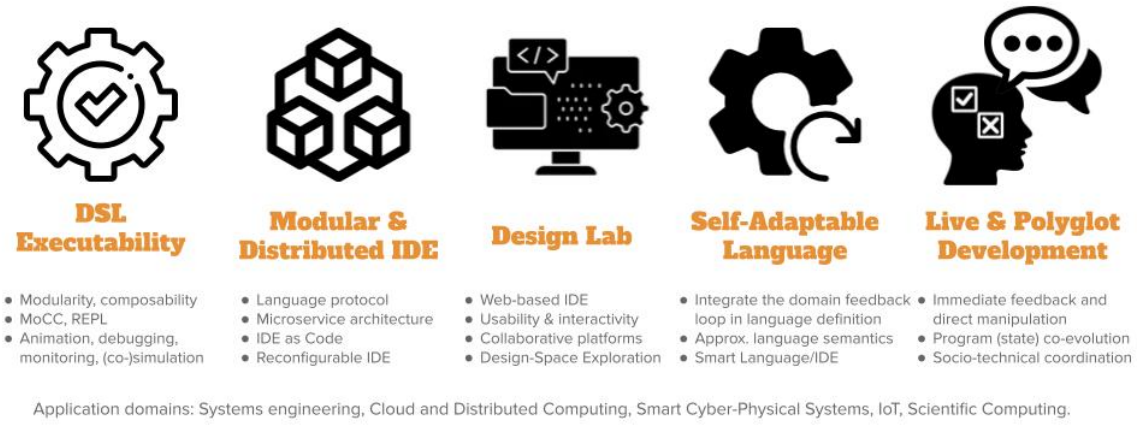

Figure 2: Perspectives on Software Language Engineering (axis #1)

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 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.

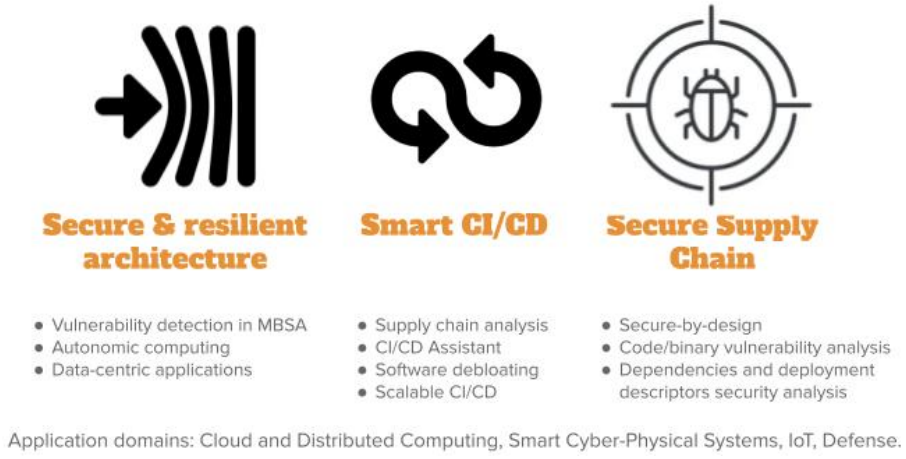

Figure 4: Perspectives on DevSecOps and Resilience Eng. for Software and Systems (axis #3)

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 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, and iii) research results about sustainability and Green IT are also disseminated through a dedicated course named Green Computing, taught at INSA Rennes.

6 Highlights of the year

6.1 Awards

- Mathieu Acher took part in a popularisation video of Monsieur Phi in April 2024. This video reached over 100,000 views [1].
- Gurvan Le Guernic chaired the organising committee for the C&ESAR 2024 conference held in Rennes. Every year since 1997, the French Ministry of Defence has organised a conference on cyber security, known as C&ESAR. This conference is now one of the main events of the European Cyber Week (ECW) organised every autumn in Rennes, France. The aim of C&ESAR is to bring together and facilitate exchanges between various governmental, industrial and academic players with an interest in cybersecurity. This event, which is both educational and scientific, brings together experts, researchers, practitioners and decision-makers. This interdisciplinary approach enables operational practitioners to understand and anticipate future technological (r)evolutions, and enables academics and industry to confront research and the development of products and services with operational realities.
- This year we launched the Code Commons project. This is a project worth more than €1.5 for the team in conjunction with Software Heritage, the aim of which is to prepare Software Heritage for use in data source scenarios for LLM training, with traceability of the data used for this training.

7 New software, platforms, open data

7.1 New software

7.1.1 GEMOC Studio

Name: GEMOC Studio

Keywords: DSL, Language workbench, Model debugging

[1] ChatGPT rêve-t-il de cavaliers électriques ?

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

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).

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

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

7.1.7 Magpie

Keywords: Artificial intelligence, Evolutionary Algorithms, Code optimisation, Automatic software repair

Functional Description: Magpie is a tool for automated software improvement. It uses the genetic improvement methodology to traverse the search space of different software variants to find improved software.

Magpie provides support for improvement of both functional (automated bug fixing) and non-functional (e.g., execution time) properties of software. Two types of language-agnostic source code representations are supported: line-by-line, and XML trees. For the latter we recommend the srcML tool with out-of-the-box support for C/C++/C# and Java. Finally, Magpie also enables parameter tuning and algorithm configuration, both independently and concurrently of the source code search process.

Contact: Aymeric Blot

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 support an approach for fingerprinting and building large reproducible datasets on top of Software Heritage.

8 New results

Publications to process:

8.1 Results for Axis #1: Software Language Engineering

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

8.1.1 Polyglot Programming

Polyglot Software Development: Wait, What? [40] 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 this article, 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. We conclude the article outlining the many challenges and perspectives raised by polyglot software development.

On Polyglot Program Testing [49] In modern applications, it has become increasingly necessary to use multiple languages in a coordinated way to deal with the complexity and diversity of concerns encountered during development. This practice is known as polyglot programming. However, while execution platforms for polyglot programs are increasingly mature, there is a lack of support in how to test polyglot programs. This work is a first step to increase awareness about polyglot testing efforts. It provides an overview of how polyglot programs are constructed, and an analysis of the impact on test writing at its different steps. More specifically, we focus on dynamic white box testing, and how polyglot programming impacts selection of input data, scenario specification and execution, and oracle expression.We discuss the related challenges in particular with regards to the current state of the practice. We envision in this work to raise interest in polyglot program testing within the software engineering community, and help in defining directions for future work.

8.1.2 Language Workbenches

Sirius Web: Insights in Language Workbenches - An Experience Report [35] Sirius Web is an open-source, web-based language workbench maintained by Obeo and hosted under the Eclipse Foundation. It is the successor of Sirius Desktop, an Eclipse-based language workbench used for producing numerous industrial graphical modeling workbenches in the past decades. Leveraging on this valuable experience, in this article we provide an overview of Sirius Web and document the rationales and good practices that have shaped its development. Specifically, we focus on: 1/ the rationales behind modeling and usability features; 2/ their impact on the development lifecycle of tool-supported modeling languages; 3/ the software architecture of the language workbench and the resulting modeling environments. Concrete examples illustrate both the detailed rationales and the use of the tool. We also discuss alternative approaches Obeo considered. In addition to introducing Sirius Web, this work also aims to help language workbench developers make informed design choices for the future development of web-based language workbenches. It also identifies current open questions for the software language engineering community. Moreover, by addressing current open questions in software language engineering, this study contributes to the ongoing dialogue in the community, potentially steering future research directions.

A Type System for Flexible User Interactions Handling [29]

Engineering user interfaces involves the use of multiple user interactions. Developers may struggle with programming and using those user interactions because of a lack of flexibility that affects the current user interface programming approaches. First, developers may want to switch from one user interaction to another close one or combine multiple user interactions without changing much code. Second, developers may also want to use several user interactions to concisely produce the same user command. Third, developers may want to be warned about conflicts between involved user interactions. Currently, developers can hardly perform these first two cases without applying numerous code changes or producing boilerplate code. Regarding the third case, developers can only observe such issues during the execution of the interactive systems, which prolongs the detection time. To overcome these three issues, this work proposes a user interaction type system. This user interaction type system reifies user interactions as first-class concerns with typing facilities for enabling user interactions substitutability and union. It also allows the writing of type checking rules to check for possible issues related to user

interactions at compile time. We implemented the type system within the TypeScript version of Interacto, a framework for processing user interactions. We evaluated the soundness and the expressiveness of our approach through several implemented use cases. This demonstrates the feasibility of the proposed approach and its ability to overcome the three mentioned issues.

There Is Only One Time in Software (Language) Engineering! Software engineering is a complex endeavor that encompasses various socio-technical activities. These activities are traditionally orchestrated over a development life cycle from development time to operation time, and applying engineering processes both at design and run times, and at the application and domain levels. Software Language engineering follows a similar pattern, including the development of domain-specific languages, and all the required tools to support the various language-related activities.This organization, often working in silos, structures the available tools and methods we use, and even the various communities of software engineering (i.e., Conway's law applied to our own discipline!). While this "divide and conquer" approach was crucial in the early days of software engineering, we argue in [45] that it now limits the adaptability required to address what I refer to as software hyper-agility. Modern software systems evolve at an accelerating pace, operate in dynamic environments, and face growing uncertainty.To manage such complexity, a shift towards continuous engineering of cyber-physical and socio-technical ecosystems is necessary, along with more adaptable DSLs. we present the concept of (self-)adaptable languages. We then explore the future of the developer experience with the support of a continuous, feedback-driven, software engineering, with challenges related to abstraction engineering, variability management and digital twins.

8.1.3 Language (co-)evolution

An Empirical Study on Leveraging LLMs for Metamodels and Code Co-evolution [38] Metamodels play an important role in MDE and in specifying a software language. They are cornerstone to generate other artifacts of lower abstraction level, such as code. Developers then enrich the generated code to build their language services and tooling, e.g., editors, and checkers. When a metamodel evolves, part of the code is regenerated and all the additional developers' code can be impacted, thus requiring erroneous code to be co-evolved accordingly. In this work, we explore a novel approach to mitigate the challenge of metamodel evolution impacts on the code using LLMs. In fact, LLMs stand as promising tools for tackling increasingly complex problems and support developers in various tasks of writing, correcting, and documenting source code, models, and other artifacts. However, while there is an extensive empirical assessment of the LLMs capabilities in generating models, code, and tests, there is a lack of work on their ability to support their maintenance. In this work, we focus on the particular problem of metamodels and code co-evolution. We first designed a prompt template structure that contains contextual information about metamodel changes, the abstraction gap between the metamodel and the code, and the erroneous code to co-evolve. To investigate the usefulness of this template, we generated three more variations of the prompts. The generated prompts are then given to the LLM to co-evolve the impacted code. We evaluated our generated prompts and other three of their variations with ChatGPT version 3.5 on seven Eclipse projects from OCL and Modisco evolved metamodels. Results show that ChatGPT can co-evolve correctly 88.7% of the errors due to metamodel evolution, varying from 75% to 100% of correctness rate. When varying the prompts, we observed increased correctness in two variants and decreased correctness in another variant. We also observed that varying the temperature hyperparameter yields better results with lower temperatures. Our results are observed on a total of 5320 generated prompts. Finally, when compared to the quick fixes of the IDE, the generated prompts co-evolutions completely outperform the quick fixes.

8.1.4 Digital Twin

Global Decision Making Support for Complex System Development [44] To succeed with the development of modern and complex systems (e.g., aircrafts or production systems), organizations must have the agility to adapt faster to constantly evolving requirements in order to deliver more reliable and optimized solutions that can be adapted to the needs and environments of their stakeholders including users, customers, suppliers, and partners. However, stakeholders do not have sufficiently explicit and

systematic support for global decision making, considering the vast decision space and complex inter-relationships. This decision space is characterized by increasing yet inadequately represented variability and the uncertainty of the impact of decisions on stakeholders and the solution space. This leads to an ad-hoc decision making process that is slow, error-prone, and often favors local knowledge over global, organization-wide objectives. As a result, one team's design decisions may impose too restrictive requirements on another team. In this work, we evaluate our understanding of global decision making in the context of complex system development based on a conceptual model which explicitly represents and manages decision spaces including variability and impacts. We have conducted our evaluation by means of an exploratory case study where we interviewed domain experts with an average of 20 years of experience in complex system industries and report the key findings and remaining challenges. In the future, we aim at providing explicit and systematic tool-supported approaches for global decision making support for complex systems.

GreyCat: A Framework to Develop Digital Twins at Large Scale [48] Digital Twins (DTs) have become a pivotal technology for enhancing the understanding, monitoring, and ultimately autonomous piloting of systems across various domains, including large-scale critical infrastructures such as smart electricity networks. The development of DTs necessitates developing diverse services that utilize different models and a digital shadow, which encompasses both real-time and historical data from the physical counterpart. The extensive scale of large infrastructures presents significant challenges, including managing numerous parameters, heterogeneous data, and the complex computations required, particularly with the increased use of AI algorithms. Current technologies, built by stacking multiple databases and using general-purpose languages, are inadequate for efficiently implementing digital twin services that need runtime reactivity. This work introduces GreyCat, a framework designed for the development of digital twins over large-scale digital shadows. GreyCat combines imperative object-oriented programming, database persistent indexes, and scalable memory management to facilitate the creation of comprehensive and efficient digital twins. We demonstrate the ease use of GreyCat through simple examples and showcase its effectiveness in constructing the national digital twin of Luxembourg's electricity grid, which is currently operational and managing billions of data points. Reflecting on the development of GreyCat over the past years, we discuss the main lessons learned and identify open questions for future digital twin development frameworks.

Multi-Partner Project: A Model-Driven Engineering Framework for Federated Digital Twins of Industrial Systems (MATISSE) [43] Digital twins are virtual representations of realworld entities or systems. Their primary goal is to help organizations understand and predict the behaviour and properties of these entities or systems. Additionally, digital twins enhance activities such as monitoring, verification, validation, and testing. However, the inherent complexity of digital twins implies challenges throughout the systems engineering process. This notably includes design, development, and analysis phases, as well as deployment, execution, and maintenance. Moreover, existing approaches, methods, techniques, and tools for modelling, simulating, validating, and monitoring single digital twins must now address the increased complexity in federation scenarios. These scenarios introduce new challenges, such as digital twin identification, shared metadata, cross-digital twin communication and synchronization, and federation governance. The KDT Joint Undertaking MATISSE project tackles these challenges by aiming to provide a model-driven framework for the continuous engineering of federated digital twins. It leverages model-driven engineering techniques and practices as the core enabling technology, with traceability serving as an essential infrastructural service for the digital twins federation. In this work, we introduce the MATISSE conceptual framework for digital twins, highlighting both the novelty of the project's concept and its technical objectives. As the project is still in its initial phase, we identify key research challenges relevant to the DATE community and propose a preliminary research roadmap. This roadmap addresses traceability and federation mechanisms, the required continuous engineering strategy, and the development of digital twin-based services for verification, validation, prediction, and monitoring. To illustrate our approach, we present two concrete scenarios that demonstrate practical applications of the MATISSE conceptual framework.

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.

Options Matter: Documenting and Fixing Non-Reproducible Builds in Highly-Configurable Systems [55] A critical aspect of software development, build reproducibility, ensures the dependability, security, and maintainability of software systems. Although several factors, including the build environment, have been investigated in the context of non-reproducible builds, to the best of our knowledge the precise influence of configuration options in configurable systems has not been thoroughly investigated. This work aims at filling this gap. This work thus proposes an approach for the automatic identification of configuration options causing non-reproducibility of builds. It begins by building a set of builds in order to detect non-reproducible ones through binary comparison. We then develop automated techniques that combine statistical learning with symbolic reasoning to analyze over 20,000 configuration options. Our methods are designed to both detect options causing non-reproducibility, and remedy non-reproducible configurations, two tasks that are challenging and costly to perform manually. We evaluate our approach on three case studies, namely Toybox, Busybox, and Linux, analyzing more than 2,000 configurations for each of them. Toybox and Busybox come exempt from nonreproducibility. In contrast, 47% of Linux configurations lead to non-reproducible builds. The approach we propose in this work is capable of identifying 10 configuration options that caused this non-reproducibility. When confronted to the Linux documentation, none of these are documented as non-reproducible. Thus, our identified non-reproducible configuration options are novel knowledge and constitutes a direct, actionable information improvement for the Linux community. Finally, we demonstrate that our methodology effectively identifies a set of undesirable option values, enabling the enhancement and expansion of the Linux kernel documentation while automatically rectifying 96% of encountered non-reproducible builds.

Un ordinateur, ça ne calcule jamais juste [71] Un ordinateur ne calcule jamais parfaitement. Par exemple, si on additionne trois nombres de différentes façons, le résultat peut changer selon le langage de programmation ou l'ordinateur utilisé. Cela peut même amener des robots, voitures ou avions à prendre des chemins différents selon les conditions. Ou à votre application préférée de retourner une réponse, une image ou une vidéo différente ! En informatique, comme dans toutes les sciences, il est important de comprendre que les calculs ont toujours une petite marge d'erreur. Ce qui compte, ce n'est pas la perfection, mais d'en être conscient, de savoir s'adapter et corriger le tir quand c'est nécessaire ! Un ordinateur, ça ne calcule jamais juste… Ou pourquoi les ordinateurs nous trompent parfoisFête de la science, INSA Rennes

Deep Software Variability and Frictionless Reproducibility [69] The ability to recreate computational results with minimal effort and actionable metrics provides a solid foundation for scientific research and software development. When people can replicate an analysis at the touch of a button using open-source software, open data, and methods to assess and compare proposals, it significantly eases verification of results, engagement with a diverse range of contributors, and progress. However, we have yet to fully achieve this; there are still many sociotechnical frictions.Inspired by David Donoho's vision, this talk aims to revisit the three crucial pillars of frictionless reproducibility (data sharing, code sharing, and competitive challenges) with the perspective of deep software variability. Our observation is that multiple layers - hardware, operating systems, third-party libraries, software versions, input data, compile-time options, and parameters - are subject to variability that exacerbates frictions but is also essential for achieving robust, generalizable results and fostering innovation. I will first review the literature, providing evidence of how the complex variability interactions across these layers affect qualitative and quantitative software properties, thereby complicating the reproduction and replication of scientific studies in various fields.I will then present some software engineering and AI techniques that can support the strategic exploration of variability spaces. These include the use of abstractions and

models (e.g., feature models), sampling strategies (e.g., uniform, random), cost-effective measurements (e.g., incremental build of software configurations), and dimensionality reduction methods (e.g., transfer learning, feature selection, software debloating). I will finally argue that deep variability is both the problem and solution of frictionless reproducibility, calling the software science community to develop new methods and tools to manage variability and foster reproducibility in software systems.

Taming uncertainty with MDE: an historical perspective [37] Uncertainty in Informatics can stem from various sources, whether ontological (inherent unpredictability, such as aleatory factors) or epistemic (due to insufficient knowledge). Effectively handling uncertainty, encompassing both ontological and epistemic aspects, to create predictable systems is a key objective for a significant portion of the software engineering community, particularly within the model-driven engineering (MDE) realm. Numerous techniques have been proposed over the years, leading to evolving trends in model-based software development paradigms. This work revisits the history of MDE, aiming to pinpoint the primary aspects of uncertainty that these paradigms aimed to tackle upon their introduction. Our claim is that MDE progressively came after more and more aspects of uncertainty, up to the point that it could now help fully embrace it.

A Demonstration of End-User Code Customization Using Generative AI [41] Producing a variant of code is highly challenging, particularly for individuals unfamiliar with programming. This demonstration introduces 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.

Embracing Deep Variability For Reproducibility and Replicability [42] Reproducibility (aka determinism in some cases) constitutes a fundamental aspect in various fields of computer science, such as floating-point computations in numerical analysis and simulation, concurrency models in parallelism, reproducible builds for third parties integration and packaging, and containerization for execution environments. These concepts, while pervasive across diverse concerns, often exhibit intricate interdependencies, making it challenging to achieve a comprehensive understanding. In this work we delve into the application of software engineering techniques, specifically variability management, to systematically identify and explicit points of variability that may give rise to reproducibility issues (eg language, libraries, compiler, virtual machine, OS, environment variables, etc). The primary objectives are: i) gaining insights into the variability layers and their possible interactions, ii) capturing and documenting configurations for the sake of reproducibility, and iii) exploring diverse configurations to replicate, and hence validate and ensure the robustness of results. By adopting these methodologies, we aim to address the complexities associated with reproducibility and replicability in modern software systems and environments, facilitating a more comprehensive and nuanced perspective on these critical aspects.

Generative AI for Generative Programming: Automating Code Variants and Exploring the Boundaries of LLMs [70] Large language models (LLMs) can be used to automate software engineering tasks, with the hope of boosting developers' productivity without sacrificing reliability. In this talk, I will briefly present two initiatives: the défi Inria LLM4Code and the CodeCommons project around SoftwareHeritage, exemplifying ongoing research efforts. Then, I will show how LLMs can automatically generate software variants across different technological spaces (Python, Rust, JavaScript, etc.) and implement new features from simple prompts. I will also highlight how LLMs can assist in modernizing legacy applications, such as those written in COBOL or in deprecated technologies. I will further discuss the concept of programming without directly manipulating programs, relying almost entirely on AI to manage technical details and customize code. For all their potential, I will also demonstrate that LLMs can dramatically fail

at times, making mistakes that human developers need to carefully verify and correct. I will conclude with an illustration of LLM prompt sensitivity in chess, showcasing how LLMs can be used both to generate variants (e.g., prompts, hypotheses, ideas) and to accelerate scientific discovery through software-based exploration and analysis. 28 november 2024 @ Caen, invited talk at Normastic.

Symfinder: Identifying and visualizing variability implementations in variability-rich Java systems [72] In many variability-intensive systems, variability is implemented in code units provided by a host language, such as classes or functions, which do not align well with the domain features. Annotating or creating an orthogonal decomposition of code in terms of features implies extra effort, as well as massive and cumbersome refactoring activities. Symfinder implements an approach for identifying and visualizing the variability implementation places within the main decomposition structure of object-oriented code assets in a single variability-rich Java system. We use symmetry, as a common property of some main implementation techniques, such as inheritance or overloading, to identify uniformly these places. We use such symmetries to find variation points with variants. Symfinder automatically identifies and visualizes places with symmetry.

FairPipes: Data Mutation Pipelines for Machine Learning Fairness [54] Machine Learning (ML) models are ubiquitous in decisionmaking applications impacting citizens' lives: credit attribution, crime recidivism, etc. In addition to seeking high performance and generalization abilities, ensuring that ML models do not discriminate against citizens regarding their age, gender, or race is essential. To this end, researchers developed various fairness assessment techniques, comprising fairness metrics and mitigation approaches, notably at the model level. However, the sensitivity of ML models to fairness data perturbations has been less explored. This work presents mutation-based pipelines to emulate fairness variations in the data once the model is deployed. FairPipes implements mutation operators that shuffle sensitive attributes, add new values, or affect their distribution. We evaluated FairPipes on seven ML models over three datasets. Our results highlight different fairness sensitivity behaviors across models, from the most sensitive perceptrons to the insensitive support vector machines. We also consider the role of model optimization in fairness performance, being variable across models. FairPipes automates fairness testing at deployment time, informing researchers and practitioners on the fairness sensitivity evolution of their ML models.

VaryMinions: Leveraging RNNs to Identify Variants in Variability-intensive Systems' Logs [64] From business processes to course management, variability-intensive software systems (VIS) are now ubiquitous. One can configure these systems'behaviour by activating options, e.g., to derive variants handling building permits across municipalities or implementing different functionalities (quizzes, forums) for a given course. These customisation facilities allow VIS to support distinct relevant customer requirements while taking advantage of reuse for common parts. Customisation thus allows realising both scope and scale economies. Behavioural differences amongst variants manifest themselves in event logs. To re-engineer this kind of system, one must know which variant(s) have produced which behaviour. Since variant information is barely present in logs, this work supports this task by employing machine learning techniques to classify behaviours (event sequences) among variants. Specifically, we train Long Short Term Memory (LSTMs) and Gated Recurrent Units (GRUs) recurrent neural networks to relate event sequences with the variants they belong to on six different datasets issued from the configurable process and VIS domains.After having evaluated 20 different architectures of LSTM/GRU, our results demonstrate that it is possible to effectively learn the trace-to-variant mapping with high accuracy (at least 80

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

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

HeROcache: Storage-Aware Scheduling in Heterogeneous Serverless Edge - The Case of IDS [51] Intrusion Detection Systems (IDS) are time-sensitive applications that aim to classify potentially malicious network traffic. IDSs are part of a class of applications that rely on short-lived functions that can be run reactively and, as such, could be deployed on edge resources, to offload processing from energy-constrained battery-backed devices. The serverless service model could fit the needs of such applications, given that the platform allows adequate levels of Quality of Service (QoS) for a variety of users, since the criticality of IDS applications depends on several parameters.Deploying serverless functions on unreserved edge resources requires to pay particular attention to (1) initialization delays that could be significant on low resources platforms, (2) inter-function communication between edge nodes, and (3) heterogeneous devices. In this work, we propose both a storage-aware allocation and scheduling policy that seek to minimize task placement costs for service providers on edge devices while optimizing QoS for IDS users.To do so, we propose a caching and consolidation strategy that minimizes cold starts and inter-function communication delays while satisfying QoS by leveraging heterogeneous edge resources.We evaluated our platform in a simulation environment using characterization data from real-world IDS tasks and execution platforms and compared it with a vanilla Knative orchestrator and a storage-agnostic policy. Our strategy achieves 18% fewer QoS penalties while consolidating applications across 80% fewer edge nodes.

Taming the Variability of Browser Fingerprints [50] Browser fingerprinting has become a prevalent technique for trackingand identifying users online, posing significant privacy risks. The increasing variability in web browser configurations, coupled with the continuous evolution of browser features, presents complex challenges in understanding and mitigating the impact of fingerprinting. In this work, we introduce a novel approach that combines feature modeling techniques with tree-based representations to capture the intricate relationships and constraints within browser fingerprints. By translating 22, 773 fingerprints into a feature model with 34, 557 nodes, we enable a comprehensive analysis of their variability and uniqueness across 1, 519 switches and 596 flags on 7 headless and headful browser versions. Our methodology facilitates various use cases, such as generating representative fingerprints for testing, detecting anomalies, and identifying discriminating attributes. We aim to provide developers and researchers with a powerful tool for studying browser fingerprints and developing effective strategies to enhance user privacy in the face of evolving tracking techniques.

Software Frugality in an Accelerating World: the Case of Continuous Integration [67] The acceleration of software development and delivery requires rigorous continuous testing and deployment of software systems, which are being deployed in increasingly diverse, complex, and dynamic environments. In recent years, the popularization of DevOps and integrated software forges like GitLab and GitHub has largely democratized Continuous Integration (CI) practices for a growing number of software. However, this trend intersects significantly with global energy consumption concerns and the growing demand for frugality in the Information and Communication Technology (ICT) sector. CI pipelines typically run in data centers which contribute significantly to the environmental footprint of ICT, yet there is little information available regarding their environmental impact. This article aims to bridge this gap by conducting the first large-scale analysis of the energy footprint of CI pipelines implemented with GitHub Actions and to provide a first overview of the energy impact of CI. We collect, instrument, and reproduce 838 workflows from 396 Java repositories hosted on GitHub to measure their energy consumption. We observe that the average unitary energy cost of a pipeline is relatively low, at 10 Wh. However, due to repeated invocations of these pipelines in real settings, the aggregated energy consumption cost per project is high, averaging 22 kWh. When evaluating CO2 emissions based on regional Wh-to-CO2 estimates, we observe that the average aggregated CO2 emissions are significant, averaging 10.5 kg. To put this into perspective, this is akin to the emissions produced by driving approximately 100 kilometers in a typical European car (110 gCO2/km). In light of our results, we advocate that developers should have the means to better anticipate and reflect on the environmental consequences of their CI choices when implementing DevOps practices.

Large Scale Heap Dump Embedding for Machine Learning: Predicting OpenSSH Key Locations [68] With the evolving landscape of cybersecurity, forensic analysis has become increasingly pivotal, especially

with the integration of machine learning (ML) techniques. However, the use of ML in cybersecurity, and especially in heap dump analysis is still in its infancy,both due to the lack of quality datasets and the difficulty of processing large-scale heap dumps collections. Another complex task lies in the transition from raw byte heap dump data into dense vector representations, or embeddings, that can be used with ML models. This work addresses these challenges by introducing a novel methodology and the Mem2Graph tool for processing large-scale heap dumps collections. This method has been introduced while developing a novel approach to enhance the detection of session keys in OpenSSH heap dumps. Such a novel approach has significantly advanced the state of the art in predicting the location of keys in OpenSSH heap dumps. Importantly, it paves the way for automated ML applications that leverage the structure and embeddings from reconstructed memory graphs, opening new frontiers in both cybersecurity and data science.—The current document is the preprint of a peer-reviewed and published paper under embargo period: "This preprint has not undergone peer review (when applicable) or any post-submission improvements or corrections. The Version of Record of this contribution is published in "ICT Systems Security and Privacy Protection: 39th IFIP International Conference, SEC 2024, Edinburgh, UK, June 12–14, 2024, Proceedings", and is available online.

Semi-Automated and Easily Interpretable Side-Channel Analysis for Modern JavaScript [47] Over the years, developers have become increasingly reliant on web technologies to build their applications, raising concerns about side-channel attacks, especially on cryptographic libraries. Despite the efforts of researchers to ensure constant-time security by proposing tools and methods to find vulnerabilities, challenges remain due to inadequate tools and integration issues in development processes.We tackle the main limitations of state-of-the-art detection tools. While Microwalk is the first and, to the best of our knowledge, only tool to find side-channel vulnerabilities in JavaScript libraries, the instrumentation framework it relies on does not support modern JavaScript features. Moreover, and common to most state-of-the-art detection tools not aimed at JavaScript, writing tests is a tedious process due to the complexity of libraries, the lack of information about test coverage, and the rudimentary interpretability of the report. Furthermore, recent studies show that developers do not use these tools due to compatibility issues, poor usability, and a lack of integration into workflows.We extend Microwalk in several directions. First, we design a generic AST-level tracing technique that is tailored to source-based dynamic side-channel leakage analysis, providing support for the latest language features. Second, we bring semi-automation to Microwalk analysis templates, considerably reducing the manual effort necessary to integrate side-channel analyses into development workflows. Third, we are the first to combine leakage reporting with coverage visualization. We evaluate the new toolchain on a set of cryptographic libraries and show that it can quickly and comprehensively uncover more vulnerabilities while writing tests with half as many lines of code as the previous Microwalk version. By open sourcing our new tracer and analysis template, we hope to increase the adoption of automated side-channel leakage analyses in cryptographic library development.

Free Proxies Unmasked: A Vulnerability and Longitudinal Analysis of Free Proxy Services [53] Free-proxies have been widespread since the early days of the Web, helping users bypass geo-blocked content and conceal their IP addresses. Various proxy providers promise faster Internet or increased privacy while advertising their lists comprised of hundreds of readily available free proxies. However, while paid proxy services advertise the support of encrypted connections and high stability, free proxies often lack such guarantees, making them prone to malicious activities such as eavesdropping or modifying content. Furthermore, there's a market that encourages exploiting devices to install proxies. In this work, we present a 30-month longitudinal study analyzing the stability, security, and potential manipulation of free web proxies that we collected from 11 providers. Our collection resulted in over 640, 600 proxies, that we cumulatively tested daily. We find that only 34.5% of proxies were active at least once during our tests, showcasing the general instability of free proxies. Geographically, a majority of proxies originate from the US and China. Leveraging the Shodan search engine, we identified 4, 452 distinct vulnerabilities on the proxies' IP addresses, including 1, 755 vulnerabilities that allow unauthorized remote code execution and 2, 036 that enable privilege escalation on the host device. Through the software analysis on the proxies' IP addresses, we find that 42, 206 of them appear to run on MikroTik routers. Worryingly, we also discovered 16, 923 proxies that manipulate content, indicating potential malicious intent by proxy

owners. Ultimately, our research reveals that the use of free web proxies poses significant risks to users' privacy and security. The instability, vulnerabilities, and potential for malicious actions uncovered in our analysis lead us to strongly caution users against relying on free proxies.

CESAR'23: Cybersecurity of Smart Peripheral Devices (Mobiles / IoT / Edge) [52] C&ESAR is an educational, professional and scientific conference on cybersecurity whose specific topic changes every year. This year C&ESAR is focused the cybersecurity of "Smart Peripheral Devices", i.e. mobiles, IoT and Edge devices. The scope covers all issues related to the cybersecurity of semi-autonomous connected devices deployed at the periphery of an information system close to its data sources and sinks. Those devices include mobiles, smartphones, IoT devices, and lightweight Edge devices. Those devices often have less computation power than devices in the core of an IT or OT network, and are more exposed to external threats. Hence, attacking or securing them may require different means that for attacking r securing the core of an IT or OT network. C&ESAR 2023 received 18 submissions for peer-review. Out of these, 9 papers were accepted for presentation at the conference. After the conference, 4 were short listed for inclusion in this volume.

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).

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: 2020 ->

Coordinator: Tijs van der Storm (storm@cwi.nl)

Partners:

- CWI Amsterdam (Pays-Bas)

Inria contact: Benoît Combemale

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

collaborative modeling environments. Second, we aim to provide more flexibility to language users by ensuring interoperability between different DSLs, offering live feedback about how the model or program behaves while it is being edited (aka. live programming/modeling), and combining with interactive environments like Jupiter Notebook for literate programming.

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

RIPOST

Title: Resilient and Reproducible Software

Duration: 2024 ->

Coordinator: Arnaud Gotlieb (arnaud@simula.no)

Partners:

- Simula (Norvège)

Inria contact: Mathieu Acher

Summary: Res lient and RePrOducible SofTware

The associate-team RIPOST will strengthen the already existing scientific collaborations between the VIAS Department of Simula Research Laboratory and the French Inria DIVERSE project-team, by addressing systematically the reproducibility testing challenge in computer science. Experimental results in computer science depend on many factors, which can lead to a huge variability in obtained results and can trigger reproducibility issues. Ideally, any published result should be reproducible whatever the configuration of the software systems used to get this result. However, it is often impossible to guarantee as the machines, operating systems, compiler versions, and datasets involved in the experimental evaluation evolve all the time and render 3quickly any result obtained on other versions and systems obsolete. Providing in publications all the ingredients needed – from access to the data and computer systems and configurations, has been for a long time the de facto solution adopted by computer scientists to ensure the scientific reproducibility of their results. We believe that this solution is no longer viable with 1) the deep variability of configurable systems which can tune their parametrization w.r.t. the provided input and 2) the rise of AI-powered methods in empirical Software Engineering which involve error-prone fine-tuning of hundreds or thousands of parameters. Observing that the issue of reproducibility is at the heart of the confidence put by the citizens into Science, we can postulate that reproducibility testing in computer science is still in its infancy, cf. the recent position paper by RIPOST co-PI Mathieu Acher [4]. This assertion is strongly supported by the explosive growth of AI and ML as mentioned above, a development that still accelerates at unprecedented rates [5]. Unless proper steps are taken to ensure the correctness and high quality of how we verify and validate computational results, in particular in the midst of the current data deluge, we risk failures that not only cause economic loss, but ultimately threaten critical infrastructure and thereby the lives and well-being of people. This concern is evident in recent research publications about reproducibility, in the varied landscape of computational sciences [6] as well as in the context of data science and AI/ML [7,8]. The European Commission report “Ethics Guidelines for Trustworthy AI” states that “It is critical that the results of AI systems are reproducible” [9], and reproducibility clearly goes hand in hand with the principles of FAIR data management and stewardship, which is becoming an international golden standard in research.

10.1.3 STIC/MATH/CLIMAT AmSud projects

CAPES/COFECUB: Nº 202/2017

Participants: Mathieu Acher, Heraldo Pimenta Borges Filho.

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.

Participants: Benoît Combemale, Gunter Mussbacher.

Other international visits to the team

- Sergio Queiros De Medeiros joins the teams the 3rd of April. He is professor of Computer Science, Universidade Federal do Rio Grande do Norte, Brazil.
- Jessie Galasso-Carbonnel (Assistant Professor at Mc Guill) joins the team for two weeks.
- Jordi Cabot joins us for the team seminar in June for one week.

10.2.2 Visits to international teams

Research stays abroad - Benoît Combemale visits Mc Guil University for two months. - Romain Lefeuvre visits Mc Guil University for three months.

10.3 European initiatives

10.3.1 Horizon Europe

HiPEAC

Participants: Olivier Zendra, Jean-Marc Jézéquel, Walter Rudametkin.

Title: Title: High Performance, Edge And Cloud computing

Duration: 12/2022 - 05/2025

Coordinator: Koen De Bosschere, UGent

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 à l'Énergie Atomique et aux Énergies Alternatives (CEA), France
- SINTEF AS (SINTEF), Norway
- IDCI Italia Srl, Italy
- Thales, France
- Cloudferro SA, Poland
- Barcelona Supercomputing Center Centro Nacional De Supercomputacion (BSC CNS), Spain

Inria contact: Olivier Zendra (Olivier.Zendra@inria.fr)

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). HiPEAC produces the yearly HiPEAC Vision, a prospective document for EU.

10.3.2 Other european programs/initiatives

MATISSE (ECSEL JU Project)

Participants: Benoît Combemale, Djamel Eddine Khelladi.

- Coordinator: Benoît Combemale
- DiverSE, IRISA Rennes
- Dates: 2024-2027
- Abstract: MATISSE is a European HORIZON-KDT-JU research project bringing together over 30 partners from 7 countries to develop an advanced framework for efficient engineering and validation of industrial systems using Digital Twins (DTs). By integrating DTs with model-based, data-driven, and cloud technologies, MATISSE aims to simulate, test, and predict system behaviours, enhancing both productivity and quality. This innovative approach helps companies optimise their industrial processes, reduce errors, and boost productivity, ultimately simplifying complex operations like machinery production and factory management.

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, Benoît 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 Challita.

- 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: Benoît 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

10.5.1 Allocation d'Installation Scientifique (AIS) - Rennes Métropole

PEEPS

Participants: Stéphanie Challita.

- Coordinator: Stéphanie Challita
- Dates: 2024-2026
- Abstract: The PEEPS project aims to address the issue of precise design for REST APIs, which serve as the foundation of modern web applications. More specifically, this project focuses on the consistency between requirements, code, and documentation of REST APIs. This area remains underexplored in the literature and is one of the challenges developers face on a daily basis.

11 Dissemination

Participants: Olivier Barais, Arnaud Blouin, Johann Bourcier, Benoît Combemale, Jean-Marc Jézéquel, Djamel Eddine Khelladi, Gurvan Leguernic, Gunter Mussbacher, Noël Plouzeau, Didier Vojtisek, Mathieu Acher, Olivier Zendra, Paul Temple, Stéphanie Challita, Aymeric Blot, Noel Plouzeau.

11.1 Promoting scientific activities

11.1.1 Scientific events: organisation

General chair, scientific chair

Jean-Marc Jézéquel has been General Chair of ECSS'24, the Informatics Europe annual conference.

Member of the organizing committees

Benoît Combemale has been co-organizer of a scientific workshop at the Bellairs institute of McGill University dedicated to digital twins and composability.

Aymeric Blot has been co-organizer of GI@ICSE'24, a scientific workshop on genetic improvement which took place within ICSE, the premier software engineering conference.

Olivier Zendra has been co-organizer of the 19th International Workshop on Implementation, Compilation, Optimization of OO Languages, Programs and Systems (ICOOOLPS 2024).

Djamel Khelladi has been co-organizer of the 18th edition of ME@MODELS'24, a scientific workshop on Models Evolution at MODELS, the premier Modeling conference.

11.1.2 Scientific events: selection

Chair of conference program committees

Benoît Combemale has been Program Chair of MODELS'24[63, 65], the premier venue in software engineering dedicated to software and system modeling.

Aymeric Blot has been the Search-Based Software Engineering (SBSE) Track Chair of GECCO'24, the international conference on genetic and evolutionary computation.

Member of the conference program committees

Olivier Barais has been involved in:

- Program committee of MODELS'24.
- Program committee of ICSR 2024.

Arnaud Blouin has been a member of the following PCs:

- 16th ACM SIGCHI symposium on Engineering interactive computing systems (EICS 2024), 2024
- The 39th ACM/SIGAPP Symposium on Applied Computing (SAC), software engineering track, 2024

Stéphanie Challita has been a member of the following PCs:

- SATrends 2024 1st International Workshop on New Trends in Software Architecture, April 14-20, Lisbon, Portugal, co-located event with ICSE 2024 46th International Conference on SofTware Engineering.
- ICSR 2024 21st International Conference on Software and Systems Reuse, June 19-20 2024, Limassol, Cyprus.

Benoît Combemale has been a member of the following PCs:

- ECMFA 2024, The 20th European Conference on Modelling Foundations and Applications.
- the Workshop Track at the 46th International Conference on Software Engineering (ICSE 2024).

Jean-Marc Jézéquel has been involved in:

- Programme board of MODELS'24.
- Programme committee of the 46th International Conference on Software Engineering (ICSE 2024).

Djamel Khelladi has been a member of the following PCs:

- Technical Track at MODELS 2024
- Doctoral Symposium Tack at MODELS 2024

- Technical Track at EDTConf 2024
- Technical Track at MSR 2024
- Early Research Achievement (ERA) Track at SANER 2024

Quentin Perez has been member of the Artifact Evaluation PC of the 27th International Conference on Model Driven Engineering Languages and Systems (MODELS 2024).

Paul Temple has been member of the following PCs:

- Technical Track at SPLC 2024
- Demonstration and tools track at SPLC 2024
- Technical track at VaMoS 2024
- Technical track at ICSR 2024

Olivier Zendra has been member of the following PCs:

- 31st Computer & Electronics Security Application Rendezvous (C&ESAR 2024)
- 19th International Workshop on Implementation, Compilation, Optimization of OO Languages, Programs and Systems (ICOOOLPS 2024)

Aymeric Blot has been member of the following PCs:

- Technical Track at PPSN 2024
- NIER Track at SCAM 2024
- Poster track at ICST 2024
- Worskop ECADA@GECCO 2024
- Workshop SBFT@ICSE 2024

11.1.3 Journal

Member of the editorial boards Benoît Combemale is Editor-in-Chief of the Journal of Software and Systems Modeling, Springer (SoSyM)[33, 31, 34, 30, 32]. He is also member of the editorial boards of the journals COLA, JOT and SQJ.

Jean-Marc Jézéquel is Associate Editor in Chief of the Journal of Software and Systems Modeling, Springer (SoSyM). He is a member of the editorial board of the Journal of Systems and Software (JSS).

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

Reviewer - reviewing activities Team members regularly review for the main journals in the field, namely TSE, Sosym, JSS, EMSE, Jot, SPE, IEEE Software, IST, . . .

11.1.4 Invited talks

Benoît Combemale has been invited to give a keynote at the ACM SIGPLAN conference SLE 2024, the SEMTL 2024 (Software Engineering at Montréal), and the EIT Summer School Green IT and IT for Green. He has been also invited to give an invited talk at Université de Montréal and UQAM. Finally, he has been invited to give a keynote and to attend the following panel in the online TCS/ACM India event dedicated to digital twins.

Aymeric Blot has been invited to give a tutorial at the GI@ICSE'24 workshop.

Quentin Perez has been invited to give a talk at the EIT Summer School Green IT and IT for Green.

11.1.5 Leadership within the scientific community

Jean-Marc Jézéquel is President of Informatics Europe. He is a member of the Luxembourg's NCER Fintech Steering Commitee.

Benoît Combemale is a member of the steering committees for the conference series MODELS, SLE, EDTconf and ICT4S.

Olivier Zendra is founder of ICOOOLPS (International Workshop on Implementation, Compilation, Optimization of OO Languages, Programs and Systems) and a member of its Steering Committee.

Olivier Zendra is a Member of the EU HiPEAC CSA project Steering Committee, and a Member of the HiPEAC Vision Editorial Board.

Olivier Zendra has been invited to co-chair a session on the impact of regulations at Inria's foresight semainar of security and privacy (15/10/2024).

Arnaud Blouin: Founding member and co-organiser of the French GDR-GPL research action on Software Engineering and Human-Computer Interaction (GL-IHM). Co-founder and co-leader of the GDR-IHM group: engineering interactive systems.

11.1.6 Scientific expertise

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

Olivier Zendra is scientific CIR/JEI expert for the Ministry of Research.

Olivier Zendra is cybersecurity officer ("chargé de mission cybersécurité") for Inria Rennes Bretagne Atlantique, representing it in regional and national cybersecurity fora like EUR CyberSchool, Bretagne Cyber Alliance (Brittany's CyberCampus), CreachLabs, etc.

Arnaud Blouin: expert for the CIR agency (research tax credit, "crédit d'impôt recherche"); external reviewer for the ANR (French national research agency).

Olivier Barais: expert for the following call for projects:

- PHC TASSILI 2025
- PHC SAKURA 2025
- SSHN-Court Séjour territoires Palestiniens 2025
- PHC MERLION CHERCHEURS 2025
- COFECUB - CAPES 2025
- PHC PROCOPE 2025
- PHC ORCHID 2025
- PHC PROCORE 2025
- PHC CAI YUANPEI PhD 2025
- PHC GERMAINE DE STAEL 2025
- PHC CAI YUANPEI PhD 2025
- PHC GALILEE 2025
- PHC CEDRE 2025
- PHC PARROT 2025
- PHC UTIQUE 2025
- PHC AL MAQDISI 2025
- ECOS NORD PEROU 2024
- PHC BRANCUSI 2024

- PHC STAR 2025
- Bourses Laplace Tunisie 2024
- VINCI 2024
- PHC MAGHREB 2025
- PHC TOUBKAL 2025
- Chateaubriand STEM 2024
- ECOS NORD MEXIQUE 2023-2024
- PHC BANTOU 2024
- MOPGA Jeunes cherheurs 2024
- SSHN-Court Séjour territoires Palestiniens 2024
- PHC JULES VERNE ISLANDE 2024
- PHC TONLE SAP CAMBODGE 2024

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

11.1.7 Research administration

Olivier Barais is a new member of the CNU 27.

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 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, Paul Temple, 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.

Olivier Barais is the head of the Master in Computer Science at the University of Rennes.

Arnaud Blouin is in charge of the 3rd year at the CS dep at INSA Rennes (around 75 students). He is: an elected member of the research council INSA-IRISA; a member of the CS dep admission council; an elected member of the CS dep council.

Quentin Perez is in charge of industrial relationships for the computer science department at INSA Rennes.

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

11.2.2 Supervision

Theo Giraudet, CIFRE with Obéo (defense in 2025). Benoît Combemale and Arnaud Blouin 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.

Georges Aaron Randrianaina (defense in 2025). Mathieu Acher, Djamel Eddine Khelladi and Olivier Zendra are co-supervisors of this PhD thesis.

Lina Bilal (defense in 2026). Benoît Combemale and Jean-Marc Jézéquel are co-supervisors of this thesis.

Ewen Brune (defense in 2026). Benoît Combemale and Arnaud Blouin are co-supervisors of this thesis.

Philémon Houdaille (defense in 2026). Benoît Combemale and Djamel Eddine Khelladi are co-supervisors of this thesis.

N'Guessan Hermann Kouadio (defense in 2026). Olivier Barais and Mathieu Acher are co-supervisors of this thesis.

Clément Lahoche (defense in 2026). Olivier Barais and Olivier Zendra are co-supervisors of this PhD thesis.

Romain Lefeuvre (defense in 2026). Benoît Combemale and Quentin Perez are co-supe rvisors of this thesis.

Chiara Relevat (defense in 2026). Benoît Combemale and Gurvan Le Guernic are co-supervisors of this thesis.

Sterenn Roux (defense in 2026). Johann Bourcier and Walter Rudametkin are co-supervisors of this thesis.

Camille Molinier (defense in 2027). Olivier Zendra, Olivier Barais and Paul Temple are co-supervisors of this thesis.

Charly Reux ((defense in 2027). Mathieu Acher, Djamel Eddine Khelladi and Olivier Barais are co-supervisors of this thesis.

Nicolò Cavalli (defense in 2027). Arnaud Blouin, Olivier Barais and Djamel Eddine Khelladi are co-supervisors of this thesis.

Haitam El Hayani (defense in 2027). Enrichissement des plateformes et des langages d'approvisionnement et de configuration à l'aide des données collectées en tests et à l'éxécution pour un support avancée au développement de code d'infrastructure. Olivier Barais and Benoît Combemale are co-supervisors of this thesis.

11.2.3 Juries

Olivier Barais was in the PhD jury (reviewer) of Manele Ait Habouche (University of Brest, France), Une infrastructure performante de services distribués pour la collecte et la visualisation de séries temporelles issues de drones marins

Olivier Barais was in the PhD jury (reviewer) of Sébastien Bertrand (University of Bordeaux, France), Modèle de maintenabilité logicielle par analyse statique du graphe de code du programme

Olivier Barais was in the PhD jury (president) of Mathieu Gestin (University of Rennes, France), Privacy Preserving and fully Distributed Identity Management Systems

Olivier Barais was in the PhD jury (president) of Josselin Enet (University of Nantes, France), Protocol-Based Debugging for Domain-Specific Languages

Olivier Barais was in the PhD jury (PhD advisor) of Vincent Lannurien (Ensta Brest, France), Ordonnancement sur ressources hétérogènes pour le Cloud

Olivier Barais was in the PhD jury (PhD advisor) of Piergiorgio Ladisa (Univ Rennes, France), Understanding and Preventing Open-Source Software Supply Chain Attacks

Olivier Barais, Benoît Combemale and Gunter Mussbacher was in the PhD jury (PhD advisor) of Gwendal Jouneaux (Univ Rennes, France), Self-Adaptable Operational Semantics

Olivier Barais has been a member of three recruiting committees for MCF at Univ Montpellier, Univ Brest, and IMT Atlantique.

Jean-Marc Jézéquel has been President of a recruiting committee at INSA Rennes.

Stéphanie Challita has been a member of a PhD students recruiting committee at Inria Rennes.

Djamel Khelladi has been a member of two recruiting committees for MCF at Paris Nanterre University, and at Nice university.

11.3 Popularization

Olivier Zendra, as a member of its editorial board, is a co-author of the **HiPEAC Vision 2024** [57] and of its detailed Rationale accompanying document [58]. He also coordinated the chapter on cybersecurity-related topics [62], composed of 6 articles, being a co-author of 3 of them (on cybersecurity [61], on privacy [59] and on browser privacy [60])

11.3.1 Productions (articles, videos, podcasts, serious games, ...)

Mathieu Acher provided a talk to the "fête de la sciences"[71] Un ordinateur, ça ne calcule jamais juste: Ou pourquoi les ordinateurs nous trompent parfois (Fête de la science, INSA Rennes).

Mathieu Acher took part in the video of the famous youtuber (science populariser) Monsieur Phi: "ChatGPT rêve-t-il de cavaliers électriques ?" video

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] B. Baudry and M. Monperrus. 'The Multiple Facets of Software Diversity: Recent Developments in Year 2000 and Beyond'. In: *ACM Computing Surveys* 48.1 (2015), 16:1–16:26. URL: https://hal.inria.fr/hal-01182103.

[3] G. Bécan, M. Acher, B. Baudry and S. Ben Nasr. 'Breathing Ontological Knowledge Into Feature Model Synthesis: An Empirical Study'. In: *Empirical Software Engineering* 21.4 (2015), pp. 1794–1841. DOI: 10.1007/s10664-014-9357-1. URL: https://hal.inria.fr/hal-01096969.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] 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.

[26] 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.

[27] 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.

[28] O. L. Vera-Pérez, B. Danglot, M. Monperrus and B. Baudry. 'A Comprehensive Study of Pseudo-tested Methods'. In: *Empirical Software Engineering* (2018), pp. 1–33. DOI: 10.1007/s10664-018-9653-2. URL: https://hal.inria.fr/hal-01867423.

12.2 Publications of the year

International journals

[29] A. Blouin. 'A Type System for Flexible User Interactions Handling'. In: *Proceedings of the ACM on Human-Computer Interaction*. EICS (1st Mar. 2024), p. 27. URL: https://inria.hal.science/hal-04485762 (cit. on p. 19).

[30] S. Challita, B. Combemale, H. Ergin, J. Gray, B. Rumpe and M. Schindler. 'Report on the state of the SoSyM journal (2023 summary)'. In: *Software and Systems Modeling* 23 (19th Feb. 2024), pp. 1–5. DOI: 10.1007/s10270-024-01152-6. URL: https://inria.hal.science/hal-04839592 (cit. on p. 36).

[31] B. Combemale, J. Gray, J.-M. Jézéquel and B. Rumpe. 'How does your model represent the system? A note on model fidelity, underspecification, and uncertainty'. In: *Software and Systems Modeling* 23 (23rd Sept. 2024), pp. 1053–1054. DOI: 10.1007/s10270-024-01210-z. URL: https://inria.hal.science/hal-04839512 (cit. on p. 36).

[32] B. Combemale, J. Gray and B. Rumpe. 'Model modularity for reuse, libraries and composition: symbol management is key'. In: *Software and Systems Modeling* 23.3 (27th June 2024), pp. 525–526. DOI: 10.1007/s10270-024-01190-0. URL: https://inria.hal.science/hal-04839542 (cit. on p. 36).

[33] B. Combemale, J. Gray and B. Rumpe. 'Model-based code generation works: But how far does it go?—on the role of the generator'. In: *Software and Systems Modeling* 23 (8th Apr. 2024), pp. 267–268. DOI: 10.1007/s10270-024-01172-2. URL: https://inria.hal.science/hal-04839561 (cit. on p. 36).

[34] B. Combemale, J. Gray and B. Rumpe. 'Modeling for sustainability: Sustainable Development Goals (SDG) of the United Nations'. In: *Software and Systems Modeling* 23 (18th July 2024), pp. 799–800. DOI: 10.1007/s10270-024-01196-8. URL: https://inria.hal.science/hal-04839534 (cit. on p. 36).

[35] T. Giraudet, M. Bats, A. Blouin, B. Combemale and P.-C. David. 'Sirius Web: Insights in Language Workbenches - An Experience Report'. In: *The Journal of Object Technology* 23.1 (2024), pp. 1–20. DOI: 10.5381/jot.2024.23.1.a6. URL: https://inria.hal.science/hal-04797246 (cit. on p. 19).

[36] P. Houdaille, D. E. Khelladi, B. Combemale, G. Mussbacher and T. van Der Storm. 'PolyDebug: a Framework for Polyglot Debugging'. In: *The Art, Science, and Engineering of Programming* 9.3 (2025). DOI: 10.22152/programming-journal.org/2025/10/13. URL: https://hal.science/hal-04906879.

[37] J.-M. Jézéquel. 'Taming uncertainty with MDE: an historical perspective'. In: *Software and Systems Modeling* (28th Oct. 2024), pp. 1–22. DOI: 10.1007/s10270-024-01227-4. URL: https://inria.hal.science/hal-04803258 (cit. on p. 23).

[38] Z. K. Kebaili, D. E. Khelladi, M. Acher and O. Barais. 'An Empirical Study on Leveraging LLMs for Metamodels and Code Co-evolution'. In: *The Journal of Object Technology*. (ECMFA 2024) 23.3 (1st Aug. 2024), pp. 1–14. DOI: 10.5381/jot.2024.23.3.a6. URL: https://hal.science/hal-04667772 (cit. on p. 20).

[39] W. B. Langdon, G. An, A. Blot, V. Nowack, J. Petke, S. Yoo, O. Krauss, E. M. Fredericks and D. Blackwell. 'The 13th International Workshop on Genetic Improvement: (GI @ ICSE 2024)'. In: *Software Engineering Notes* 49.3 (17th July 2024), pp. 42–50. DOI: 10.1145/3672089.3672102. URL: https://hal.science/hal-04884280.

[40] 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 (cit. on p. 19).

International peer-reviewed conferences

[41] M. Acher. 'A Demonstration of End-User Code Customization Using Generative AI'. In: VAMOS 2024 - 18th International Working Conference on Variability Modelling of Software-Intensive Systems. Bern, Switzerland, 2024, pp. 1–6. DOI: 10.1145/3634713.3634732. URL: https://hal.science/hal-04312909 (cit. on p. 23).

[42] M. Acher, B. Combemale, G. A. Randrianaina and J.-M. Jézéquel. 'Embracing Deep Variability For Reproducibility and Replicability'. In: REP 2024 - ACM Conference on Reproducibility and Replicability. Rennes, France, 2024, pp. 1–7. URL: https://hal.science/hal-04582287 (cit. on p. 23).

[43] A. Bucaioni, R. Eramo, L. Berardinelli, H. Bruneliere, B. Combemale, D. E. Khelladi, V. Muttillo, A. Sadovykh and M. Wimmer. 'Multi-Partner Project: A Model-Driven Engineering Framework for Federated Digital Twins of Industrial Systems (MATISSE)'. In: Design, Automation and Test in Europe Conference (DATE 2025). Lyon, France, 2025. URL: https://inria.hal.science/hal-04839759 (cit. on p. 21).

[44] L. Burgueño, D. Foures, B. Combemale, J. Kienzle and G. Mussbacher. 'Global Decision Making Support for Complex System Development'. In: RE 2024 - IEEE 32nd International Requirements Engineering Conference. Reykjavik, Iceland: IEEE, 2024, pp. 252–263. DOI: 10.1109/RE59067.2024.00032. URL: https://inria.hal.science/hal-04839629 (cit. on p. 20).

[45] B. Combemale. 'There Is Only One Time in Software (Language) Engineering! (Keynote)'. In: SLE 2024 - 17th ACM SIGPLAN International Conference on Software Language Engineering. Pasadena CA, United States: ACM, 2024, pp. 1–1. DOI: 10.1145/3687997.3700296. URL: https://inria.hal.science/hal-04827348 (cit. on p. 20).

[46] A. El fraihi, N. Amieur, W. Rudametkin and O. Goga. 'Client-side and Server-side Tracking on Meta: Effectiveness and Accuracy'. In: *Proceedings on Privacy Enhancing Technologies*. PETS 2024 - 24th Privacy Enhancing Technologies Symposium. Vol. 2024. 3. Bristol, United Kingdom, July 2024, pp. 431–445. DOI: 10.56553/popets-2024-0086. URL: https://hal.science/hal-04665102.

[47] I. Fayolle, J. Wichelmann, A. Köhl, W. Rudametkin, T. Eisenbarth and C. Maurice. 'Semi-Automated and Easily Interpretable Side-Channel Analysis for Modern JavaScript'. In: CANS 2024 - 23rd International Conference on Cryptology And Network Security. Cambridge, United Kingdom, 2024, pp. 1–22. URL: https://hal.science/hal-04652991 (cit. on p. 26).

[48] F. Fouquet, T. Hartmann, C. Cecchinel and B. Combemale. 'GreyCat: A Framework to Develop Digital Twins at Large Scale'. In: EDTConf 2024 - 1st International Conference on Engineering Digital Twins. MODELS Companion '24: ACM/IEEE 27th International Conference on Model Driven Engineering Languages and Systems. Linz, Austria: ACM, 2024, pp. 492–495. DOI: 10.1145/3652620.3688265. URL: https://inria.hal.science/hal-04839614 (cit. on p. 21).

[49] P. Houdaille, D. E. Khelladi, B. Combemale and G. Mussbacher. 'On Polyglot Program Testing'. In: FSE 2024 - 32nd ACM International Conference on the Foundations of Software Engineering. Porto de Galinhas, Brazil, 2024, pp. 1–5. DOI: 10.1145/3663529.3663787. URL: https://hal.science/hal-04588744 (cit. on p. 19).

[50] M. Huyghe, C. Quinton and W. Rudametkin. 'Taming the Variability of Browser Fingerprints'. In: SPLC'24 - 28th ACM International Systems and Software Product Lines Conference. Luxembourg, Luxembourg, 2nd Sept. 2024, pp. 1–6. URL: https://hal.science/hal-04622269 (cit. on p. 25).

[51] V. Lannurien, C. Slimani, L. d'Orazio, O. Barais, S. Paquelet and J. Boukhobza. 'HeROcache: Storage-Aware Scheduling in Heterogeneous Serverless Edge - The Case of IDS'. In: CCGrid 2024 - 24th IEEE/ACM international Symposium on Cluster, Cloud and Internet Computing. Philadelphia, United States, 2024, pp. 1–11. URL: https://hal.science/hal-04571484 (cit. on p. 25).

[52] G. Le Guernic. 'C&ESAR'23: Cybersecurity of Smart Peripheral Devices (Mobiles / IoT / Edge)'. In: C&ESAR 2023 - Cybersecurity of Smart Peripheral Devices (Mobiles / IoT / Edge). Vol. 3610. Rennes, France, 8th Jan. 2024. URL: https://inria.hal.science/hal-04408267 (cit. on p. 27).

[53] N. Mehanna, W. Rudametkin, P. Laperdrix and A. Vastel. 'Free Proxies Unmasked: A Vulnerability and Longitudinal Analysis of Free Proxy Services'. In: *Workshop on Measurements, Attacks, and Defenses for the Web (MADWeb'24)*. MADWeb 2024 - Workshop on Measurements, Attacks, and Defenses for the Web. San Diego (CA), United States, 2024, pp. 1–12. DOI: 10.14722/madweb.2024.23035. URL: https://hal.science/hal-04489166 (cit. on p. 26).

[54] C. Molinier, P. Temple and G. Perrouin. 'FairPipes: Data Mutation Pipelines for Machine Learning Fairness'. In: AST 2024 - 5th ACM/IEEE International Conference on Automation of Software Test. Lisbonne, Portugal: ACM, 2024, pp. 1–11. DOI: 10.1145/3644032.3644465. URL: https://hal.science/hal-04440201 (cit. on p. 24).

[55] G. A. Randrianaina, D. E. Khelladi, O. Zendra and M. Acher. 'Options Matter: Documenting and Fixing Non-Reproducible Builds in Highly-Configurable Systems'. In: MSR 2024 - 21th International Conference on Mining Software Repository. Lisbon, Portugal, 2024, pp. 1–11. URL: https://inria.hal.science/hal-04441579 (cit. on p. 22).

[56] B. P. Sanwouo, C. Quinton and P. Temple. 'Toward AI-based Complex Self-Adaptive Systems'. In: *CEUR Workshop Proceedings*. BENEVOL 2024 - 23rd Belgium-Netherlands Software Evolution Workshop. Namur, Belgium, 21st Nov. 2024, pp. 1–2. URL: https://inria.hal.science/hal-04896457.

Scientific books

[57] M. Duranton, P. Carpenter, K. de Bosschere, T. Hoberg, C. Robinson, T. Vardanega and O. Zendra. *HiPEAC Vision 2024*. Jan. 2024, p. 24. URL: https://inria.hal.science/hal-04884248 (cit. on p. 40).

[58] M. Duranton, P. Carpenter, K. de Bosschere, T. Hoberg, C. Robinson, T. Vardanega and O. Zendra. *HiPEAC Vision 2024 - Rationale*. Jan. 2024, p. 226. URL: https://inria.hal.science/hal-04884304 (cit. on p. 40).

Scientific book chapters

[59] B. Coppens and O. Zendra. 'More data for the NCP implies more privacy risks'. In: *HiPEAC Vision 2024 Rationale*. Jan. 2024, pp. 1–6. URL: https://inria.hal.science/hal-04884504 (cit. on p. 40).

[60] W. Rudametkin and O. Zendra. 'The browser: the key to your privacy on the Web'. In: *HiPEAC Vision 2024 Rationale*. Jan. 2024, p. 5. URL: https://inria.hal.science/hal-04884681 (cit. on p. 40).

[61] O. Zendra and B. Coppens. 'The NCP cybersecurity challenges'. In: *HiPEAC Vision 2024 Rationale*. Jan. 2024, p. 7. URL: https://inria.hal.science/hal-04884436 (cit. on p. 40).

[62] O. Zendra and B. Coppens. 'The race for NCP cybersecurity'. In: *HiPEAC Vision 2024 Rationale*. Jan. 2024, p. 3. URL: https://inria.hal.science/hal-04884403 (cit. on p. 40).

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

[63] *Proceedings of the ACM/IEEE 27th International Conference on Model Driven Engineering Languages and Systems, MODELS 2024, Linz, Austria, September 22-27, 2024*. MODELS '24: ACM/IEEE 27th International Conference on Model Driven Engineering Languages and Systems. ACM, 2024. DOI: 10.1145/3640310. URL: https://inria.hal.science/hal-04839651 (cit. on p. 35).

[64] *VaryMinions: Leveraging RNNs to Identify Variants in Variability-intensive Systems' Logs* (2024). URL: https://hal.science/hal-04505454. In press (cit. on p. 24).

[65] *Proceedings of the ACM/IEEE 27th International Conference on Model Driven Engineering Languages and Systems, MODELS Companion 2024, Linz, Austria, September 22-27, 2024*. MODELS Companion '24: ACM/IEEE 27th International Conference on Model Driven Engineering Languages and Systems. ACM, 2024. DOI: 10.1145/3652620. URL: https://inria.hal.science/hal-04839663 (cit. on p. 35).

Doctoral dissertations and habilitation theses

[66] Q. Le Dilavrec. 'Precise temporal analysis of source code histories at scale'. Université de Rennes, 5th Feb. 2024. URL: https://theses.hal.science/tel-04583698.

Reports & preprints

[67] Q. Perez, R. Lefeuvre, T. Degueule, O. Barais and B. Combemale. *Software Frugality in an Accelerating World: the Case of Continuous Integration*. 2024. DOI: 10.48550/arXiv.2410.15816. URL: https://hal.science/hal-04753097 (cit. on p. 25).

[68] F. Rascoussier and L. Lahoche. *Large Scale Heap Dump Embedding for Machine Learning: Predicting OpenSSH Key Locations*. 26th July 2024. DOI: 10.1007/978-3-031-65175-5_28. URL: https://hal.science/hal-04669620 (cit. on p. 25).

Other scientific publications

[69] M. Acher. *Deep Software Variability and Frictionless Reproducibility*. 5th June 2024. URL: https://hal.science/hal-04601752 (cit. on p. 22).

[70] M. Acher. *Generative AI for Generative Programming: Automating Code Variants and Exploring the Boundaries of LLMs*. 28th Nov. 2024. URL: https://hal.science/hal-04809253 (cit. on p. 23).

[71] M. Acher. *Un ordinateur, ça ne calcule jamais juste: Ou pourquoi les ordinateurs nous trompent parfois (Fête de la science, INSA Rennes)*. 10th Oct. 2024. URL: https://hal.science/hal-04736121 (cit. on pp. 22, 40).

Software

[72] [SW] J. Mortara, P. Collet and X. Tërnava, *Symfinder: Identifying and visualizing variability implementations in variability-rich Java systems* 9th Feb. 2024. LIC: GNU Lesser General Public License v3.0 only. HAL: ⟨hal-04449959⟩, URL: https://hal.science/hal-04449959, VCS: https://github.com/DeathStar3/symfinder, SWHID: ⟨swh:1:dir:e6bcb32df0449c85efbc69955449917b9f7b6d90;origin=https://github.com/deathstar3/symfinder;visit=swh:1:snp:1f775e2314a770b871f40df1b69c6019baf1e35a;anchor=swh:1:rev:dd2d24dbaadf5a6f9c21157033584ca99423a759⟩ (cit. on p. 24).

12.3 Cited publications

[73] 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 (cit. on p. 9).

[74] A. Avizienis. 'The N-version approach to fault-tolerant software'. In: *Software Engineering, IEEE Transactions on* 12 (1985), pp. 1491–1501 (cit. on p. 9).

[75] 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 (cit. on p. 7).

[76] 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 (cit. on p. 9).

[77] 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 (cit. on p. 6).

[78] 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 (cit. on p. 9).

[79] 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 (cit. on p. 8).

[80] 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 (cit. on p. 8).

[81] 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 (cit. on p. 8).

[82] A. Beugnard, J.-M. Jézéquel and N. Plouzeau. 'Contract Aware Components, 10 years after'. In: *WCSI*. 2010, pp. 1–11 (cit. on p. 8).

[83] 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 (cit. on p. 7).

[84] 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 (cit. on p. 7).

[85] 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 (cit. on p. 9).

[86] 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) (cit. on p. 9).

[87] 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 (cit. on p. 8).

[88] 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 (cit. on p. 8).

[89] J. Coplien, D. Hoffman and D. Weiss. 'Commonality and Variability in Software Engineering'. In: *IEEE Software* 15.6 (1998), pp. 37–45 (cit. on p. 7).

[90] 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 (cit. on p. 8).

[91] 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 (cit. on p. 9).

[92] R. DeMilli and A. J. Offutt. 'Constraint-based automatic test data generation'. In: *Software Engineering, IEEE Transactions on* 17.9 (1991), pp. 900–910 (cit. on p. 9).

[93] 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 (cit. on p. 6).

[94] 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 (cit. on p. 9).

[95] 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 (cit. on p. 7).

[96] 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 (cit. on p. 9).

[97] 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 (cit. on p. 9).

[98] 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 (cit. on p. 6).

[99] 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 (cit. on p. 8).

[100] 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 (cit. on p. 7).

[101] J. Kramer and J. Magee. 'Self-Managed Systems: an Architectural Challenge'. In: *Future of Software Engineering*. IEEE, 2007, pp. 259–268 (cit. on p. 8).

[102] K.-K. Lau, P. V. Elizondo and Z. Wang. 'Exogenous connectors for software components'. In: *Component-Based Software Engineering*. Springer, 2005, pp. 90–106 (cit. on p. 8).

[103] P. McMinn. 'Search-based software test data generation: a survey'. In: *Software Testing, Verification and Reliability* 14.2 (2004), pp. 105–156 (cit. on p. 9).

[104] J. Meekel, T. B. Horton and C. Mellone. 'Architecting for Domain Variability'. In: *ESPRIT ARES Workshop*. 1998, pp. 205–213 (cit. on p. 7).

[105] 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 (cit. on p. 8).

[106] 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 (cit. on p. 9).

[107] 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 (cit. on p. 6).

[108] 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 (cit. on p. 8).

[109] 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 (cit. on p. 7).

[110] 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 (cit. on p. 7).

[111] 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 (cit. on p. 6).

[112] L. M. Northrop. 'SEI's Software Product Line Tenets'. In: *IEEE Softw.* 19.4 (2002), pp. 32–40 (cit. on p. 7).

[113] 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 (cit. on p. 9).

[114] D. L. Parnas. 'On the Design and Development of Program Families'. In: *IEEE Trans. Softw. Eng.* 2.1 (1976), pp. 1–9 (cit. on p. 6).

[115] 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 (cit. on p. 8).

[116] 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 (cit. on p. 7).

[117] 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 (cit. on p. 5).

[118] B. Randell. 'System structure for software fault tolerance'. In: *Software Engineering, IEEE Transactions on* 2 (1975), pp. 220–232 (cit. on p. 9).

[119] 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 (cit. on p. 8).

[120] 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 (cit. on p. 9).

[121] D. Schmidt. 'Guest Editor's Introduction: Model-Driven Engineering'. In: *IEEE Computer* 39.2 (2006), pp. 25–31 (cit. on p. 6).

[122] F. Shull, J. Singer and D. I. Sjberg. *Guide to advanced empirical software engineering*. Springer, 2008 (cit. on p. 9).

[123] 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 (cit. on p. 6).

[124] C. Szyperski, D. Gruntz and S. Murer. *Component software: beyond object-oriented programming*. Addison-Wesley, 2002 (cit. on p. 8).

[125] J.-C. Trigaux and P. Heymans. *Modelling variability requirements in Software Product Lines: a comparative survey*. Tech. rep. FUNDP Namur, 2003 (cit. on p. 7).

[126] M. Utting and B. Legeard. *Practical model-based testing: a tools approach*. Morgan Kaufmann, 2010 (cit. on p. 9).

[127] 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 (cit. on p. 8).

[128] 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 (cit. on p. 9).

[129] T. Ziadi and J.-M. Jézéquel. 'Product Line Engineering with the UML: Deriving Products'. In: Springer Verlag, 2006, pp. 557–586 (cit. on p. 7).