UMR IRISA

Activity Report 2025

Team DIVERSE

Diversity-centric Software Engineering

Joint team with Centre Inria de l'Université de Rennes

D4 – Language and Software Engineering

Université de Rennes | rennes | INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES RENNES | cnrs | Inria | Inserm | CentraleSupélec | IMT Atlantique Bretagne-Pays de la Loire École Mines-Télécom | Université Bretagne Sud ubs

Project-Team DIVERSE

Creation of the Project-Team: 2014 July 01

Each year, Inria research teams publish an Activity Report presenting their work and results over the reporting period. These reports follow a common structure, with some optional sections depending on the specific team. They typically begin by outlining the overall objectives and research programme, including the main research themes, goals, and methodological approaches. They also describe the application domains targeted by the team, highlighting the scientific or societal contexts in which their work is situated. The reports then present the highlights of the year, covering major scientific achievements, software developments, or teaching contributions. When relevant, they include sections on software, platforms, and open data, detailing the tools developed and how they are shared. A substantial part is dedicated to new results, where scientific contributions are described in detail, often with subsections specifying participants and associated keywords. Finally, the Activity Report addresses funding, contracts, partnerships, and collaborations at various levels, from industrial agreements to international cooperations. It also covers dissemination and teaching activities, such as participation in scientific events, outreach, and supervision. The document concludes with a presentation of scientific production, including major publications and those produced during the year.

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

Contents

Project-Team DIVERSE **1**

1 Team members, visitors, external collaborators **5**

2 Overall objectives **8**

3 Research program **8**
3.1 Context . . . 8
3.2 Scientific background . . . 10
3.2.1 Model-Driven Engineering . . . 10
3.2.2 Variability modeling . . . 10
3.2.3 Component-based software development . . . 12
3.2.4 Validation and verification . . . 13
3.2.5 Empirical software engineering . . . 13
3.3 Research axis . . . 13
3.3.1 Axis #1: Software Language Engineering . . . 14
3.3.2 Axis #2: Spatio-temporal Variability in Software and Systems . . . 16
3.3.3 Axis #3: DevSecOps and Resilience Engineering for Software and Systems . . . 17

4 Application domains **18**

5 Social and environmental responsibility **18**
5.1 Footprint of research activities . . . 18
5.2 Impact of research results . . . 18

6 Highlights of the year **19**
6.1 Awards . . . 19

7 Latest software developments, platforms, open data **19**
7.1 Latest software developments . . . 19
7.1.1 GEMOC Studio . . . 19
7.1.2 Interacto . . . 20
7.1.3 HyperAST . . . 20
7.1.4 CorrectExam . . . 21
7.1.5 PolyglotAST . . . 21
7.1.6 HydroPredictUI . . . 21
7.1.7 Magpie . . . 21
7.2 New platforms . . . 22
7.3 Open data . . . 22

8 New results **23**
8.1 Results for Axis #1: Software Language Engineering . . . 23
8.1.1 Digital Twin . . . 23
8.1.2 Polyglot Programming . . . 24
8.1.3 Language Workbench . . . 25
8.1.4 Scientific Computing . . . 25
8.1.5 Model and code (co-)evolution / Validation and Verification . . . 26
8.2 Results for Axis #2: Spatio-temporal Variability in Software and Systems . . . 27
8.2.1 Reproducibility . . . 27
8.2.2 Variability . . . 29
8.2.3 Performance and Energy consumption . . . 30
8.2.4 Variability and generative AI . . . 32
8.3 Results for Axis #3: DevSecOps and Resilience Engineering for Software and Systems . . . 33
8.3.1 Cybersecurity forecast and policy aspects . . . 33

8.3.2 Vulnerabilities in source code . . . 34
8.3.3 AI security . . . 34
8.3.4 Systems resilience via self-adaptation . . . 34
8.3.5 Browser and privacy . . . 34

9 Bilateral contracts and grants with industry 35
9.1 Bilateral contracts with industry . . . 35

10 Partnerships and cooperations 36
10.1 International initiatives . . . 36
10.1.1 Associate Teams in the framework of an Inria International Lab or in the framework of an Inria International Program . . . 36
10.1.2 Inria associate team not involved in an IIL or an international program . . . 37
10.1.3 COFECUB PUC Rio . . . 37
10.2 International research visitors . . . 38
10.2.1 Visits of international scientists . . . 38
10.2.2 Visits to international teams . . . 38
10.3 European initiatives . . . 38
10.3.1 Horizon Europe . . . 38
10.3.2 Other european programs/initiatives . . . 39
10.4 National initiatives . . . 39
10.4.1 ANR . . . 39
10.4.2 DGA . . . 41
10.4.3 PEPR . . . 41
10.4.4 Campus Cyber . . . 42
10.4.5 Défi LLM4Code . . . 42
10.4.6 Code Commons . . . 42
10.4.7 Défi FRAIME with MERCE and Mitsubishi . . . 43
10.5 Regional initiatives . . . 43

11 Dissemination 43
11.1 Promoting scientific activities . . . 44
11.1.1 Scientific events: organisation . . . 44
11.1.2 Scientific events: selection . . . 44
11.1.3 Journal . . . 45
11.1.4 Invited talks . . . 45
11.1.5 Leadership within the scientific community . . . 46
11.1.6 Scientific expertise . . . 46
11.1.7 Research administration . . . 46
11.2 Teaching - Supervision - Juries - Educational and pedagogical outreach . . . 46
11.2.1 Teaching . . . 46
11.2.2 Supervision . . . 47
11.2.3 Juries . . . 47

12 Scientific production 48
12.1 Major publications . . . 48
12.2 Publications of the year . . . 50
12.3 Cited publications . . . 55

1 Team members, visitors, external collaborators

Research Scientists

- Benoît Combemale [UNIV RENNES, Professor Detachement, from May 2025, HDR]
- Djamel Khelladi [CNRS, Researcher, HDR]
- Gunter Mussbacher [UNIV MCGILL, Senior Researcher, from May 2025 until Jul 2025]
- 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, Professor]
- Arnaud Blouin [INSA RENNES, Associate Professor, HDR]
- Stéphanie Challita [UNIV RENNES, Associate Professor]
- Benoît Combemale [UNIV RENNES, Professor, until Apr 2025, HDR]
- Jean-Marc Jezequel [UNIV RENNES, Professor, HDR]
- Quentin Perez [INSA RENNES, Professor]
- Noël Plouzeau [UNIV RENNES, Associate Professor]
- Walter Rudametkin Ivey [UNIV RENNES, Associate Professor, HDR]
- Paul Temple [UNIV RENNES, Associate Professor]

Post-Doctoral Fellows

- Valentin Bourcier [INRIA, Post-Doctoral Fellow, from Nov 2025]
- Hafiyyan Sayyid Fadhlillah [UNIV RENNES, Post-Doctoral Fellow, from Aug 2025]
- Gauthier Le Bartz Lyan [INRIA, Post-Doctoral Fellow]
- Theo Matricon [INRIA, Post-Doctoral Fellow]
- Samuel Pelissier [INRIA, Post-Doctoral Fellow, until Sep 2025]
- Jolan Philippe [UNIV RENNES, Post-Doctoral Fellow]
- Heraldo Pimenta Borges Filho [UNIV RENNES, Post-Doctoral Fellow, from Sep 2025]

PhD Students

- Axel Allain [INRIA, from Nov 2025]
- Lina Bilal [UNIV RENNES]
- Valere Billaud [CNRS, from Nov 2025]
- Ewen Brune [INRIA]
- Nicolo Cavalli [INRIA]

- Malvin Chevallier [UNIV RENNES, from Oct 2025]
- Jean-Baptiste Doderlein [UNIV RENNES, from Sep 2025]
- Haitam El Hayani [UNIV RENNES]
- Jean-Baptiste Espinasse [SOPRA STERIA, CIFRE, from May 2025]
- Theo Giraudet [OBEO, CIFRE, until Sep 2025]
- Philemon Houdaille [CNRS]
- Imene Issolah [UNIV RENNES, from May 2025]
- Zohra Kebaili [CNRS, until Mar 2025]
- Jacob Kohav [INSA RENNES, from Oct 2025]
- N'Guessan Hermann Kouadio [CGI , CIFRE]
- Lise Lahoche [INRIA]
- Maiwenn Le Goasteller [UNIV RENNES, from Oct 2025]
- Romain Lefeuvre [UNIV RENNES]
- Camille Molinier [UNIV RENNES]
- Yann Paillard [UNIV RENNES]
- Chiara Relevat [UNIV RENNES]
- Charly Reux [INRIA]
- Sterenn Roux [UNIV RENNES]

Technical Staff

- Valere Billaud [CNRS, Engineer, from Mar 2025 until Oct 2025]
- Emmanuel Chebbi [INRIA, Engineer, until Mar 2025]
- Guillaume Claudic [INRIA, Engineer, from May 2025]
- Mathieu Goessens [INRIA, Engineer, from Dec 2025]
- Zohra Kebaili [INRIA, Engineer, from May 2025]
- Jacob Kohav [INRIA, Engineer, from Apr 2025 until Sep 2025]
- Ulysse Kuchler [INRIA, Engineer, from Dec 2025]
- Stephan Kunne [INRIA, Engineer, from May 2025]
- Caroline Landry [INRIA, Engineer]
- Bignon Lokonon [INRIA, Engineer, from Oct 2025]
- Baptiste Mehat [INRIA, Engineer]
- Sergiu Mocanu [INRIA, Engineer, until May 2025]
- Axel Amour N Cho [INRIA, Engineer]
- Corentin Ollivier [INRIA, Engineer, from Jun 2025]

- Alan Prado [INRIA, Engineer, from Dec 2025]
- Georges Aaron Randrianaina [INRIA, Engineer, from Mar 2025 until Aug 2025]
- Antoine Rault [UNIV RENNES, Engineer, from Nov 2025]
- Pierre Treton [UNIV RENNES, Engineer, from Sep 2025]

Interns and Apprentices

- Elson Arampillykudy Eldo [UNIV RENNES, Intern, from May 2025 until Aug 2025]
- Thibault Chanus [INRIA, Intern, from May 2025 until Sep 2025]
- Tom Chauvel [UNIV RENNES, Intern, from Jul 2025 until Sep 2025]
- Romain Debreu [CNRS, Apprentice, from Jun 2025 until Aug 2025]
- Jean-Baptiste Doderlein [ENS RENNES, Intern, until Jul 2025]
- Vincent Gaultier [CESI, Intern, from Feb 2025 until Apr 2025]
- Malo Goetgheluck [UNIV RENNES, Intern, from Jun 2025 until Sep 2025]
- Olivier Henry [ENS PARIS, Intern, from Jun 2025 until Aug 2025]
- Ulysse-Neo Lartigaud [INSA RENNES, Intern]
- Maiwenn Le Goasteller [INSA RENNES, Intern, from Feb 2025 until Aug 2025]
- Nathan Le Guillou [UNIV RENNES, Intern, from Jun 2025 until Jul 2025]
- Gerry Longfils [INRIA, Intern, from Feb 2025 until Apr 2025]
- Margaux Millour [INRIA, Intern, from Jun 2025 until Aug 2025]
- Malo Monin [ENS RENNES, Intern, from Feb 2025 until May 2025]
- Matteo Renzo [INRIA, Intern, from Jun 2025 until Aug 2025]
- Rodrigue Brandon Yando Djamen [INRIA, Intern, from Feb 2025 until May 2025]

Administrative Assistants

- Nathalie Denis [INRIA]
- Sophie Maupile [CNRS]

Visiting Scientists

- Gunter Mussbacher [UNIV MCGILL, from Sep 2025]
- Gunter Mussbacher [UNIV MCGILL, until May 2025]
- Heraldo Pimenta Borges Filho [UNIV PUC RIO, until Sep 2025]

External Collaborators

- Johann Bourcier [UPPA, until Aug 2025, HDR]
- Theo Giraudet [OBEO, from Sep 2025]
- 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 [130]. 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) [134]. 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 [106]. 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 [90]. 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 [136].

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. [111] 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 [120].

3.2.2 Variability modeling

While the basic vision underlying *Software Product Lines* (SPL) can probably be traced back to David Parnas' seminal article [127] 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 [124]. 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* [125]. Bosch provides a different definition [96]: *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 [102]. 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 [108]. 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 [97]. Variability management is thus increasingly seen as the cornerstone of SPL development, covering the entire development life cycle, from requirements elicitation [138] to product derivation [142] to product testing [123, 122].

Halmans *et al.* [108] 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.* [129]: 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.* [117] (feature, hardware platform, performance and attributes variability) or Bass *et al.* [88] 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* [113]. 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 [97]. 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 [92]. Model Driven Engineering (MDE) makes it possible to automate this weaving process [112]. 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 [121]) manipulating these models to produce for instance a detailed design that can ultimately be transformed to code, or to test suites [128], or other software artifacts.

3.2.3 Component-based software development

Component-based software development [137] 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 [103]: description of requirements on services required from the other components; indirect connections between components thanks to ports and connectors constructs [115]; hierarchical definition of components (assemblies of components can define new component types); connectors supporting various communication semantics [100]; quantitative properties on the services [95].

In recent years component-based architectures have evolved from static designs to dynamic, adaptive designs (e.g., SOFA [100], Palladio [93], Frascati [118]). 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 [114]. 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 [101, 140]. 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 [94], 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 [132]". 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 [107]. Multi Objectives Search based techniques [104] 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 [139] mainly relies on model analysis, constraint solving [105] and search-based reasoning [116]. DIVERSE leverages in particular the applications of model-based testing in the context of highly-configurable systems and [141] interactive systems [119] as well as recent advances based on diversity for test cases selection [110].

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 [126], 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 [99] 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 [89] provides modeling elements amenable to semantically equivalent mathematical models. Metropolis offers a precise semantics flexible enough to support different models of computation. ModHel'X [109] 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 [87], recovery blocks [131] and code randomization [91], 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 [135, 133]. Such methods have been used for example to understand the impact of new software development paradigms [98]. 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 [86].

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. . .).

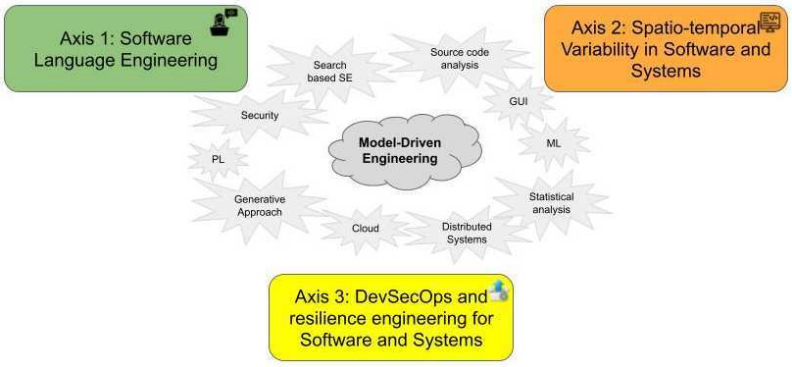


Figure 1: The three research axes of DIVERSE, relying on model driven engineering scientific background and leveraging several related fields

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.

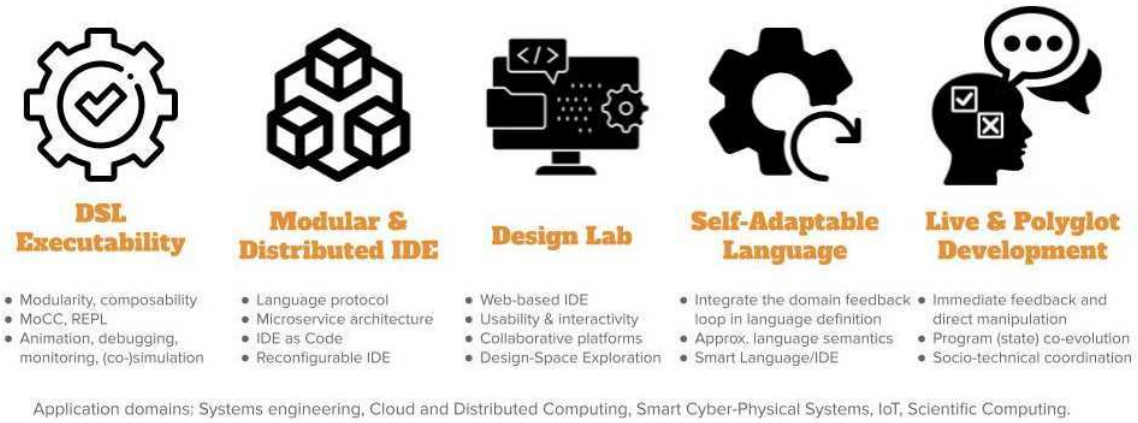


Figure 2: Perspectives on Software Language Engineering (axis #1)

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

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

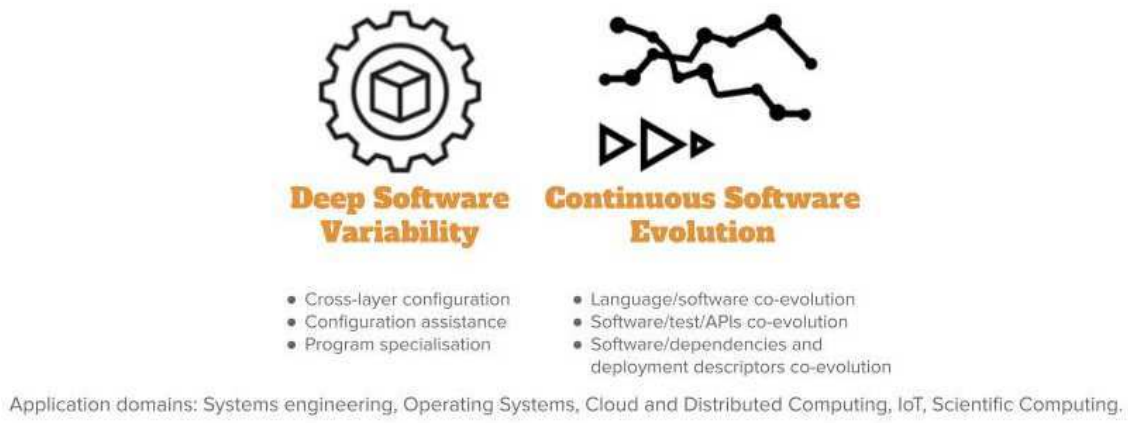


Figure 3: Perspectives on Spatio-temporal Variability in Software and Systems (axis #2)

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.

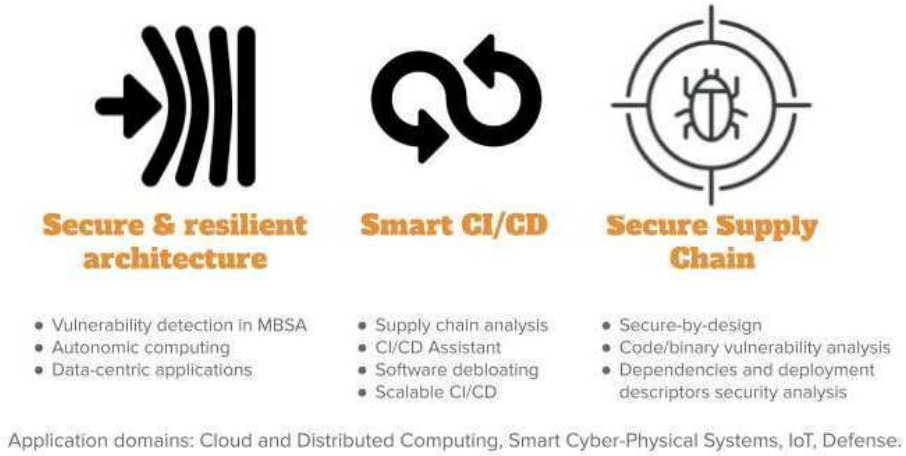


Figure 4: Perspectives on DevSecOps and Resilience Eng. for Software and Systems (axis #3)

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

Three ANR projects have been accepted:

- PEEPS ANR JCJC of Stéphanie Challita
- FutureFlow (France-Switzerland)
- CLEM (ANR France-Austria), PI: Benoit Combemale

One European Erasmus+ Project has been accepted (AIM-PRO).

The new program Engineering Digital Twins – EDT (Agence de programmes 'Numérique'), has been accepted. The diverSE team is rootly involved in its management: Benoit Combemale is co-PI of the program; Jean-Marc Jézéquel is PI of one project of EDT; Arnaud Blouin is Co-PI of another project of EDT.

Djamel Khelladi defended his habilitation.

Jean-Marc Jézéquel has organized ECSS'25 in Rennes, the anual conference of Informatics Europe.

Benoit Combemale has been nominated as IEEE Senior Member

6.1 Awards

- 10-year most influential paper for the 2015 conference paper: *Melange: a meta-language for modular and reusable development of DSLs*, at the 18th ACM SIGPLAN International Conference on Software Language Engineering (SLE'25)
- Best paper award for the conference paper: *Augmenting graphical modeling workbenches with semantic-aware interactive features*, at the 17th ACM SIGCHI Symposium on Engineering Interactive Computing Systems (EICS'25)

7 Latest software developments, platforms, open data

7.1 Latest software developments

7.1.1 GEMOC Studio

Name: GEMOC Studio

Keywords: DSL, Language workbench, Model debugging

Scientific Description: The language workbench put together the following tools seamlessly integrated to the Eclipse Modeling Framework (EMF):

1) Melange, a tool-supported meta-language to modularly define executable modeling languages with execution functions and data, and to extend (EMF-based) existing modeling languages. 2) MoCCML, a tool-supported meta-language dedicated to the specification of a Model of Concurrency and Communication (MoCC) and its mapping to a specific abstract syntax and associated execution functions of a modeling language. 3) GEL, a tool-supported meta-language dedicated to the specification of the protocol between the execution functions and the MoCC to support the feedback of the data as well as the callback of other expected execution functions. 4) BCOoL, a tool-supported meta-language dedicated to the specification of language coordination patterns to automatically coordinates the execution of, possibly heterogeneous, models. 5) Monilog, an extension for monitoring and logging executable domain-specific models 6) Sirius Animator, an extension to the model editor designer Sirius to create graphical animators for executable modeling languages.

Functional Description: The GEMOC Studio is an Eclipse package that contains components supporting the GEMOC methodology for building and composing executable Domain-Specific Modeling Languages (DSMLs). It includes two workbenches: The GEMOC Language Workbench: intended to be used by language designers (aka domain experts), it allows to build and compose new executable DSMLs. The GEMOC Modeling Workbench: intended to be used by domain designers to create, execute and coordinate models conforming to executable DSMLs. The different concerns of a DSML, as defined with the tools of the language workbench, are automatically deployed into the modeling workbench. They parametrize a generic execution framework that provides various generic services such as graphical animation, debugging tools, trace and event managers, timeline.

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

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

Contact: Benoît Combemale

Participants: Didier Vojtisek, Erwan Bousse, Julien Deantoni

Partners: I3S, Université de Nantes

7.1.2 Interacto

Keyword: Interaction

Scientific Description: Interacto is a front-end framework for processing user interface events. With Interacto developers handle user interactions (DnD, drag-lock, double-click, button click, pan, multi-touch, etc.) instead of low-level UI events. Developers configure how to turn a selected user interaction into a (undoable) UI command using a fluent API. Interacto also provides a native support for undo/redo operations.

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

LLM Code Customization with Visual Results: A Benchmark on TikZ [62]

With the rise of AI-based code generation, customizing existing code out of natural language instructions to modify visual results -such as figures or images -has become possible, promising to reduce the need for deep programming expertise. However, even experienced developers can struggle with this task, as it requires identifying relevant code regions (feature location), generating valid code variants, and ensuring the modifications reliably align with user intent. In this paper, we introduce vTikZ, the first benchmark designed to evaluate the ability of Large Language Models (LLMs) to customize code while preserving coherent visual outcomes. Our benchmark consists of carefully curated vTikZ editing scenarios, parameterized ground truths, and a reviewing tool that leverages visual feedback to assess correctness. Empirical evaluation with state-of-the-art LLMs shows that existing solutions struggle to reliably modify code in alignment with visual intent, highlighting a gap in current AI-assisted code editing approaches. We argue that vTikZ opens new research directions for integrating LLMs with visual feedback mechanisms to improve code customization tasks in various domains beyond TikZ, including image processing, art creation, Web design, and 3D modeling.

Linux Kernel Configurations at Scale: A Dataset for Performance and Evolution Analysis [69]

Configuring the Linux kernel to meet specific requirements, such as binary size, is highly challenging due to its immense complexity—with over 15,000 interdependent options evolving rapidly across different versions. Although several studies have explored sampling strategies and machine learning methods to understand and predict the impact of configuration options, the literature still lacks a comprehensive and large-scale dataset encompassing multiple kernel versions along with detailed quantitative measurements. To bridge this gap, we introduce TuxKConfig, an accessible collection of kernel configurations spanning several kernel releases, specifically from versions 4.13 to 5.8. This dataset, gathered through automated tools and build processes, comprises over 240,000 kernel configurations systematically labeled with compilation outcomes and binary sizes. By providing detailed records of configuration evolution and capturing the intricate interplay among kernel options, our dataset enables innovative research in feature subset selection, prediction models based on machine learning, and transfer learning across kernel versions. Throughout this paper, we describe how the dataset has been made easily accessible via OpenML and illustrate how it can be leveraged using only a few lines of Python code to evaluate AI-based techniques, such as supervised machine learning. We anticipate that this dataset will significantly enhance reproducibility and foster new insights into configuration-space analysis at a scale that presents unique opportunities and inherent challenges, thereby advancing our understanding of the Linux kernel's configurability and evolution.

8 New results

Publications to process:

8.1 Results for Axis #1: Software Language Engineering

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

8.1.1 Digital Twin

Towards a Unifying Reference Model for Digital Twins of Cyber-Physical Systems [61] Digital twins are sophisticated software systems for the representation, monitoring, and control of cyber-physical systems, including automotive, avionics, smart manufacturing, and many more. Existing definitions and reference models of digital twins are overly abstract, impeding their comprehensive understanding and implementation guidance. Consequently, a significant gap emerges between abstract concepts and their industrial implementations. We analyze popular reference models for digital twins and combine these into a significantly detailed unifying reference model for digital twins that reduces the concept-implementation gap to facilitate their engineering in industrial practice. This enhances the understanding of the concepts of digital twins and their relationships and guides developers to implement digital twins effectively.

Evolution at the Core of Digital Twin Engineering [50] Engineering Digital Twins (EDT) presents a multifaceted challenge that extends beyond managing the lifecycle of a Digital Twin (DT) to include its continuous, dynamic interaction with the lifecycle of the actual object, system, or process it represents, referred to as the Actual Twin (AT). The relationship between the lifecycles of DT and AT necessitates a rethinking of the software development lifecycle of DTs. This work examines the deeply intertwined lifecycles of DT and AT, arguing that effective methods for EDT must embrace the mutual and adaptive evolution of both over time. We propose placing evolution at the core of EDT. We identify key triggers of DT evolution, examine the engineering dimensions involved, and explore how the best practices, technologies, and tools of DevOps can support this evolution. Finally, we discuss current challenges and opportunities in the field. This work serves as a call to action for the EDT community to adopt evolution as a crucial factor and core principle in EDT.

On the Challenges of Integrating Digital Twins [53] Digital Twins (DTs) are a key technology for smart ecosystems to provide accurate digital representation of their constituents, e.g., smart buildings, farms, transportation, and citizens, as well as synchronization between the digital and the real subject, and the exploration of what-if scenarios and tradeoff reasoning. To cope with emerging complex socio-technical ecosystems, we need to bring DTs together, which is a challenging endeavor. After giving a historical overview of system adaptation, we review the many enabling technologies that can help with DT integration. Using a smart city as an illuminating example to highlight scenarios that require integration of DTs, we discuss a model-based conceptual framework that identifies DT integration strategies and elaborate on nine key integration challenges that still need to be addressed. We call on the DT community to investigate these challenges.

Model-Driven Engineering for Digital Twins: Opportunities and Challenges [45] Digital twins are increasingly used across a wide range of industries. Modeling is a key to digital twin development - both when considering the models which a digital twin maintains of its real-world complement ("models in digital twin") and when considering models of the digital twin as a complex (software) system itself. Thus, systematic development and maintenance of these models is a key factor in effective and efficient digital twin development, maintenance, and use. We argue that model-driven engineering (MDE), a field with almost three decades of research, will be essential for improving the efficiency and reliability of future digital twin development. To do so, we present an overview of the digital twin life cycle, identifying the different types of

models that should be used and re-used at different life cycle stages (including systems engineering models of the actual system, domain-specific simulation models, models of data processing pipelines, etc.). We highlight some approaches in MDE that can help create and manage these models and present a roadmap for research towards MDE of digital twins.

Modeling and Digital Twins: Insights and Strategies for Software Engineers [29] Using software modeling to address complex systems offers several significant benefits, enabling engineers to design, manage, and evolve such systems more effectively. However, digital twins transform how software engineers design, operate, and maintain complex systems, fostering innovation, efficiency, and reliability in diverse industries. Despite their potential, adopting this conceptual framework poses significant challenges for software engineers, including technical and organizational hurdles and the complexity of integrating digital twin engineering into existing workflows. To provide practical insights, this column discusses challenges and highlights approaches discussed at the ACM/IEEE 27th International Conference on Model Driven Engineering Languages and Systems (MODELS 2024) and the 1st International Conference on Engineering Digital Twins (EDTconf 2024). The selected papers showcase advancements in the engineering of complex software systems, emphasizing modeling techniques and digital twins to address challenges such as improving system understanding and stakeholder communication, enhancing design and development processes, optimizing lifecycles, and supporting the integration of complex systems. The goal of this work is to welcome feedbacks and suggestions on the topics covered.

Engineering Digital Twins: A Research Roadmap [54] As society transitions from traditional engineered physical systems to software-driven ecosystems, the demand for adaptive, cost-effective, and rapidly evolving technologies continues to rise. Digital Twins (DTs) have recently emerged as a key architectural and conceptual tool to address these needs by decoupling high-level system control and decision making from physical operations. By integrating deductive engineering models with inductive, data-driven insights, DTs offer powerful capabilities for simulation, prediction, and adaptive system management. However, despite their promise, current DT implementations often remain fragmented, domain-specific, and expensive to build and maintain. This work proposes a research agenda for a structured and principled approach to the engineering of digital twins (EDT), grounded in established software and systems engineering practices. Our goal is to empower the creation of digital twins that are scalable, reusable, and trustworthy. Building on the capabilities outlined by the Digital Twin Consortium, we offer a complementary perspective by identifying key research challenges associated with their integration in the context of digital twin engineering. In addition, we examine the engineering lifecycle of DT systems and present a comprehensive research agenda for the EDT community.

8.1.2 Polyglot Programming

PolyDebug: a Framework for Polyglot Debugging [41] As software grows increasingly complex, the quantity and diversity of concerns to be addressed also rises. To answer this diversity of concerns, developers may end up using multiple programming languages in a single software project, a practice known as polyglot programming. This practice has gained momentum with the rise of execution platforms capable of supporting polyglot systems. However, despite this momentum, there is a notable lack of development tooling support for developers working on polyglot programs, such as in debugging facilities. Not all polyglot execution platforms provide debugging capabilities, and for those that do, implementing support for new languages can be costly. This work addresses this gap by introducing a novel debugger framework that is language-agnostic yet leverages existing language-specific debuggers. The proposed framework is dynamically extensible to accommodate the evolving combination of languages used in polyglot software development. It utilizes the Debug Adapter Protocol (DAP) to integrate and coordinate existing debuggers within a debugging session. We found that using our approach, we were able to implement polyglot debugging support for three different languages with little development effort. We also found that our debugger did not introduce an overhead significant enough to hinder debugging tasks in many scenarios; however performance did deteriorate with the amount of polyglot calls, making the approach not suitable for every polyglot program structure. The effectiveness of this approach is demonstrated through the development of a prototype, PolyDebug, and its application to use cases involving C, JavaScript, and Python. We evaluated PolyDebug on a dataset of traditional benchmark programs, modified to fit our criteria of polyglot programs. We also assessed the

development effort by measuring the source lines of code (SLOC) for the prototype as a whole as well as its components. Debugging is a fundamental part of developing and maintaining software. Lack of debug tools can lead to difficulty in locating software bugs and slow down the development process. We believe this work is relevant to help provide developers proper debugging support regardless of the runtime environment.

8.1.3 Language Workbench

Modeling: The Heart and Soul of Engineering Smart Ecosystems [52] The pervasive digitalization of our world has ushered in a new era marked by increased complexity and diversity in the development, optimization, and maintenance of modern software-intensive systems. These systems, often characterized by intricate socio-technical components and AI integration, pose challenges for conventional systems engineering approaches and require an alliance of different disciplines. In this work, we argue that they demand a paradigm shift towards integrative modeling across systems engineering, software engineering, data science, and simulation engineering. We highlight the key challenges faced in the development of modern complex systems that need to be addressed by this paradigm shift. We argue that achieving this shift requires new research, tools, and education.

Augmenting graphical modeling workbenches with semantic-aware interactive features [39] Domain-Specific Modeling Languages (DSMLs) usually come with a dedicated integrated environment called a modeling workbench. In the context of graphical DSMLs, such environments provide modelers with dedicated interactive features that help them perform navigation and editing tasks. Many of these features are generic and can be used by graphical DSMLs without any specialization (e.g., a physical zoom). Others require specializations in accordance with the involved DSML. For instance, a semantic zoom requires specifying which elements of the model must be graphically modified at the different zoom levels. However, current language workbenches do not propose facilities to help language designers in developing such semantic-aware features for their graphical modeling workbenches. So language designers must develop these features by hand, which is a complex and time-consuming task. This work proposes a novel approach to help language designers in this task. In addition to existing DSML concerns, such as the syntaxes, we propose to capture the interactive features of the targeted modeling workbench in the form of DSML pragmatics. We propose an implementation of our proposal within one industrial language workbench, Sirius Web. We evaluate our proposal through two representative use cases that support discussion of the feasibility of the proposal. We also evaluate its scalability. The evaluation brings forward challenges the community has to consider while developing highly interactive modeling workbenches.

8.1.4 Scientific Computing

HydroModPy: A Python toolbox for deploying catchment-scale shallow groundwater models [80] In response to the growing demand for groundwater flow models, we present HydroModPy, an open-source toolbox designed to automate their deployment at the catchment scale. Built on top of the MODFLOW-enabling FloPy library, HydroModPy combines the robust WhiteboxTools toolbox for geospatial analysis and the well-validated MODFLOW code for groundwater modeling. This Python-based toolbox streamlines the construction, calibration, and analysis of unconfined aquifer models while adhering to FAIR (Findable, Accessible, Interoperable, and Reusable) principles. It enhances model reproducibility through editable Python code, supports multi-site deployment, and provides compatibility with alternative groundwater flow solvers. Furthermore, it integrates pre- and post-processing functionalities to simplify workflows. The toolbox enables catchment delineation and hydrological feature extraction from DEMs, followed by semi-automatic model construction and advanced visualization of hydraulic head and flow results. Users can choose from predefined aquifer structures and hydraulic properties such as exponential decay of hydraulic conductivity and porosity with depth or import complex 3D geological models. HydroModPy outputs can be exported in standard formats (e.g., raster, shapefile, netCDF), including water table elevation, water table depth, groundwater storage, groundwater-dependent hydrographic network and streamflow rates, and subsurface residence times. HydroModPy is tailored for the deployment in diverse geomorphological and hydrological settings, enabling the testing and exploration of aquifer models under varying recharge conditions. Its deployment capabilities are demonstrated in complex shallow basement and crystalline aquifers, where topography and geology primarily govern groundwater flow dynamics from hillslope to catchment scales. As

an open-source toolbox, HydroModPy is designed for the community and actively encourages contributions from its users. It supports research in hydro(geo)logy and land and water management, while also providing valuable opportunities for teaching and education.

Facilitating Heterogeneity Management on the Computing Continuum [73] The computing continuum combines edge and cloud computing resources to create a seamless infrastructure. This integration brings about a variety of resources, leading to heterogeneity. Therefore, meeting application requirements such as fast response times, high-quality results, detailed data, low costs, and energy efficiency becomes challenging. We suggest using self-adaptive systems and software variability management methods to manage this complexity. Specifically, we propose modeling software variability and infrastructure heterogeneity to help configure and deploy systems effectively. We illustrate this approach with a video analytics use-case.

Climate Change: What is Computing's Responsibility? [85] This Manifesto was produced from the Perspectives Workshop 25122 entitled "Climate Change: What is Computing's Responsibility?" held March 16-19, 2025 at Schloss Dagstuhl, Germany. The Workshop provided a forum for world-leading computer scientists and expert consultants on environmental policy and sustainable transition to engage in a critical and urgent conversation about computing's responsibilities in addressing climate change - or more aptly, climate crisis. The resulting Manifesto outlines commitments and directions for future action which, if adopted as a basis for more responsible computing practices, will help ensure that these technologies do not threaten the long-term habitability of the planet. We preface our Manifesto with a recognition that humanity is on a path that is not in agreement with international global warming targets and explore how computing technologies are currently hastening the overshoot of these boundaries. We critically assess the vaunted potential for harnessing computing technologies for the mitigation of global warming, agreeing that, under current circumstances, computing is contributing to negative environmental impacts in other sectors. Computing primarily improves efficiency and reduces costs which leads to more consumption and more negative environmental impact. Relying solely on efficiency gains in computing has thus far proven to be insufficient to curb global greenhouse gas emissions. Therefore, computing's purpose within a strategy for tackling climate change must be reimagined. Our recommendations cover changes that need to be urgently made to the design priorities of computing technologies, but also speak to the more systemic shift in mindset, with sustainability and human rights providing a necessary moral foundation for developing the kinds of computing technologies most needed by society. We also stress the importance of digital policy that accounts for both the direct material impacts of computing and the detrimental indirect impacts arising from computing-enabled efficiencies, and the role of computing professionals in informing policy making.

8.1.5 Model and code (co-)evolution / Validation and Verification

Automated co-evolution of metamodels and code [42] In Software Engineering, Model-Driven Engineering (MDE) is a methodology that considers Metamodels as a cornerstone. As an abstract artifact, a metamodel plays a significant role in the specification of a software language, particularly, in generating 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. Problem. When a metamodel evolves, the generated code is automatically updated. As a consequence, the developers' additional code is impacted and needs to be co-evolved accordingly. Contribution. This work proposes a new fully automatic code co-evolution approach with the evolution of the Ecore metamodel. The approach relies on pattern matching of the additional code errors. This process aims to analyze the abstraction gap between the evolved metamodel elements and the code errors to co-evolve them. Evaluation and Results. We evaluated our approach on nine Eclipse projects from OCL, Modisco, and Papyrus over several evolved versions of three metamodels. Results show that we automatically co-evolved 771 errors due to metamodel evolution with 631 matched and applied resolutions. Our approach reached an average of 82% of precision and 81% of recall, varying from 48% to 100% for precision and recall respectively. To check the effect of the co-evolution and its behavioral correctness, we rely on generated test cases before and after co-evolution. We observed that the percentage of passing, failing, and erroneous tests remained the same with insignificant variations in some projects. Thus, suggesting the behavioral correctness of the co-evolution Moreover, we conducted a comparison with the use of quick fixes that represent a usual tool for correcting code errors in an IDE. We found that our automatic co-evolution approach outperforms the use of quick fixes that lacked the context of metamodel

evolution. Finally, we also compared our approach with the state-of-the-art semi-automatic co-evolution approach. As expected, precision and recall are slightly better with semi-automation, but with the burden of manual intervention, which is alleviated with our automatic co-evolution.

A language-parametric test amplification framework for executable domain-specific languages [43] Behavioral models are important assets that must be thoroughly verified early in the design process. This can be achieved with manually-written test cases that embed carefully hand-picked domain-specific input data. However, such test cases may not always reach the desired level of quality, such as high coverage or being able to localize faults efficiently. Test amplification is an interesting emergent approach to improve a test suite by automatically generating new test cases out of existing manually-written ones. Yet, while ad-hoc test amplification solutions have been proposed for a few programming languages, no solution currently exists for amplifying the test suites of behavioral models. In order to fill this gap, we propose an automated and generic test amplification approach for executable Domain Specific Languages (DSLs). Hence, given an executable DSL, a conforming behavioral model, and an existing test suite, our approach synthesizes new regression test cases in three steps: (i) generating new test inputs by applying a set of generic modifiers on the existing test inputs; (ii) running the model under test with new inputs and generating assertions from the execution traces; and (iii) selecting the new test cases that increase the initial test quality. We provide a textual DSL to control and configure the amplification process, along with tool support for the whole approach atop the Eclipse GEMOC Studio. For assessment, we report on empirical evaluations over two different executable DSLs, which show improved test quality in terms of both coverage and mutation score.

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

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

8.2.1 Reproducibility

Teaching Reproducibility and Embracing Variability: From Floating-Point Experiments to Replicating Research [49] Reproducibility is often discussed but rarely practiced in undergraduate computer science education. In this work, we present the design, implementation, and evaluation of a 24-hour hands-on course entirely dedicated to reproducibility and variability in computational experiments. Taught to fourth-and fifth-year students at INSA Rennes in Fall 2024, the course combines scientific thinking, software engineering practices, and variability analysis. Students first explored the non-associativity of floating-point arithmetic as a reproducibility "Hello World" using Docker, GitHub Actions, and templated experimentation to analyze sources of variability across programming languages, compiler flags, and numerical precision. The second half of the course focused on reproducing and replicating actual research papers, including studies on large language models playing chess, home advantage in football during COVID-19, and energy efficiency across programming languages. Students successfully reproduced key results, identified subtle reproducibility issues such as changes in library defaults, and designed replications that extended or challenged original findings. We describe the course structure, pedagogical strategies, and lessons learned, including when students found reproducibility flaws in the instructor's own prior work. Our experience suggests that reproducibility and variability deserve a central place in computer science education and can be taught in a way that is both technically rigorous and scientifically engaging.

Software Variability for Replicable Science [47] The ability to recreate computational results provides a solid foundation for scientific research. Yet, reproducibility and getting identical results with the original data, code, and environment are challenging, due to many uncontrolled or undocumented variability factors. When reproduction is achieved, replicability can be envisioned to check whether conclusions still hold or generalize when well-controlled changes created by inevitable software variability-are introduced. In this work, we first provide evidence that deep software variability - spanning operating systems, compilers, input data, versions, and configurations, etc. - is impacting reproducibility and replicability in numerous fields of computational

science. We then present an approach based on "modelling, sampling, measuring, learning" to systematically explore variability spaces of neuroimaging pipelines. We also illustrate how Masters' students et INSA Rennes leverage variability to reproduce and replicate studies in soccer, chess, and energy consumption. Throughout this work, we try to convince the audience that software-engineering and variability researchers have a key role to play for truly replicable science. Keynote (invited talk) at CBSoft Brazilian Symposium on Software Components, Architectures, and Reuse

Re-evaluating Metamorphic Testing of Chess Engines: A Replication Study [44] This study aims to confirm, replicate and extend the findings of a previous work entitled "Metamorphic Testing of Chess Engines" that reported inconsistencies in the analyses provided by Stockfish, the most widely used chess engine, for transformed chess positions that are fundamentally identical. Initial findings, under conditions strictly identical to those of the original study, corroborate the reported inconsistencies. However, the original work considers a specific dataset (including randomly generated chess positions, end-games, or checkmate problems) and very low analysis depth (10 plies,1 corresponding to 5 moves). These decisions pose threats that limit generalizability of the results, but also their practical usefulness both for chess players and maintainers of Stockfish. Thus, we replicate the original study. We consider this time (1) positions derived from actual chess games, (2) analyses at appropriate and larger depths, and (3) different versions of Stockfish. We conduct novel experiments on thousands of positions, employing significantly deeper searches. The replication results show that the Stockfish chess engines demonstrate significantly greater consistency in its evaluations. The metamorphic relations are not as effective as in the original work, especially on realistic chess positions. We also demonstrate that, for any given position, there exists a depth threshold beyond which further increases in depth do not result in any evaluation differences for the studied metamorphic relations. We perform an in-depth analysis to identify and clarify the implementation reasons behind Stockfish's inconsistencies when dealing with transformed positions. A first concrete result is thus that metamorphic testing of chess engines is not yet an effective technique for finding faults of Stockfish. Another result is the lessons learned through this replication effort: metamorphic relations must be verified in the context of the domain's specificities; without such contextual validation, they may lead to misleading or irrelevant conclusions; changes in parameters and input dataset can drastically alter the effectiveness of a testing method.

In the Search for Truth: Navigating Variability in Neuroimaging Software Pipelines [71] Neuroimaging pipelines -software-driven analysis workflows of brain images -are characterized by a wide range of tools, parameters, and configuration choices. Such flexibility, while enabling diverse scientific inquiries, gives rise to analytical variability: different pipeline variants can lead to different outcomes. In practice, each neuroimaging pipeline variant produces a statistic map -a complex, structured 3D output whose relevance can only be assessed with specific domain expertise, unlike simple metrics such as execution time. And, in most cases, there is no ground truth against which to judge these outputs, making it unclear which variant yields the best result. In this work, we introduce a "sampling, variant scoring, learning" methodology to study variability in the absence of a quantitative target -i.e. ground truth. We report our experience in developing an Universal Variability Language (UVL) feature model of 90 features representing the configuration space of a well-established open source neuroimaging analysis software (SPM). We sample and run 1000 valid configurations generating 1000 statistic maps as outputs. We studied various candidate proxy ground truths and computed Spearman correlations as a quantitative metric of the performance of each pipeline. We tested the following 12 proxy ground truths: the average statistic map (across variants), the output of an expertderived configuration, and a set of randomly selected outputs (as baseline). Then, we used a decision tree learning approach to inspect variability. We evaluated the sensitivity of our method to the choice of (proxy) ground truth, both in terms of predictive accuracy and in the identification of important features. These first results outline the challenge of choosing and validating a referential to assess our understanding of variability in the absence of ground truth.

PyroBuildS: Speeding up the exploration of large configuration spaces with incremental build [46] Software developers are acutely aware that software build is an essential but resource-intensive step in any software development process, all the more when building large and/or highly configurable systems, whose vast number of configuration options leads to an explosion in the number of variants to build and evaluate. A potential approach to speed up the builds of multiple configurations is to do incremental build, i.e., to not clean

the build environment and reuse previous builds when building a new configuration. Previous exploratory studies showed some benefits and limitations of incremental build, but mainly on small configurable software systems and on a limited set of close configurations. However, for large configuration spaces, little is known whether the large distance across configurations impacts the correctness and efficiency of incremental build. This work presents PyroBuildS, a new approach to speed up incremental builds while keeping reproducibility, featuring a configuration variation operator parameterized by two deny lists of problematic options and a mutation size (diversity). We evaluate PyroBuildS through an empirical study on three large complex configurable systems, namely Linux, BusyBox, and ToyBox, with respectively 18637, 1078, 330 configuration options. We first show that for all configurations PyroBuildS produces the exact same binaries as a clean build, except for Linux with some non-reproducible random configurations. We identify the reasons why incremental build speeds up or slows down the build of large configuration spaces – a knowledge that can be integrated into PyroBuildS. Incremental build systematically pays off, since problematic options are avoided in the first place — something only PyroBuildS does. We also show that a naive use of incremental build on random Linux configurations backfires, taking more time than clean builds. Thus, PyroBuildS controls diversity to avoid too many differences across configurations to perform efficient incremental builds. Thanks to its ability to operate over non-problematic options and close enough configurations, PyroBuildS significantly speeds up the exploration of large configuration spaces, with a gain in build time from 16% to 22% in all three systems with mutated configurations. Finally, with random configurations, PyroBuildS also speeds up the build time from 15% to 20% for ToyBox and BusyBox.

8.2.2 Variability

Variability Exploration for Decision Making: Supporting Domain Experts in Configuring Business Processes [38] Designing a model with built-in variability that can later be specialized for specific needs has become a common practice. This approach enables the consolidation of company expertise within a single model, extending its applicability beyond a single system, process, or behavior. Such modeling reveals a set of choices that Domain Experts (DEs) must evaluate, collectively forming the variability space-the range of potential decisions available to achieve desired project outcomes. Selecting the optimal decision within this variability space is often challenging for DEs, especially those without a technical background. The abundance of alternatives, each with the potential to significantly influence the capabilities of the final solution, adds complexity to the decision-making process. To assist DEs in exploring their models, we propose a tool-supported method for discovering and visualizing the variability space captured within feature models. This method allows experts to explore and evaluate different options against predefined objectives. By representing the variability space in a format conducive to decision-making, our method helps identify key choices that impact overall business processes, assess the implications of each option, and explore alternative configurations. We validate this method through a simplified case study of the OneWay project of Airbus, a leading international aircraft manufacturer. Applying our method to their feature model and business processes for avionics program development planning demonstrated its effectiveness in supporting decision-making activities and its overall performance.

Reinforcement Learning for Mutation Operator Selection in Automated Program Repair [40] Automated program repair techniques aim to aid software developers with the challenging task of fixing bugs. In heuristic-based program repair, a search space of mutated program variants is explored to find potential patches for bugs. Most commonly, every selection of a mutation operator during search is performed uniformly at random, which can generate many buggy, even uncompilable programs. Our goal is to reduce the generation of variants that do not compile or break intended functionality which waste considerable resources. In this work, we investigate the feasibility of a reinforcement learning-based approach for the selection of mutation operators in heuristic-based program repair. Our proposed approach is programming language, granularity-level, and search strategy agnostic and allows for easy augmentation into existing heuristic-based repair tools. We conducted an extensive empirical evaluation of four operator selection techniques, two reward types, two credit assignment strategies, two integration methods, and three sets of mutation operators using 30,080 independent repair attempts. We evaluated our approach on 353 real-world bugs from the Defects4J benchmark. The reinforcement learningbased mutation operator selection results in a higher number of test-passing variants, but does not exhibit a noticeable improvement in the number of bugs patched in comparison with the baseline, uniform random selection. While reinforcement learning has

been previously shown to be successful in improving the search of evolutionary algorithms, often used in heuristic-based program repair, it has yet to demonstrate such improvements when applied to this area of research.

Small Yet Configurable: Unveiling Null Variability in Software [83] Many small-scale software systems, that is, with limited codebase or binary size, are widely used in everyday tasks, yet their configurability remains largely unexplored. At the same time, studies on modern software systems show a trend toward increasing configurability, alongside growing interest in building immutable, specialized, and reproducible software. In this work, we present the first empirical study on the extent of configurability in small-scale software systems. By analyzing 108 programs from GNU coreutils, we show that even small programs can exhibit significant compile-time and run-time variability, with up to 76 options per program. Then, there is a high correlation (0.78) between run-time variability and codebase size. Furthermore, an analysis of the 20 smallest programs across 85 releases reveals that variability tends to increase over time, primarily due to the added compile-time variability. This suggests that shifting options between run-time and compile-time, removing unnecessary run-time variability, or resolving compile-time variability early, can help reduce codebase complexity and size. We also introduce, for the first time, the concept of null-variable software system, one with no configurability beyond mandatory features. Our findings show that high configurability is not exclusive to largescale systems and that reducing unnecessary variability can lead to lightweight, smaller, and more maintainable software. We hope this effort contributes to designing new software by understanding how to balance its configurability with codebase size.

A Comprehensive Survey of Benchmarks for Improvement of Software's Non-Functional Properties [30] Despite recent increase in research on improvement of non-functional properties of software, such as energy usage or program size, there is a lack of standard benchmarks for such work. This absence hinders progress in the field, and raises questions about the representativeness of current benchmarks of real-world software. To address these issues and facilitate further research on improvement of non-functional properties of software, we conducted a comprehensive survey on the benchmarks used in the field thus far. We searched five major online repositories of research work, collecting 5499 publications (4066 unique), and systematically identified relevant papers to construct a rich and diverse corpus of 425 relevant studies. We find that execution time is the most frequently improved property in research work (63%), while multi-objective improvement is rarely considered (7%). Static approaches for improvement of non-functional software properties are prevalent (51%), with exploratory approaches (18% evolutionary and 15% non-evolutionary) increasingly popular in the last 10 years. Only 39% of the 425 papers describe work that uses benchmark suites, rather than single software, of those SPEC is most popular (63 papers). We also provide recommendations for future work, noting, for instance, lack of benchmarks for non-functional improvement that covers Python, JavaScript, or mobile devices. All the details regarding the 425 identified papers are available on our dedicated webpage.

8.2.3 Performance and Energy consumption

On the Effect of Feature Reduction on Energy Consumption: An Exploratory Study [74] Energy consumption is a growing concern for sustainable software. Although increasingly studied, it remains largely unexplored in configurable systems growing in complexity with features. Feature reduction can eliminate software bloat, but to our knowledge, its impact on energy use has not been investigated. To fill this gap, we investigated how both on-demand and built-in feature reduction (defined later) affect the energy consumption of configurable systems. We conducted a first exploratory study using 28 programs from three systems with built-in feature reduction, namely ToyBox, BusyBox, and GNU, as well as 6 GNU programs debloated on-demand using the Chisel, Debop, and Cov tools. In our results, built-in feature reduction led to statistically significant energy decreases in 7% of the cases, while on-demand reduction, despite achieving energy decreases in 67% of cases, showed no statistical significance. However, when energy consumption increased, it was often more substantial than the reductions observed (occurring in 25% of built-in cases and 11% of on-demand cases) showing the complex and sometimes counterintuitive interplay between feature reduction and energy. Additionally, the observed strong correlation between energy consumption and execution time motivates a shift from traditional debloating goals, centered on binary size/attack surface, to energy-aware strategies that prioritize performance concerns. Finally, we provide an in-depth analysis and discuss the perspective.

Evaluating the Energy Profile of Tasks Managed by Build Automation Tools in Continuous Integration Workflows: The Case of Apache Maven and Gradle [58] There is a growing interest in the energy impact of computing-related activities, which is expected to increase in the coming years. Modern software development usually relies on Continuous Integration (CI) to support short iterations, where code changes are integrated on a daily basis. The implementation of CI workflows usually relies on build automation tools, (e.g. Apache Maven or Gradle), to automate several activities such as testing or compiling. The large adoption of CI practices raises concerns about the impact of such tasks usually run on cloud environments, where the underlying hardware and its associated energy consumption remain intangible to developers. To better understand the energy footprint related to modern software development, it is essential to investigate the energy profile of the tasks managed by Apache Maven and Gradle. To achieve such goal, we performed a large-scale study with 1167 CI workflows implemented through GitHub Actions and mined from popular Java projects hosted on GitHub. After executing them locally, in a controlled environment, we analyzed in depth the energy profile of 183 355 tasks managed by Apache Maven and Gradle. These tasks represent a quarter of the total energy consumption associated with the CI workflows. We found that tasks from workflows of small-sized projects do not necessarily consume less energy than tasks from workflows of medium-sized and large-sized projects. We also found that testing-related tasks consume the most energy, and that the larger the project, the higher the percentage of energy consumption related to testing. Moreover, tasks of different categories have a different profile regarding energy consumption per task and per unit of time.

Exploring Performance of Configurable Software Systems: the JHipster Case Study [55] The performance of software systems remains a key concern in software engineering. Configurable software systems, with their numerous configurations, complicate the performance evaluation process. This work investigates the impact of web stack configurations on performance, using the JHipster web stack generator as a case study. We analyze JHipster configurations to understand how component choices influence system performance and explore individual configuration options for their specific effects. Our study shows that correlations across performance indicators exist but are often weak, and different options affect performance unevenly, with some impacting one indicator minimally while significantly influencing another. We developed a performance model for JHipster to automate the identification of configurations optimized for specific metrics, identifying four configurations that outperform the current default. Overall, this study underscores JHipster's relevance as a use case for studying component-level variability in software systems and highlights the importance of selecting configurations based on performance indicators rather than preferred technologies.

Event-Driven Adaptation in the Computing Continuum Using Software Variability [59] The computing continuum aggregates edge, fog, and cloud infrastructure layers, promising lower latencies and improved performance for data-driven applications. In this context, keeping service level objectives (SLOs) is challenging due to fluctuations in resource capacities and network uncertainties. We present an adaptive runtime that manages software configuration parameters, application placement, and scaling at runtime to enforce SLOs. Our approach aims to maintain a separation between developers and providers roles while providing performance estimations for previously unseen configurations. This work describes the proposed architecture, event model, and preliminary results over the Grid'5000 testbed.

Linux Kernel Configurations at Scale: A Dataset for Performance and Evolution Analysis [69] Configuring the Linux kernel to meet specific requirements, such as binary size, is highly challenging due to its immense complexity-with over 15,000 interdependent options evolving rapidly across different versions. Although several studies have explored sampling strategies and machine learning methods to understand and predict the impact of configuration options, the literature still lacks a comprehensive and large-scale dataset encompassing multiple kernel versions along with detailed quantitative measurements. To bridge this gap, we introduce TuxKConfig, an accessible collection of kernel configurations spanning several kernel releases, specifically from versions 4.13 to 5.8. This dataset, gathered through automated tools and build processes, comprises over 240,000 kernel configurations systematically labeled with compilation outcomes and binary sizes. By providing detailed records of configuration evolution and capturing the intricate interplay among kernel options, our dataset enables innovative research in feature subset selection, prediction models based on machine learning, and transfer learning across kernel versions. Throughout this work, we describe how the dataset has been made easily accessible via OpenML and illustrate how it can be leveraged using only a few

lines of Python code to evaluate AI-based techniques, such as supervised machine learning. We anticipate that this dataset will significantly enhance reproducibility and foster new insights into configuration-space analysis at a scale that presents unique opportunities and inherent challenges, thereby advancing our understanding of the Linux kernel's configurability and evolution.

A systematic and large-scale exploration of analytical variability in task-fMRI [82] Configurability of neuroimaging pipelines is essential for tailoring the analyses to different experimental conditions but also gives rise to analytical variability : i.e. distinct pipelines, equally valid from a scientific point of view, may produce different findings. Previous works have explored the sources of this variability typically by studying the variability induced (in the results) by a small set of pre-defined pipeline variations – such as the choice of software, operating systems or preprocessing methods. In this work, we leverage methods from software engineering to systematically explore analytical variability from preprocessing up to group analysis in a multi-task fMRI dataset. We investigate 20 parameters – including interpolation algorithm, coregistration cost function, number of motion regressors – expressing a total of over 27,000 distinct pipelines. We propose a method to study variability in this large space without any assumptions made on the expected results or target downstream task. We separate variability of interest (differences observed across valid results) from unwanted variability (differences linked to pipelines producing unacceptable or low quality results) while minimizing the required amount of visual quality control. Finally, among the valid pipelines, we identify the parameters that have the strongest impact on the final results.

8.2.4 Variability and generative AI

Piloting Copilot, Codex, and StarCoder2: Hot temperature, cold prompts, or black magic? [37] Language models are promising solutions for tackling increasing complex problems. In software engineering, they recently gained attention in code assistants, which generate programs from a natural language task description (prompt). They have the potential to save time and effort but remain poorly understood, limiting their optimal use. In this work, we investigate the impact of input variations on two configurations of a language model, focusing on parameters such as task description, surrounding context, model creativity, and the number of generated solutions. We design specific operators to modify these inputs and apply them to three LLM-based code assistants (Copilot, Codex, StarCoder2) and two benchmarks representing algorithmic problems (HumanEval, LeetCode). Our study examines whether these variations significantly affect program quality and how these effects generalize across models. Our results show that varying input parameters can greatly improve performance, achieving up to 79.27% success in one-shot generation compared to 22.44% for Codex and 31.1% for Copilot in default settings. Actioning this potential in practice is challenging due to the complex interplay in our study-the optimal settings for temperature, prompt, and number of generated solutions vary by problem.

Generative AI-based Adaptation in Microservices Architectures: A Systematic Mapping Study [65] Microservices have seen widespread adoption in academia and industry. Despite their benefits, challenges persist in resilience, performance, scalability, and adaptation to dynamic contexts. Generative AI (GenAI) has emerged as a promising approach to address these issues, though concerns remain about the suitability of various models and potential drawbacks. To assess the state of the art, we conducted a systematic mapping study analyzing 22 primary studies. Results reveal significant potential of GenAI in enhancing microservice adaptation, with emphasis on Large Language Models and optimization techniques. Applications primarily target maintenance and monitoring, especially anomaly management. This study also highlights research gaps and outlines future directions to advance GenAI integration for more resilient and autonomous microservices architectures.

LLM Code Customization with Visual Results: A Benchmark on TikZ [62] With the rise of AI-based code generation, customizing existing code out of natural language instructions to modify visual results -such as figures or images -has become possible, promising to reduce the need for deep programming expertise. However, even experienced developers can struggle with this task, as it requires identifying relevant code regions (feature location), generating valid code variants, and ensuring the modifications reliably align with user intent. In this work, we introduce vTikZ, the first benchmark designed to evaluate the ability of Large

Language Models (LLMs) to customize code while preserving coherent visual outcomes. Our benchmark consists of carefully curated vTikZ editing scenarios, parameterized ground truths, and a reviewing tool that leverages visual feedback to assess correctness. Empirical evaluation with state-of-the-art LLMs shows that existing solutions struggle to reliably modify code in alignment with visual intent, highlighting a gap in current AI-assisted code editing approaches. We argue that vTikZ opens new research directions for integrating LLMs with visual feedback mechanisms to improve code customization tasks in various domains beyond TikZ, including image processing, art creation, Web design, and 3D modeling.

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, Walter Rudametkin, Paul Temple, Olivier Zendra.

8.3.1 Cybersecurity forecast and policy aspects

HiPEAC Vision 2025 [75] In the wake of a series of reports painting a bleak picture of European competitiveness – and hence future prosperity – the HiPEAC Vision 2025 takes a clear-eyed look at the computing sector in the European Union and sets out a series of actionable recommendations applicable to the next 10 years of computing research, covering the computing continuum from edge to cloud to high-performance computing, and tailored to the European context. Taking an analysis of the state of the technology sector as its starting point, the roadmap describes how to implement the "next computing paradigm" (NCP): a computing continuum tailored to the user, where data and code migrate to where it makes sense to process and execute them, and which draws on European strengths such as edge computing, digital twins, factory automation, and management of complex systems to deliver technological solutions to real-world problems. Chapters in this year's HiPEAC Vision cover the state of the European Union, the NCP, artificial intelligence (AI), new hardware, tools, cyber-physical systems, cybersecurity and sustainability. The document also includes a series of recommendations covering technological themes, methodological approaches, standardization issues, and policy directions.

Software Identification for Cybersecurity: Survey and Recommendations for Regulators [79] Global momentum around software supply chain security has increased over the last few years. High-profile cybersecurity incidents — together with new regulatory imperatives — underscore the need for robust, verifiable identification of all software components, to clearly identify software artifacts whose vulnerabilities are the root cause of potential attacks. Yet, existing naming schemes and repository-based references often prove ephemeral, inexact, or insufficiently secure. By contrast, content-based, persistent software identifiers enable unambiguous references that outlive any particular development platform or hosting service, thus providing a pathway to meet these newly mandated obligations. We contend that SWHIDs (Software Hash IDentifiers) — now international standard under ISO/IEC 18670 — offer a relevant and practical solution to unify software identification requirements in the EU Cyber Resilience Act, the U.S. Executive Orders, and broader international policy frameworks. Combined with the Software Heritage archival infrastructure, SWHIDs enable transparent, verifiable identification of source code at various granularities (files, directories, version control systems commits, etc.) and, by extension, of any derived artifact. We examined the current situation, which requires a reflection on the challenges related to the identifications of open source components. Next, we analyzed prevalent identifier architectures deployed in open source development and Software Bill of Materials (SBOM) standardization frameworks. Building on this foundation, we introduced Software Heritage persistent Identifiers (SWHIDs), emphasizing their structural advantages and pivotal role in addressing compliance requirements outlined in modern regulatory regimes. Next, we formulated a strategic implementation roadmap to facilitate cross-sector adoption of SWHIDs, delineating actionable pathways for integration across governmental policy frameworks, industrial supply chains, and open source communities. Finally, we call to action regulators, industry and OSS projects on this topic

8.3.2 Vulnerabilities in source code

Did You Forkget It? Detecting One-Day Vulnerabilities in Open-source Forks With Global History Analysis [81] Tracking vulnerabilities inherited from third-party open-source software is a well-known challenge, often addressed by tracing the threads of dependency information. However, vulnerabilities can also propagate through forking: a code repository forked after the introduction of a vulnerability, but before it is patched, may remain vulnerable long after the vulnerability has been fixed in the initial repository. History analysis approaches are used to track vulnerable software versions at scale. However, such approaches fail to track vulnerabilities in forks, leaving fork maintainers to identify them manually. This work presents a global history analysis approach to help software developers identify one-day (known but unpatched) vulnerabilities in forked repositories. Leveraging the global graph of public code, as captured by the Software Heritage archive, our approach propagates vulnerability information at the commit level and performs automated impact analysis. Starting from 7162 repositories with vulnerable commits listed in OSV, we propagate vulnerability information to 2.2 million forks. We evaluate our approach by filtering forks with significant user bases whose latest commit is still potentially vulnerable, manually auditing the code, and contacting maintainers for confirmation and responsible disclosure. This process identified 135 high-severity one-day vulnerabilities, achieving a precision of 0.69, with 9 confirmed by maintainers.

8.3.3 AI security

Approaches for strengthening embedded AI against attacks disrupting federated learning [72] Machine learning is increasingly integrated into our daily lives, particularly through personalized systems. However, data confidentiality poses a major challenge. Federated learning (FL) addresses this issue by enabling models to be trained without sharing sensitive data. However, although FL guarantees data confidentiality, it remains vulnerable to various attacks, such as data poisoning attacks or inference on model weights. The observation made is that all the models deployed in this FL context can be considered as variants of the global model, thus echoing the world of software variability. This thesis therefore proposes to adapt software testing and variability management techniques to reinforce the security and robustness of the FL, taking into account its evolution over time and its specificities.

8.3.4 Systems resilience via self-adaptation

Breaking the Loop: AWARE is the new MAPE-K [66] Self-adaptive systems have traditionally relied on the MAPE-K loop. It consists of a centralized, reactive, and sequential loop for monitoring, analyzing, planning, and executing system adaptations. However, the increasing complexity and dynamic nature of modern systems have exposed the limitations of MAPE-K loops, including their lack of proactivity, scalability challenges, and difficulty integrating continuous learning or distributed decision-making. We introduce AWARE (Assess, Weigh, Act, Reflect, Enrich), a distributed, goal-driven framework that addresses these limitations. AWARE employs autonomous AI agents capable of proactive adaptation, collaboration, and continuous learning to enhance decision-making and system resilience. The modular design of our framework supports dynamic agent integration and optimized resource utilization, enabling seamless scalability and adaptability. AWARE not only anticipates changes and optimizes responses but also iteratively refines its strategies based on contextual insights. Through a comparison with MAPE-K and a real-world use case, we demonstrate how AWARE-distributed intelligence redefines the capabilities of self-adaptive systems, offering a solution better aligned with the demands of complex real-world systems.

8.3.5 Browser and privacy

FP-Rainbow: Fingerprint-Based Browser Configuration Identification [57] Browser fingerprinting is a tracking technique that collects attributes and calls functions from the browser's APIs. Unlike cookies, browser fingerprints are difficult to evade or delete, raising significant privacy concerns for users as they can be used to re-identify individuals over browsing sessions without their consent. Yet, there has been limited research on the impact of browser configuration settings on these fingerprints. This work introduces FP-Rainbow, a novel approach to systematically explore and map the configuration space of Chromium-based web browsers aiming to identify the impact of configuration parameters on browser fingerprints and their changes over time. We explore 1,748 configuration parameters (switches) and identify their impact on

the browser's BOM (Browser Object Model). By collecting and analyzing over 61,000 fingerprints from 18 versions of Chromium, our study reveals that 32 to 56 of these configuration parameters (depending on versions), such as disable-3d-apis or disable-notifications, influence the fingerprint of a web browser. FP-Rainbow also proves efficient in identifying browser configuration parameters from unknown fingerprints, achieving an average successful identification rate of 84% when considering a single configuration parameter and 78% when multiple parameters are involved, across all evaluated browser versions. These findings emphasize the importance of measuring the impact of configuration parameters on browsers to develop safer and more privacy-friendly web browsers.

BrowserFM: A Feature Model-based Approach to Browser Fingerprint Analysis [56] Web browsers have become complex tools used by billions of people. The complexity is in large part due to its adaptability and variability as a deployment platform for modern applications, with features continuously being added. This also has the side effect of exposing configuration and hardware properties that are exploited by browser fingerprinting techniques. In this work, we generate a large dataset of browser fingerprints using multiple browser versions, system and hardware configurations, and describe a tool that allows reasoning over the links between configuration parameters and browser fingerprints. We argue that using generated datasets that exhaustively explore configurations provides developers, and attackers, with important information related to the links between configuration parameters (i.e., browser, system and hardware configurations) and their exhibited browser fingerprints. We also exploit Browser Object Model (BOM) enumeration to obtain exhaustive browser fingerprints composed of up to 16,000 attributes. We propose to represent browser fingerprints and their configurations with feature models, a tree-based representation commonly used in Software Product Line Engineering (SPLE) to respond to the challenges of variability, to provide a better abstraction to represent browser fingerprints and configurations. With translate 89,486 browser fingerprints into a feature model with 35,857 nodes from 1, 748 configurations. We show the advantages of this approach, a more elegant tree-based solution, and propose an API to query the dataset. With these tools and our exhaustive configuration exploration, we provide multiple use cases, including differences between headless and headful browsers or the selection of a minimal set of attributes from browser fingerprints to re-identify a configuration parameter from the browser.

9 Bilateral contracts and grants with industry

9.1 Bilateral contracts with industry

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.

CGI

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

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

Sopra Steria

Participants: Mathieu Acher, Djamel Khelladi.

- Partners: Inria/Sopra Steria (as part of LLM4Code)
- Dates: 2024-2027
- Abstract: LLM-based techniques for code modernization. One PhD thesis (CIFRE) and one post-doc (24 months) have been recruited and are working on migrating legacy systems, typically written in COBOL, with modern stacks (eg Java). The challenge is not only to master the COBOL programming language (and many variants/subtleties), but also to re-achitect the system with modern software engineering principles (modularity, abstraction, testability), knowing that lots of business knowledge is somehow lost in thousands of lines of code. Another related challenge is to validate and verify that the new migrated system works as expected (e.g., through testing).

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 -> 2027

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 -> 2027

Coordinator: Arnaud Gotlieb (arnaud@simula.no)

Partners:

- Simula (Norvège)

Inria contact: Mathieu Acher

Summary: Resilient and RePrOducible SofTware The associated team Resilient and Reproducible Software (RIPOST) will research the foundations of "Resilient Software" - software systems which can resist failures without significantly degrading their functionality. Over the past years, resilient software systems have become extremely important in various application domains. By combining Inria's and Simula's expertise in software engineering, resilient software, and AI-based software testing, the RIPOST associate team will address the following challenges:

- Reproducing experiments with deep variability,
- Replication through automated variation of computational experiments,
- Safe and minimal explanations for reproducible scenarios Interpretable explanations for reproducible and replicable scenarios,

The goal of the RIPOST associate team is to address systematically the reproducibility testing challenge in computer science. It is the continuation of the earlier associate team RESIST EA. The RIPOST associate team is composed of ten researchers, five from the Simula department VIAS within the Software Engineering area and five researchers from the DiverSE team at Inria Rennes. The associate team will be jointly led by Professor Mathieu Acher at the University of Rennes (INSA Rennes, DiverSE Inria), and Research Professor Arnaud Gotlieb from Simula in Norway. More information with activities, publications, etc.: here

10.1.3 COFECUB PUC Rio

Title: Learning from Software Evolution to Learn Software Variability

Duration: 2024–2027

Coordinator: Alves Pereira

Partners:

- PUC Rio (Brazil)
- Université de Rennes / IRISA / Inria (France)

Inria contact: Mathieu Acher

Summary: This CAPES–COFECUB project focuses on leveraging software evolution data to better understand and manage software variability in highly configurable systems. Using the Linux kernel as a primary case study, the project investigates machine learning techniques for performance prediction, feature selection, interpretability, and reproducible experimentation. It supports strong bilateral collaboration through joint supervision, shared datasets, co-authored publications, and reciprocal research visits between PUC-Rio and Inria/IRISA. The project contributes to the training of graduate and undergraduate students and strengthens long-term research ties between France and Brazil in software engineering.

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.

10.2.2 Visits to international teams

Research stays abroad - Jean-Baptise Doderlein visited CWI (Prof. Tijs von der Storm), the Netherlands, for two months.

10.3 European initiatives

10.3.1 Horizon Europe

HiPEAC HiPEAC project on cordis.europa.eu

Title: High Performance, Edge And Cloud computing

Duration: From December 1, 2022 to July 31, 2026

Partners:

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

Inria contact: Olivier Zendra

Coordinator: Koen De Bosschere, UGent

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)

- Coordinator: Olga Hendel and Alessio Bucaioni Mälardalen University, Sweden
- Local coordinator: Benoit Combemale
- Other local participant: Djamel Eddine Khelladi
- Dates: 2024-2027
- Budget: 200 k€
- 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 behaviors, 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

PEEPS ANR JCJC

Participants: Stéphanie Challita.

- Coordinator: Stéphanie Challita
- DiverSE, University of Rennes
- Dates: 2025-2030
- Abstract: Representational State Transfer (REST) is the dominant architectural style for Web APIs, with thousands of APIs currently in use across domains such as social networking, e-commerce, and weather services. However, the development and use of REST APIs remain challenging due to inconsistencies between business requirements, API code, and documentation, which are produced by different stakeholders using heterogeneous concepts and terminology. These inconsistencies lead to ambiguities, misunderstandings, and misuses, with studies reporting inappropriate usage examples in a significant proportion of API endpoints. The PEEPS project addresses this issue by promoting coherence between requirements, code, and documentation, referred to as preciseness in REST APIs, through higher-level abstractions and explicit bridges between these three facets.

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 PEPR

PEPR Cloud Taranis

Participants: Olivier Barais, Stéphanie Challita, Paul Temple.

- 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.

10.4.4 Campus Cyber

Software Heritage Sec

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

- Coordinator: INRIA
- Partners: INRIA, IMT, CEA, Université Sorbonne.
- 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.

10.4.5 Défi LLM4Code

Mathieu Acher is co-responsible with Guillaume Baudart of the défi LLM4Code. DiverSE is co-supervising 3 PhD students and 1 postdoc as part of the défi (and also the collaboration with Sopra Steria)

Participants: Mathieu Acher, Djamel Khelladi, Aymeric Blot, Theo Matricon.

- Abstract: Generative AI, in particular the recent Large Language Models (LLMs), show great promise for software developments. Specialized models are now able to perform an impressive variety of programming tasks: solving programming exercises, assisting software developers, or even generating mechanized proofs. Yet, many challenges still need to be addressed to build reliable and productive LLM-based coding assistants: improving the quality of the generated code, increasing the developers' confidence in the generated code, enabling interaction with other software development tools (verification, test), and providing new capabilities (automated migration and evolution of software). The goal of the Défi Inria LLM4Code is to leverage LLM capabilities to build code assistants that can enhance both reliability and productivity. The défi is organized along three work packages (Self-improving code generation, Evolution of existing software, Interactive tools with AI-in-the-loop).
- Dates: 2024-2028

10.4.6 Code Commons

Participants: Mathieu Acher, Djamel Khelladi, Arnaud Blouin, Theo Matricon.

- Abstract: Building on the existing foundation of Software Heritage, the largest publicly available source code archive, CodeCommons aims to bring into one place all the critical and qualified information needed to create smaller, better datasets for the next generation of AI tools. At its core, the project prioritizes transparency and traceability, enabling model builders and users to respect creators' rights while promoting sovereign and sustainable AI.
- Dates: 2025-2028
- DiverSE, as part of the CodeCommons project, has recruited 8 engineers and 1 postdoc.

10.4.7 Défi FRAIME with MERCE and Mitsubishi

Participants: Mathieu Acher, Djamel Khelladi, Aymeric Blot, Theo Matricon.

- Abstract: Mitsubishi Electric Corporation and Inria launched a joint research project titled “Formal Reasoning applied to AI for Methodological Engineering” (FRAIME) with the aim of realizing trustworthy AI systems. The goal is to achieve trustworthy AI systems and establish next-generation AI technology by integrating Formal Methods technologies, a mathematical approach, with AI technologies.
- 2 PhD students and 1 postdoc

10.5 Regional initiatives

ANIME-WATER (IRIS-E)

- Coordinator: Laurent Longuevergne (Géosciences Rennes, CNRS)
- Participants: Benoit Combemale, Djamel Eddine Khelladi
- Partners: Géosciences Rennes, ESO Rennes
- Dates: 2024-2027
- Budget: 125 k€
- Abstract: ANIME-WATER addresses the challenge to formulate new water management and sharing rules under uncertainties allowing for anticipation of future climate changes and taking into account human-non human interactions. This endeavor involves the creation of territorial digital twins, designed to bolster territorial foresight and water governance. These digital twins function as mediation tools, tailored to local expectations and enriched by a suite of generic hydrological models.

Allocation d’Installation Scientifique (AIS) - Rennes Métropole

Participants: Stéphanie Challita.

- Coordinator: Stéphanie Challita
- Dates: 2024-2026
- Abstract: This 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

Mathieu Acher Olivier Barais Arnaud Blouin Aymeric Blot Stéphanie Challita Benoît Combemale Djamel Khelladi Jean-Marc Jezequel Noël Plouzeau Walter Rudametkin Ivey Olivier Zendra

11.1 Promoting scientific activities

11.1.1 Scientific events: organisation

Member of the organizing committees Jean-Marc Jézéquel has organized ECSS'25 in Rennes, the anual conference of Informatics Europe.

Djamel Khelladi has Organized ERMSEI 2025 in Rennes, the summer school on Empirical Research Methods in Software engineering and Informatics.

Aymeric Blot has been a co-organizer of GI@ICSE'25, a scientific workshop on genetic improvement which took place within ICSE, the premier software engineering conference.

Benoit Combemale co-organized the 1st 1-week Workshop on Model Hybridization for Digital Twins (Bellairs, McGill Institute, February 2025), and the 4th 1-week Winter Modeling Meeting (Italy, February, 2025).

The DiverSE team has organized the 19th International Working Conference on Variability Modelling of Software-Intensive Systems (aka VaMoS) 4-6 February 2025 in Rennes, France. The general chair was Mathieu Acher, the local arrangements chairs were Djamel Eddine Khelladi and Paul Temple, the publicity chair was Olivier Zendra, while Charly Reux, Heraldo Pimenta Borges Filho, Jolan Philippe, Romain Lefeuvre were part of the organization.

DJamel Khelladi co-organized the Models and Evolution workshop colocated with MODELS 2025.

11.1.2 Scientific events: selection

Member of the conference program committees Stéphanie Challita has been a member of the following PCs:

- Technical track of ECMFA 2025
- NIER track of MODELS 2025

Olivier Barais has been member of the following PCs:

- European Conference on Software Engineering Education Zum Inhalt springen 2025.
- DebConf scientific track 2025.

Arnaud Blouin has been a member of the following PCs:

- SAC 2025
- EICS 2025

Mathieu Acher has been a member of the following PCs:

- ECAI 2025
- SPLC 2025
- MODEVAR@SPLC 2025

Benoit Combemale has been a member of the following PCs:

- MODELS 2025 (Program Board)
- RE 2025
- EDTconf 2025
- SusMod 2025

Djamel Khelladi has been a member of the following PCs:

- ICSE 2025/2026

- FSE 2025/2026
- ASE 2025
- MSR 2025
- MODELS 2025
- ECMFA 2025

Olivier Zendra has been a member of the following PCs:

- 20th International Workshop on Implementation, Compilation, Optimization of OO Languages, Programs and Systems (ICOOOLPS 2025).
- 31st Computer & Electronics Security Application Rendezvous (C&ESAR 2025)
- 26th Debian Conference Academic Track (DebConf25 AcademicTrack)

11.1.3 Journal

Benoit Combemale is Editor-in-Chief of the Journal of Software and Systems Modeling, Springer (SoSyM).

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, TOSEM, Sosym, JSS, EMSE, Jot, SPE, IEEE Software, IST, ...

11.1.4 Invited talks

Olivier Barais give a keynote on software supply chain security @ RESSI 2025.

Mathieu Acher gave the following talks:

- Software Variability for Replicable Science Brazilian Symposium on Software Components, Architectures, and Reuse @ CBSoft 2025
- Lighting talk at Lorentz center on "Rethinking Software Systems" mainly about reproducible science, generative AI, variability, and software engineering
- Can Generative AI Generate Computational Experiments? Organized by ORAP
- Who Codes When AI Can Generate Code? Impacts of Generative AI on Professional Developers, End Users, and Researchers séminaire « De l'impact de l'IA-générative sur le génie logiciel » SOTELO seminar organized by CEA
- DiverSE Team activities review (pre-recorded) on Project context metadata collection and Programming language identification @ CodeCommons annual review

Benoit Combemale gave the following talks:

- "Engineering Digital Twins", UDST Winter School (Doha, Qatar)
- "Unlocking the Potential of Digital Twins Experiences in Manufacturing and Engineering", SPACERAISE Summer School (L'Aquila, Italy)

Olivier Zendra has been invited to a panel on the CRA (EU Cyber Resilience Act) at the Software Heritage Symposium at UNESCO Paris (01/2025)

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.

Mathieu Acher is a member of the steering committees for the conference SPLC, VaMoS, and VARIABILITY.

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.

Arnaud Blouin is a member of the IFIP WG 2.7 (aka 13.4) devoted to User Interface Engineering.

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.

11.1.6 Scientific expertise

Olivier Zendra and Arnaud Blouin are scientific CIR/JEI experts 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.

Benoit Combemale is software engineering officer ("chargé de mission sciences du logiciel") for Inria in the context of his national mission.

11.1.7 Research administration

Olivier Barais is a member of the CNU 27.

11.2 Teaching - Supervision - Juries - Educational and pedagogical outreach

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

Zohra Kebaili (defended the 15th of May 2025). Mathieu Acher, Olivier Barais and Djamel Eddine Khelladi are co-supervisors of this 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 cosupervisors of this thesis.

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

Lise 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-supervisors 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.

Theo Giraudet (defense in 2026). Arnaud Blouin and Benoit Combemale are co-supervisors of this thesis.

Haitam El Hayani (defense in 2027). Olivier Barais, Stéphanie Challita and Benoît Combemale 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 cosupervisors of this thesis.

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

Axel Allain (defense in 2028). Mathieu Acher, Djamel Eddine Khelladi and Aymeric Blot are co-supervisors of this thesis.

Valère Billaud (defense in 2028). Olivier Zendra, Olivier Barais and Paul Temple are co-supervisors of this thesis.

Malvin Chevallier (defense in 2028). Olivier Zendra, Olivier Barais and Paul Temple are co-supervisors of this thesis.

Jean-Baptiste Doderlein (defense in 2028). Benoit Combemale, Djamel Eddine Khelladi are co-supervisors of this thesis.

Imene Issolah (defense in 2028). Benoit Combemale, Djamel Eddine Khelladi are co-supervisors of this thesis.

Maiwenn Le Goasteller (defense in 2028). Olivier Barais, Aymeric Blot and Quentin Perez are co-supervisors of this thesis.

Sami Chtitah (defense in 2028). David Gross-Amblard (research team Druid) and Benoit Combemale are co-supervisors of this thesis.

Jean-Baptiste Espinasse (defense in 2028). Mathieu Acher and Djamel Eddine Khelladi are co-supervisors of this thesis.

Jacob Kohav (defense in 2028). Mathieu Acher and Walter Rudametkin are co-supervisors of this thesis.

Yann Paillard started a PhD that he gave up on after three days in the lab to get a software engineer position at Luxembourg.

11.2.3 Juries

Stéphanie Challita was in the PhD jury (examiner) of Hiba Awad (IMT Atlantique, Nantes, France).

Stéphanie Challita has been a member of a recruiting committee for MCF at Nantes Université.

Arnaud Blouin was in the PhD jury (examiner) of Axel Carayon (University of Toulouse, France).

Arnaud Blouin has been a member of a recruiting committee for associate professor (MCF) at Toulouse University.

Olivier Barais has been a member of a recruiting committee for associate professor (MCF) at Telecom Sud Paris.

Olivier Barais has been a member of a recruiting committee for associate professor (MCF) at Grenoble INP (Ensimag).

Olivier Barais has been a member of a recruiting committee for a full professor (PR) at Univ Rennes (Esir).

Olivier Barais has been a member of a HCERES committee for LIG Lab (Grenoble).

Benoit Combemale has been a member of the PhD Jury (Reviewer) of Christopher Esterhuyse (University of Amsterdam, NL).

Olivier Barais has been a member of the PhD Jury (Reviewer) of Minh Khoi Nguyen (Université de Toulouse).

Olivier Barais has been a member of the PhD Jury (Reviewer) of Nassim Bounouas (Université Côte d'Azur).

Olivier Barais has been a member of the PhD Jury (President) of Damien Jaime (Sorbonne université).

Olivier Barais has been a member of the PhD Jury (Reviewer) of Biagio MONTARULI (Eurocom, Nice).

Olivier Barais has been a member of the HdR Jury (Reviewer) of Pierre Laperdrix (Université de Lille).

Olivier Barais has been a member of the PhD Jury (Reviewer) of Gabriel Darbord (Université de Lille).

Olivier Barais has been a member of the PhD Jury (Examiner) of Alexandre Bonvoisin (Université de Lille).

Olivier Barais has been a member of the PhD Jury (Reviewer) of Jean-Marie Mineau (Supelec Rennes).

Djamel Khelladi was in the PhD jury (reviewer) of Adiel Tuyishime, (GSSI - Gran Sasso Science Institute, L'Aquila, Italy).

Djamel Khelladi has been a member of two recruiting committees for associate professor (MCF) at Nice University and at Paris Nanterre University.

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] S. Abrahão, M. Staron, J. Michael, B. Combemale and M. Chechik. 'Modeling and Digital Twins: Insights and Strategies for Software Engineers'. In: *IEEE Software* (2025). URL: https://inria.hal.science/hal-05485243 (cit. on p. 24).

[30] A. Blot and J. Petke. 'A Comprehensive Survey of Benchmarks for Improvement of Software's Non-Functional Properties'. In: *ACM Computing Surveys* 57.7 (20th Feb. 2025), 1–36 / Article n°168. DOI: 10.1145/3711119. URL: https://hal.science/hal-04936383 (cit. on p. 30).

[31] S. Challita, M. Chechik, B. Combemale, H. Ergin, J. Gray, B. Rumpe and M. Schindler. 'Antonio Vallecillo'. In: *Software and Systems Modeling* 24 (8th May 2025), pp. 593–594. DOI: 10.1007/s10270-025-01290-5. URL: https://inria.hal.science/hal-05487852.

[32] S. Challita, B. Combemale, H. Ergin, J. Gray, B. Rumpe and M. Schindler. 'Report on the State of the SoSyM Journal (2024 summary)'. In: *Software and Systems Modeling* (20th Feb. 2025). URL: https://inria.hal.science/hal-05487850.

[33] M. Chechik, B. Combemale, J. Gray and B. Rumpe. 'Formal methods in the scope of the Software and Systems Modeling journal'. In: *Software and Systems Modeling* 24.2 (2025), pp. 271–272. DOI: 10.1007/s10270-025-01287-0. URL: https://hal.science/hal-05108048.

[34] M. Chechik, B. Combemale, J. Gray and B. Rumpe. 'On theory and management of dependencies between models'. In: *Software and Systems Modeling* 24 (20th June 2025), pp. 975–976. DOI: 10.1007/s10270-025-01301-5. URL: https://inria.hal.science/hal-05487855.

[35] M. Chechik, B. Combemale, J. Gray and B. Rumpe. 'Pragmatic specification of software behavior, configuration, and orchestration: the precision and usability of domain-specific modeling'. In: *Software and Systems Modeling* 24 (7th Nov. 2025), pp. 1621–1622. DOI: 10.1007/s10270-025-01339-5. URL: https://inria.hal.science/hal-05487858.

[36] M. Chechik, B. Combemale, J. Gray and B. Rumpe. 'Standards in software development and modeling'. In: *Software and Systems Modeling* 24 (2nd Aug. 2025), pp. 1315–1316. DOI: 10.1007/s10270-025-01312-2. URL: https://inria.hal.science/hal-05487857.

[37] J.-B. Döderlein, N. H. Kouadio, M. Acher, D. E. Khelladi and B. Combemale. 'Piloting Copilot, Codex, and StarCoder2: Hot temperature, cold prompts, or black magic?' In: *Journal of Systems and Software* 230 (Dec. 2025), p. 112562. DOI: 10.1016/j.jss.2025.112562. URL: https://hal.science/hal-05474229 (cit. on p. 32).

[38] H. El Hayani, B. Combemale, O. Barais and S. Zschaler. 'Variability Exploration for Decision Making: Supporting Domain Experts in Configuring Business Processes'. In: *The Journal of Object Technology* (2025), pp. 1–12. URL: https://inria.hal.science/hal-05029778 (cit. on p. 29).

[39] T. Giraudet, A. Blouin, B. Combemale, M. Bats and P.-C. David. 'Augmenting graphical modeling workbenches with semantic-aware interactive features'. In: *Proceedings of the ACM on Human-Computer Interaction* 9.4 (30th June 2025), pp. 1–29. DOI: 10.1145/3733050. URL: https://inria.hal.science/hal-04931045 (cit. on p. 25).

[40] C. Hanna, A. Blot and J. Petke. 'Reinforcement Learning for Mutation Operator Selection in Automated Program Repair'. In: *Automated Software Engineering* 32.2 (2025), pp. 1–31. DOI: 10.1007/s10515-025-00501-z. URL: https://hal.science/hal-04996050 (cit. on p. 29).

[41] 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), pp. 1–24. DOI: 10.22152/programming-journal.org/2025/10/13. URL: https://hal.science/hal-04906879 (cit. on p. 24).

[42] Z. K. Kebaili, D. E. Khelladi, M. Acher and O. Barais. 'Automated co-evolution of metamodels and code'. In: *IEEE Transactions on Software Engineering* 51.4 (20th Feb. 2025), pp. 1067–1085. DOI: 10.1109/tse.2025.3540545. URL: https://hal.science/hal-04973171 (cit. on p. 26).

[43] F. Khorram, E. Bousse, J.-M. Mottu, G. Sunyé, D. E. Khelladi, P. Gómez-Abajo, P. Cañizares, E. Guerra and J. de Lara. 'A language-parametric test amplification framework for executable domain-specific languages'. In: *Software and Systems Modeling* 24.4 (27th May 2025), pp. 1187–1212. DOI: 10.1007/s10270-025-01283-4. URL: https://hal.science/hal-05377263 (cit. on p. 27).

[44] A. Martin, D. E. Khelladi, T. Matricon and M. Acher. 'Re-evaluating Metamorphic Testing of Chess Engines: A Replication Study'. In: *Information and Software Technology* (Feb. 2025), pp. 1–38. DOI: 10.1016/j.infsof.2025.107679. URL: https://hal.science/hal-04943474 (cit. on p. 28).

[45] J. Michael, L. Cleophas, S. Zschaler, T. Clark, B. Combemale, T. Godfrey, D. E. Khelladi, V. Kulkarni, D. Lehner, B. Rumpe, M. Wimmer, A. Wortmann, S. Ali, B. Barn, I. Barosan, N. Bencomo, F. Bordeleau, G. Grossmann, G. Karsai, O. Kopp, B. Mitschang, P. Muñoz Ariza, A. Pierantonio, F. Polack, M. Riebisch, H. Schlingloff, M. Stumptner, A. Vallecillo, M. van den Brand and H. Vangheluwe. 'Model-Driven Engineering for Digital Twins: Opportunities and Challenges'. In: *Systems Engineering* 28.5 (2nd Apr. 2025), pp. 659–670. DOI: 10.1002/sys.21815. URL: https://inria.hal.science/hal-05060561 (cit. on p. 23).

[46] G. A. Randrianaina, D. E. Khelladi, O. Zendra and M. Acher. 'PyroBuildS: Speeding up the exploration of large configuration spaces with incremental build'. In: *Journal of Systems and Software* 231 (2026), p. 112592. DOI: 10.1016/j.jss.2025.112592. URL: https://hal.science/hal-05213625 (cit. on p. 28).

Invited conferences

[47] M. Acher. 'Software Variability for Replicable Science'. In: CBSoft 2025 - Congresso Brasileiro de Software: Teoria e Prática. Rio, Brazil, 23rd Sept. 2025. URL: https://hal.science/hal-05334168 (cit. on p. 27).

[48] B. Jonglez, M. Simonin, J. Philippe and S. M. Kaddour. 'Multi-provider capabilities in EnOSlib: driving distributed system experiments on the edge-to-cloud continuum'. In: *Springer LNCS-IFIP, Lecture Notes in Computer Science (LNCS)*. DAIS 2025: 25th International Conference on Distributed Applications and Interoperable Systems. Vol. 15730. Lille, France: Springer, 2025, pp. 25–42. DOI: 10.1007/978-3-031-95728-4_2. URL: https://inria.hal.science/hal-05052776.

International peer-reviewed conferences

[49] M. Acher, A. Gotlieb, H. Spieker and G. Le Bartz Lyan. 'Teaching Reproducibility and Embracing Variability: From Floating-Point Experiments to Replicating Research'. In: REP 2025 - ACM Conference on Reproducibility and Replicability. Vancouver (BC), Canada, 2025, pp. 1–9. DOI: 10.1145/3736731.3746162. URL: https://inria.hal.science/hal-05190848 (cit. on p. 27).

[50] T. Alskaif, Ö. Babur, F. Bordeleau, L. Cleophas, B. Combemale, J. Denil, Ø. Haugen, J. Michael, P. H. Nguyen, T. Seceleanu, M. van den Brand and H. Vangheluwe. 'Evolution at the Core of Digital Twin Engineering'. In: *Proceedings of the 2nd International Conference on Engineering Digital Twins*. EDTconf 2025 - 2nd International Conference on Engineering Digital Twins. Grand Rapids, Michigan, United States, 2025, pp. 1–6. URL: https://inria.hal.science/hal-05221886 (cit. on p. 23).

[51] 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: DATE 2025 - Design, Automation and Test in Europe Conference. Lyon, France, 2025, pp. 1–6. DOI: 10.23919/DATE64628.2025.10993046. URL: https://inria.hal.science/hal-04839759.

[52] A. Bucchiarone, B. Combemale, A. Pierantonio, N. Bencomo, M. van den Brand, J.-M. Bruel, A. Cicchetti, J. Di Rocco, L. Lambers, J. Michael, B. Rumpe, M. Sjödin, G. Taentzer, M. Tichy, H. Vangheluwe, M. Wimmer and S. Zschaler. 'Modeling: The Heart and Soul of Engineering Smart Ecosystems'. In: *Proceedings of the 17th System Analysis and Modelling conference*. SAM 2025 - 17th System Analysis and Modelling conference. Grand Rapids, Michigan, United States, 2025, pp. 1–8. URL: https://inria.hal.science/hal-05229815 (cit. on p. 25).

[53] B. Combemale, J. Kienzle, G. Mussbacher, P. Archambault, J.-M. Bruel, L. Burgueño, B. H. C. Cheng, L. Cleophas, G. Engels, D. Foures, S. Klikovits, V. Kulkarni, J. Michael, S. Mosser, H. Sahraoui, E. Syriani and A. Wortmann. 'On the Challenges of Integrating Digital Twins'. In: *Proceedings of the 2nd International Conference on Engineering Digital Twins*. EDTconf 2025 - 2nd International Conference on Engineering Digital Twins. Grand Rapids, Michigan, United States, 2025, pp. 1–7. URL: https://inria.hal.science/hal-05221809 (cit. on p. 23).

[54] B. Combemale, P. Vicat-Blanc, A. Blouin, H. Bril El Haouzi, J.-M. Bruel, J. Deantoni, T. Duval, S. Gérard and J.-M. Jézéquel. 'Engineering Digital Twins: A Research Roadmap'. In: *Proceedings of the 2nd International Conference on Engineering Digital Twins (EDTconf 2025)*. EDTconf 2025 - 2nd International Conference on Engineering Digital Twins. Grand Rapids, Michigan, United States, 2025, pp. 1–7. URL: https://inria.hal.science/hal-05223776 (cit. on p. 24).

[55] É. Guégain, A. Bonvoisin, M. Acher, C. Quinton and R. Rouvoy. 'Exploring Performance of Configurable Software Systems: the JHipster Case Study'. In: EASE'25 - 29th International Conference on Evaluation and Assessment in Software Engineering. Istanbul, Turkey, 2025, pp. 1–10. URL: https://hal.science/hal-05003699 (cit. on p. 31).

[56] M. Huyghe, C. Quinton and W. Rudametkin. 'BrowserFM: A Feature Model-based Approach to Browser Fingerprint Analysis'. In: MADWeb'25 - Workshop on Measurements, Attacks, and Defenses for the Web. San Diego, United States, 2025, pp. 1–10. DOI: 10.14722/madweb.2025.23017. URL: https://hal.science/hal-04949268 (cit. on p. 35).

[57] M. Huyghe, C. Quinton and W. Rudametkin. 'FP-Rainbow: Fingerprint-Based Browser Configuration Identification'. In: WWW'25 - International World Wide Web Conference. Sydney, Australia: ACM, 2025, pp. 4325–4335. DOI: 10.1145/3696410.3714699. URL: https://hal.science/hal-04932081 (cit. on p. 34).

[58] S. Q. de Medeiros, R. Lefeuvre, B. Combemale and Q. Perez. 'Evaluating the Energy Profile of Tasks Managed by Build Automation Tools in Continuous Integration Workflows: The Case of Apache Maven and Gradle'. In: ICT4S 2025 - International Conference on Information and Communications Technology for Sustainability. Dublin, Ireland: IEEE, 2025, pp. 1–11. DOI: 10.1109/ICT4S68164.2025.00011. URL: https://hal.science/hal-05090865 (cit. on p. 31).

[59] M. Molli, D. Balouek, P. Temple and T. Ledoux. 'Event-Driven Adaptation in the Computing Continuum Using Software Variability'. In: *2025 IEEE/ACM 18th International Conference on Utility and Cloud Computing (UCC '25)*. UCC 2025 - IEEE/ACM 18th International Conference on Utility and Cloud Computing. Nantes, France: IEEE, 1st Dec. 2025, pp. 1–2. DOI: 10.1145/3773274.3774672. URL: https://hal.science/hal-05379151 (cit. on p. 31).

[60] S. Pélissier, N. Mehanna, S. Roux, Q. Perez, W. Rudametkin, J. Bourcier and P. Laperdrix. 'Users Pay Twice: The Hidden Energy Cost of Web Advertising'. In: WWW 2026 - ACM Web Conference. Dubai, United Arab Emirates: ACM, 2026. DOI: 10.1145/3774904.3792414. URL: https://hal.science/hal-05479340.

[61] J. Pfeiffer, J. Zhang, B. Combemale, J. Michael, B. Rumpe, M. Wimmer and A. Wortmann. 'Towards a Unifying Reference Model for Digital Twins of Cyber-Physical Systems'. In: *Proceedings of the 30th IEEE International Conference on Emerging Technologies and Factory Automation*. ETFA 2025 - 30th IEEE International Conference on Emerging Technologies and Factory Automation. Porto (Portugal), Portugal: IEEE, 2025, pp. 1–4. URL: https://inria.hal.science/hal-05224789 (cit. on p. 23).

[62] C. Reux, M. Acher, D. E. Khelladi, O. Barais and C. Quinton. 'LLM Code Customization with Visual Results: A Benchmark on TikZ'. In: *Proceedings of The 29th International Conference on Evaluation and Assessment in Software Engineering (EASE 2025)*. EASE'25 - Evaluation and Assessment in Software Engineering. Istanbul, Turkey, 2025, pp. 1–10. DOI: 10.1145/3756681.3757003. URL: https://hal.science/hal-05049250 (cit. on pp. 22, 32).

[63] R. Robbes, T. Matricon, T. Degueule, A. Hora and S. Zacchiroli. 'Promises, Perils, and (Timely) Heuristics for Mining Coding Agent Activity'. In: MSR 2026 - 23rd IEEE Working Conference on Mining Software Repositories. Rio de Jaineiro, Brazil, 2026. DOI: 10.1145/3793302.3793375. URL: https://hal.science/hal-05487636.

[64] B. Sanwouo, P. Temple and C. Quinton. 'Dynamic Agent Generation for Self-Adaptive Root Cause Analysis'. In: SEAMS'26 - 21st International Conference on Software Engineering for Adaptive and Self-Managing Systems. Rio de Janeiro, Brazil, 2026. URL: https://hal.science/hal-05402186.

[65] B. Sanwouo, P. Temple and C. Quinton. 'Generative AI-based Adaptation in Microservices Architectures: A Systematic Mapping Study'. In: ICWS'25 - International Conference on Web Services. Helsinki, Finland, 2025, pp. 1–8. URL: https://hal.science/hal-05082732 (cit. on p. 32).

[66] B. P. Sanwouo, C. Quinton and P. Temple. 'Breaking the Loop: AWARE is the new MAPE-K'. In: FSE'25 - ACM International Conference on the Foundations of Software Engineering. Trondheim, Norway: ACM, 2025, pp. 626–630. DOI: 10.1145/3696630.3728512. URL: https://hal.science/hal-04992342 (cit. on p. 34).

[67] H. Spieker, T. Matricon, N. Belmecheri, J. E. Betten, G. Le Bartz Lyan, H. Borges, Q. Mazouni, D. Gross, A. Gotlieb and M. Acher. 'Prompting for Performance: Exploring LLMs for Configuring Software'. In: *IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*. ICTAI 2025 - 37th IEEE International Conference on Tools with Artificial Intelligence. Athens, Greece, 2025. URL: https://hal.science/hal-05494832.

[68] T. E. J. Vos, T. van der Storm, A. Serebrenik, L. Briand, R. Di Cosmo, J.-M. Bruel and B. Combemale. 'Reclaiming Software Engineering as the Enabling Technology for the Digital Age'. In: ICSE 2026 - Future of Software Engineering. Rio De Janeiro, Brazil: IEEE, 2026. DOI: 10.1145/XXXXXXX.XXXXXXX. URL: https://inria.hal.science/hal-05468508.

Conferences without proceedings

[69] H. Borges, J. A. Pereira, D. E. Khelladi and M. Acher. 'Linux Kernel Configurations at Scale: A Dataset for Performance and Evolution Analysis'. In: EASE 2025 - Evaluation and Assessment in Software Engineering. Istambul, Turkey, 2025, pp. 1–10. URL: https://hal.science/hal-05063560 (cit. on pp. 22, 31).

[70] R. Lefeuvre, M. Le Goasteller, J. Galasso, B. Combemale, Q. Perez and H. Sahraoui. 'Modeling Sampling Workflows for Code Repositories'. In: 23rd International Conference on Mining Software Repositories (MSR '26). Rio de Janeiro (BRAZIL), Brazil, 13th Apr. 2026. URL: https://hal.science/hal-05478146.

[71] Y. Merel Jourdan, M. Acher and C. Maumet. 'In the Search for Truth: Navigating Variability in Neuroimaging Software Pipelines'. In: SPLC 2025 - 29th ACM International Systems and Software Product Line Conference. Coruna, Spain, Spain, 2025, pp. 129–135. DOI: 10.1145/3744915.3748470. URL: https://inria.hal.science/hal-05158426 (cit. on p. 28).

[72] C. Molinier, P. Temple, O. Zendra and O. Barais. 'Approaches for strengthening embedded AI against attacks disrupting federated learning'. In: Ressi 2025 - Rendez-Vous de la Recherche et de l'Enseignement de la Sécurité des Systèmes d'Information. Lanniron, France, 2025, pp. 1–3. URL: https://hal.science/hal-05004582 (cit. on p. 34).

[73] M. Molli, D. Balouek, P. Temple and T. Ledoux. 'Facilitating Heterogeneity Management on the Computing Continuum'. In: COMPAS 2025 - Conférence francophone d'informatique en Parallélisme, Architecture et Système. Bordeaux, France, 2025, pp. 1–8. URL: https://hal.science/hal-05138189 (cit. on p. 26).

[74] X. Tërnava, R. Lefeuvre, Q. Perez, D. E. Khelladi, M. Acher and B. Combemale. 'On the Effect of Feature Reduction on Energy Consumption: An Exploratory Study'. In: SPLC 2025 - 29th ACM International Systems and Software Product Line Conference. A Coruña, Spain, 2025, pp. 1–11. DOI: 10.1145/3744915.3748463. URL: https://hal.science/hal-05166140 (cit. on p. 30).

Scientific books

[75] M. Duranton, P. Carpenter, K. D. Bosschere, C. Gamrat, M. Gray, H. Munk, C. Robinson, T. Vardanega and O. Zendra. *HiPEAC Vision 2025*. Mar. 2025, p. 62. URL: https://inria.hal.science/hal-04989779 (cit. on p. 33).

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

[76] G. Le Guernic, ed. *Proceedings of the 32nd Computer & Electronics Security Application Rendezvous (C&ESAR 2025)*. C&ESAR 2025 - 32nd Computer & Electronics Security Application Rendezvous. Vol. 4143. Rennes, France: CEUR-WS.org, 22nd Dec. 2025. URL: https://inria.hal.science/hal-05483716.

Doctoral dissertations and habilitation theses

[77] Z. K. Kebaili. 'Supporting metamodel and code co-evolution'. Université de Rennes, 15th May 2025. URL: https://theses.hal.science/tel-05272600.

[78] G. A. Randrianaina. 'Incremental, reproducible builds of software variants'. Université de Rennes, 28th May 2025. URL: https://theses.hal.science/tel-05308121.

Reports & preprints

[79] O. Barais, R. D. Cosmo, L. Mé, S. Zacchiroli and O. Zendra. *Software Identification for Cybersecurity: Survey and Recommendations for Regulators*. Software Heritage Security project (SWHSec), 28th Mar. 2025. URL: https://hal.science/hal-05009757 (cit. on p. 33).

[80] A. Gauvain, R. Abhervé, A. Coche, M. Le Mesnil, C. Roques, C. Bouchez, J. Marçais, S. Leray, E. Marti, R. Figueroa, E. Bresciani, C. Vautier, B. Boivin, J. Sallou, J. Bourcier, B. Combemale, P. Brunner, L. Longuevergne, L. Aquilina and J.-R. de Dreuzy. *HydroModPy: A Python toolbox for deploying catchment-scale shallow groundwater models*. 8th Jan. 2025. DOI: 10.5194/egusphere-2024-3962. URL: https://hal.science/hal-05116367 (cit. on p. 25).

[81] R. Lefeuvre, C. Reux, S. Zacchiroli, O. Barais and B. Combemale. *Did You Forkget It? Detecting One-Day Vulnerabilities in Open-source ForksWith Global History Analysis*. 2025. URL: https://hal.science/hal-05349203 (cit. on p. 34).

[82] Y. Merel Jourdan, M. Acher and C. Maumet. *A systematic and large-scale exploration of analytical variability in task-fMRI*. 15th Dec. 2025. URL: https://inria.hal.science/hal-05458481 (cit. on p. 32).

[83] X. Tërnava, G. A. Randrianaina, L. Lesoil and M. Acher. *Small Yet Configurable: Unveiling Null Variability in Software*. 2025. URL: https://hal.science/hal-05097580 (cit. on p. 30).

Other scientific publications

[84] M. Huyghe, W. Rudametkin and C. Quinton. 'FP-Rainbow: Fingerprint-Based Browser Configuration Identification'. In: WWW'25 - International World Wide Web Conference. Sydney, Australia, 2025, pp. 1–1. URL: https://hal.science/hal-05029828.

[85] B. Knowles, V. L. Hanson, C. Becker, M. Berners-Lee, A. A. Chien, B. Combemale, V. Coroamă, K. de Bosschere, Y. Ding, A. Friday, B. Gamazaychikov, L. Hardman, S. Hinterholzer, M. Höjer, L. Kaack, L. Kuijer, A.-L. Ligozat, J. T. Muehlberg, Y. Nah, T. Olsson, A.-C. Orgerie, D. Pargman, B. Penzenstadler, T. Romanoff, E. Strubell, C. Venters and J. Zhao. *Climate Change: What is Computing's Responsibility?* 2025. DOI: 10.4230/DagMan.11.1.1. URL: https://hal.science/hal-05369257 (cit. on p. 26).

12.3 Cited publications

[86] 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. 13).

[87] A. Avizienis. 'The N-version approach to fault-tolerant software'. In: *Software Engineering, IEEE Transactions on* 12 (1985), pp. 1491–1501 (cit. on p. 13).

[88] 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. 11).

[89] 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. 13).

[90] 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. 10).

[91] 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. 13).

[92] 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. 11).

[93] 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. 12).

[94] 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. 12).

[95] A. Beugnard, J.-M. Jézéquel and N. Plouzeau. 'Contract Aware Components, 10 years after'. In: *WCSI*. 2010, pp. 1–11 (cit. on p. 12).

[96] 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. 10).

[97] 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. 11).

[98] 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. 13).

[99] 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. 13).

[100] 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. 12).

[101] 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. 12).

[102] 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. 10).

[103] 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. 12).

[104] 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. 12).

[105] 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. 13).

[106] 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. 10).

[107] 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. 12).

[108] 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. 11).

[109] 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. 13).

[110] 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. 13).

[111] 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. 10).

[112] 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. 11).

[113] 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. 11).

[114] J. Kramer and J. Magee. 'Self-Managed Systems: an Architectural Challenge'. In: *Future of Software Engineering*. IEEE, 2007, pp. 259–268 (cit. on p. 12).

[115] 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. 12).

[116] 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. 13).

[117] J. Meekel, T. B. Horton and C. Mellone. ‘Architecting for Domain Variability’. In: *ESPRIT ARES Workshop*. 1998, pp. 205–213 (cit. on p. 11).

[118] 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. 12).

[119] 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. 13).

[120] 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. 10).

[121] 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. 11).

[122] 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. 11).

[123] 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. 11).

[124] 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. 10).

[125] L. M. Northrop. ‘SEI’s Software Product Line Tenets’. In: *IEEE Softw.* 19.4 (2002), pp. 32–40 (cit. on p. 10).

[126] 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. 13).

[127] 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. 10).

[128] 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. 11).

[129] 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. 11).

[130] 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. 9).

[131] B. Randell. ‘System structure for software fault tolerance’. In: *Software Engineering, IEEE Transactions on* 2 (1975), pp. 220–232 (cit. on p. 13).

[132] 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. 12).

[133] 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. 13).

[134] D. Schmidt. ‘Guest Editor’s Introduction: Model-Driven Engineering’. In: *IEEE Computer* 39.2 (2006), pp. 25–31 (cit. on p. 10).

[135] F. Shull, J. Singer and D. I. Sjberg. *Guide to advanced empirical software engineering*. Springer, 2008 (cit. on p. 13).

[136] 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. 10).

[137] C. Szyperski, D. Gruntz and S. Murer. *Component software: beyond object-oriented programming*. Addison-Wesley, 2002 (cit. on p. 12).

[138] J.-C. Trigaux and P. Heymans. *Modelling variability requirements in Software Product Lines: a comparative survey*. Tech. rep. FUNDP Namur, 2003 (cit. on p. 11).

[139] M. Utting and B. Legeard. *Practical model-based testing: a tools approach*. Morgan Kaufmann, 2010 (cit. on p. 13).

[140] 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. 12).

[141] 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. 13).

[142] T. Ziadi and J.-M. Jézéquel. ‘Product Line Engineering with the UML: Deriving Products’. In: Springer Verlag, 2006, pp. 557–586 (cit. on p. 11).