JOHANN MORTARA, PH.D.

Ph.D. in Computer Science

Born and living in Monaco, MONACO 51 Boulevard du Jardin Exotique, MC 98000 Monaco
(+33) 6.45.00.12.17 johann.mortara@gmail.com https://j-mortara.github.io
0000-0002-1779-5511 https://github.com/j-mortara linkedin.com/in/johann-mortara

EDUCATION

Ph.D. in Computer Science

Université Côte d'Azur – I3S Laboratory, CNRS

October 2019 – December 2022

Title Mastering variability in the wild: On object-oriented variability implementations and variability-aware build systems
University Université Côte d'Azur
Research Unit Laboratoire d'Informatique, de Signaux et Systèmes de Sophia Antipolis (I3S), UMR 7271, CNRS
Funding Scholarship funded by the French state, awarded by the STIC Doctoral School of Université Côte d'Azur by competition.
Date obtained December 20, 2022
Ph.D. advisor Philippe COLLET, Full Professor, Université Côte d'Azur, CNRS, I3S, France

Jury members

Xavier BLANC Full Professor, Université de Bordeaux, Bordeaux INP, CNRS, LaBRI, France (reviewer)
Jean-Marc JÉZÉQUEL Full Professor, Université de Rennes 1, Inria/IRISA, France (reviewer)
Anne-Marie DERY-PINNA Associate Professor, Université Côte d'Azur, CNRS, I3S, France (examiner)
Romain ROUVOY Full Professor, Université de Lille, Inria, France (examiner)

Master's Degree in Management and Business Administration (Graduated with honours)

Université Côte d'Azur – IAE Nice

January 2019 – November 2019

Exact title Master Management et Administration des Entreprises – Direction d'Entreprise (MAE-DE)
Dissertation title The Open Innovation: a major stake for research
Advisor Inès ABID, Associate Professor, Université Côte d'Azur, CNRS, GRM, France
Description One-year Master's in partnership with Polytech Nice Sophia.

Master's Degree in Computer Science (FR: Diplôme d'Ingénieur)

Université Côte d'Azur – Polytech Nice Sophia

September 2016 – September 2019

Exact Title Diplôme d'Ingénieur en Informatique
Dissertation title Identification and visualization of variability implementations in large variability-rich software systems
Advisor Philippe COLLET, Full Professor, Université Côte d'Azur, CNRS, I3S, France

PROFESSIONAL EXPERIENCE

Teaching Assistant (DCME)

October 2019 – December 2022 Université Côte d'Azur – Polytech Nice Sophia

During my Ph.D. thesis, in parallel of my research activities, I taught classes at the Polytech Nice Sophia engineering school as part of a DCME (*Doctorant Contractuel avec Mission d'Enseignement*) contract. I taught in 3rd, 4th and 5th years

of engineering school, equivalent to Licence 3 (3rd year of B.Sc.), Master 1 (1st year of M.Sc.) and Master 2 (2nd year of M.Sc.) years. My teaching activities are detailed in the "Teaching experience" section.

Identification and visualization of variability implementations in large variability-rich software systems

5th year internship — **Advisor : Prof. Philippe COLLET**

March 2019 – September 2019 — Université Côte d'Azur – I3S, CNRS

Design and development of *symfinder*, a prototype identifying and visualizing variability implementations in *Java* software projects.

This internship led to two publications at SPLC '19 conference [15, 16], obtaining the **ACM Artifact Reusable badge** assessing reusability of the tool.

Blockchain deployment and sensors network simulation

4th year internship — **Advisor : Prof. Sébastien MOSSER**

July 2018 – August 2018 — Université Côte d'Azur – I3S, CNRS

This internship is part of the I-WIN (*Intelligent Wireless IoT Networks*) project of the IDEX UCA^{Jedi} program and aims to design a solution allowing to deploy an *Ethereum* blockchain to store data from a sensors network and trigger devices in a *Smart City* context.

- Tackling problems linked to *blockchains* and *smart cities* such as data sanctity and the propagation time of information;
- Development of a simulation prototype using *Docker* and physical deployment on *Raspberry Pi* boards.

TEACHING EXPERIENCE

I taught at the Polytech Nice Sophia engineering school (Université Côte d'Azur) in the Computer Science department. The students integrating this programme have diverse backgrounds such as *classe préparatoire* (intensive two-year study course preparing for the competitive entrance examinations to the French "Grandes Écoles", the top French and highly-selective institutions) or BUT (*Bachelor Universitaire de Technologie*, previously called DUT, being a two-year programme where students acquire mainly technical skills). I taught on the three years of this programme, corresponding to the French Licence 3, Master 1 and Master 2 years (*i.e.*, third year of B.Sc. and two years of M.Sc.).

EIIN525 – 5th semester project: introduction to software development / EIIN525B – *Projet trinôme* (level: L3, second year of B.Sc.)

In this course, the students learn the basics of software engineering and are introduced to the tools and development practices used in the industry (*e.g.*, Agile methodologies, use of a version control system and a dependency manager, unit tests). The different concepts and tools are presented during the lectures and progressively incorporated during the labs in a project extending over the duration of the course that the students carry out in teams of 3–4. In this course, I had the responsibility to follow the students each week during the lab session in the evolution of their project. During these sessions, the follow-up of the students consisted in an individual discussion time with each team of students to make sure that they assimilated the concepts presented in the lecture and that they correctly apply these concepts in their project. I also participated in the evaluation of the different deliverables (code, reports).

EIIN716 – Software design (level: M1, first year of M.Sc.)

In this course, the students learn to design software systems using UML (*Unified Modeling Language*). The different concepts and tools are presented during the lectures and progressively incorporated during the labs in a project extending over the duration of the course that the students carry out in teams of 3–4. In this course, I had the responsibility to follow the students each week during the lab session in the evolution of their project. During these sessions, the follow-up of the students consisted in an individual discussion time with each team of students to make sure that they assimilated the concepts presented in the lecture and that they correctly apply these concepts in their project. I also participated in the evaluation of the different deliverables (code, reports).

EIINA904 – Reverse-Engineering, Maintenance and Software Evolution (level: M2, second year of M.Sc.)

In this course, students tackle research questions related to code reverse-engineering problems using a scientific approach. At the beginning of the course, the teachers propose several research questions. The students, in teams of 4–5, choose one of them and then propose an approach to solve it that they design and realize throughout the course. The lectures consist of several presentations by academics and industrials facing reverse-engineering problems in their work. During the labs, the students work on their question and develop the necessary tools to answer it. In this course, I was responsible for (*i*) proposing research questions related to the research topic of my Ph.D., (*ii*) following the students who chose these questions and evaluating their different deliverables (code, reports) and (*iii*) presenting a 1 to 2 hour lecture on the application of the scientific approach in the context of my Ph.D.

Other teaching duties

I also participated in oral evaluations in the **EIIN717B – 7th semester project (level: M1, first year of M.Sc.)** course at Polytech Nice Sophia, during which students work on a project during three full-time weeks allowing them to discover the different specializations proposed in the last year. I was also invited in the **M3301 – Software production methodology (level: L2, second year of B.Sc.)** course at the IUT Nice Côte d'Azur where I presented an introductory lecture on Continuous Integration. In this course, students are introduced to the tools and methods allowing to determine the specifications, technically design and implement a software solution.

All the lectures I gave are available on my website. My teaching duties are summarized in the table hereafter.

Course	Year	Taught hours	Responsibilities
EIIN525 – 5th semester project: introduction to software development / EIIN525B – *Projet trinôme* (L3, 3rd year of B.Sc.) Nature: Projects Given at: Polytech Nice Sophia Programme: Computer Science Engineer	2019 – 2020 2020 – 2021 2021 – 2022	52.75h labs 42.50h labs 64.75h labs	Supervising students, following their project's evolution and reexplaining the notions, designing subjects, evaluating deliverables
EIIN716 – Software design (M1, 1st year of M.Sc.) Nature: Projects Given at: Polytech Nice Sophia Programme: Computer Science Engineer	2019 – 2020 2021 – 2022	33.50h labs 36.50h labs	Supervising students, following their project's evolution and reexplaining the notions, designing subjects, evaluating deliverables
EIINA904 – Reverse-Engineering, Maintenance and Software Evolution (M2, 2nd year of M.Sc.) Nature: Projects Given at: Polytech Nice Sophia Programme: Computer Science Engineer	2019 – 2020 2020 – 2021 2021 – 2022 2022 – 2023	10h labs, 1h lecture 10h labs, 1.50h lecture 14.50h labs, 1.50h lecture 4h labs, 2h lecture	Supervising students, following their project's evolution and reexplaining the notions, designing subjects, evaluating deliverables, invited lecture
EIIN717B – 7th semester project (M1, 1st year of M.Sc.) Nature: Projects Given at: Polytech Nice Sophia Programme: Computer Science Engineer	2020 – 2021	3.75h TD	Evaluating oral presentations
M3301 – Software production methodology (L2, 2nd year of B.Sc.) Nature: Projects Given at: IUT Nice Côte d'Azur Programme: Computer Science DUT	2019 – 2020 2020 – 2021	1h CM, 4h TD 1h CM	Invited lecture and associated lab Invited lecture

RESEARCH ACTIVITIES

Research topics

My research is focused in the field of software engineering. I am particularly interested in how large configurable software systems are built and the challenges related to their understanding, maintenance and evolution.

The ever-increasing demand for new and up-to-date software solutions requires software practitioners to develop and maintain customizable software systems while ensuring a high level of quality and reliability. Software Product Lines (SPLs) provide a solution to achieve this goal by providing a formal framework for controlling the system's *variability*. In this context, variability is documented with an exhaustive list of implemented features and an explicit link between each feature and its implementation. However, as the implementation of this formalism is very cumbersome, its use is limited in practice and many systems are not architected as such. They progressively increase their variable parts, relying on the multiple existing mechanisms to implement their variability in the code and their build system. By adapting existing tools to implement variability, these systems do not have the ability to check the consistency of these implementations and anomalies can occur when resolving this variability. Finally, when these systems implement their variability without dedicated mechanisms (such as object-oriented (OO) systems that reuse OO mechanisms for this purpose), it is buried in the code and difficult to understand. These properties are factors of technical debt and consequently hamper the maintenance of the system and its evolution. To prevent this technical debt, there is therefore a need to identify and understand the variability implemented in such systems. In my work, I have tackled this problem on two axes.

1) Comprehending the variability implemented in OO software systems

When implementing their variability in a single codebase, most OO systems do not document it and implement it by reusing traditional OO mechanisms for organizing code (such as inheritance, method and constructor overloading, and certain design patterns). Because these mechanisms are similar in nature, such implementations of variability are buried in the code. As the system evolves, the trace of these implementations is lost, hindering their evolution and maintenance, hampering the quality of the system. There is therefore a need to identify these variability implementations. On large systems, such identification will lead to a large volume of metrics that must be made understandable. Finally, since these implementations can lead to technical debt, we must also be able to measure their quality.

I started working on these topics during a research internship in my last year of engineering school at the I3S laboratory (Université Côte d'Azur, CNRS) under the supervision of Prof. Philippe COLLET. In collaboration with Dr. Xhevahire TËRNAVA, postdoctoral researcher in the DiverSE team at Inria/IRISA Rennes, who proposed an identification technique based on the presence of a density of symmetries in OO structures, I developed the *symfinder* toolchain allowing to automate the application of this technique on large codebases written in Java and which proposes a visualization in the form of a graph of the identified implementations of variability. The application of the approach on eight open source systems allowed the visual identification of areas where relevant variability implementations are concentrated. The results obtained were the subject of two publications [15, 16] at the SPLC 2019 international conference (CORE B), a reference conference in the software product line domain, and constitute the basis of my thesis work on this axis. I also presented *symfinder* and the approach it demonstrates in a tutorial [8] at the SPLC 2021 international conference (CORE B).

This first evaluation of the relevance of the approach highlights several limitations. First of all, the technique has only been applied on Java systems. Second, the validation of the relevance of the identification result is only visual. Finally, the comprehensibility of the visualization has not been evaluated.

My thesis work is part of an empirical approach. Models are proposed from observations in existing systems and are implemented in reusable prototypes allowing their application and validation on large open source systems. The results obtained allow to refine the research questions and to increment the proposed contributions.

The first steps of my doctoral work consisted in leveraging the limitations of the evaluation of the *symfinder* approach. First, the approach was extended to the C++ language by applying it to several open source systems written in this language. The results obtained, similar in nature to those obtained for Java systems, validate the relevance of this approach to this other OO language. These results were published at the SPLC 2020 (CORE B) conference [13]. Secondly, the relevance of the proposed visualization was validated by an experiment with Daniel Le Berre, who used *symfinder* on the Sat4j system of which he is the architect. Thanks to the visualization, he was able to find the variability he had implemented. The results obtained have been published in the AUSE journal (Q2 SCImagoJR) [1]. Third, the relevance of the identified implementations was validated by comparing for two systems (ArgoUML and Sat4j) the identified implementations with traces indicating in the code the implemented features. The obtained results, published at the VaMoS 2020 international conference [14] and in the AUSE journal [1], show that although the approach is able to identify the majority of documented variability implementations, it also identifies a large number of false positives.

In order to increase the accuracy of our identification approach, we extended it by taking into account the usage relationships (*e.g.*, aggregation, composition) as mechanisms involved in the creation of dense zones of variability implementations and by formalizing this notion of density of variability implementations allowing, thanks to thresholds, to automatically

identify *hotspots* being classes that are part of zones dense in variability implementations. The new *symfinder-2* experimental toolchain proposes an improved visualization including usage relationships and allowing, thanks to several options, to configure the view to be able to focus on a subpart of the implementation and to progressively explore a large system. The application of the toolchain on a set of open source systems has shown that the *hotspots* allow to obtain a smaller and more relevant set of variability implementations. The obtained results have been published at the REVE 2021 workshop [11], co-located with SPLC 2021.

Although the new customization capabilities of the visualization make it possible to focus on one part of the implementation, the view becomes difficult to understand when the set of visualized classes grows. Many visualizations designed to represent metrics and properties of software systems (such as architecture or quality) employ metaphors to make them understandable. For example, several visualizations for visualizing quality metrics of an OO system such as CodeCity or Evo-Streets represent the system as a city. We therefore proposed *VariCity*, an interactive 3D visualization adapting the city metaphor to represent an OO system and reveal areas dense with variability implementations. Variability understanding activities being one of the steps of an onboarding process, where a person will discover a codebase that is new to them and will have to quickly understand its main characteristics, we validated *VariCity* by unfolding several onboarding scenarios, showing the capabilities of the visualization to highlight the areas of the system where variability implementations are concentrated. We also performed a controlled experiment with a panel of students in last year of M.Sc. in Computer Science, showing the contribution of *VariCity* in the resolution of variability understanding tasks. The results obtained have been published at the VISSOFT 2021 international conference (CORE B) [9], the reference conference in software visualization, except for the experiment in controlled environment which will be submitted in a special issue of the JSS (Journal of Systems and Software) journal (Q1 SCImagoJR, SJR 2021 1.42).

While it allows to understand the implemented variability, *VariCity* does not allow for the measurement and understanding of the quality of these implementations, leaving the challenge regarding the quality of variability implementations unaddressed. Understanding the quality of an OO system is often achieved through the use of visualizations, such as CodeCity or Evo-Streets mentioned above, which rely on the city metaphor, but do not provide information about the implemented variability. To address this challenge, *VariMetrics* has been proposed. This new visualization extends *VariCity* to simultaneously visualize information about the implemented variability and the quality of the system by proposing additional visualization axes (colors, textures) and thus revealing the critical classes concentrating variability implementations. The application of the visualization on several open source systems shows its ability to highlight the areas of the system being both critical and dense in variability implementations. A refactoring of several of these areas on a system has allowed an improvement of the quality of the impacted classes and of the system as a whole. The results have been published at the SPLC 2022 international conference (CORE B) [2]. *VariMetrics* has also been integrated into a development environment in order to limit context switching on the part of a developer using the visualization and to enable bidirectional navigation between the code and its visual representation. This extension has also been published at the SPLC 2022 international conference [3].

2) Comprehending the variability managed by build systems

In order to speed up their development, most large software systems reuse tools that they adapt to implement their variability (*e.g.*, the C preprocessor or Makefiles). Since these tools are not designed to handle variability, they do not have the ability to check the consistency of the variability they implement, potentially leading to anomalies, *i.e.*, places in the code where the realized variability differs from the implemented variability (*e.g.*, code that can never be selected). Furthermore, when a system relies on several of these mechanisms, a build system will be in charge of resolving the variability in several steps, each selecting parts of the code thanks to conditions on features. Since each mechanism is only aware of the variability it implements, no global view of the variability is available and conflicts can arise between these steps, also causing anomalies. In order to avoid these anomalies, there is therefore a need to provide an overview of the variability implemented in the different stages in order to characterize and identify the different anomalies that may occur.

State-of-the-art contributions dealing with anomalies in build systems are focused on the Linux kernel and present several limitations. Definitions of anomalies with similar names diverge, while the formalisms proposed for their identification are often coarse-grained and do not allow for a fine-grained view of the conditions that allow a part of the code to be selected. These limitations hinder not only the understanding of these contributions but also their application to other build systems. We therefore proposed a model abstracting a software build system based on two concepts: a *configurator* generating a configuration which is then consumed by one or more *derivators* which will define conditions allowing to select parts of the implementation. The model allows to describe precisely for each part of the code the set of conditions involved in its selection in the whole build system. Finally, the anomalies are characterized with this formalism. The relevance of our representation has been evaluated by instantiating all 25 definitions proposed in the state of the art, showing the coverage of existing work and revealing the inconsistencies between them. These results have been published at the SPLC 2021 international conference [6].

Although showing the coverage of existing work on the Linux kernel, the evaluation conducted of the proposed model needs to be extended to validate its relevance to represent build systems. We have therefore studied the build system of Mozilla Firefox, which is similar in nature to the Linux kernel's, and whose diversity of mechanisms led to an extension of the model. Moreover, in practice, an anomaly identification mechanism must be able to explain precisely its origin. We therefore propose a representation of anomalies in the form of an object metamodel allowing, after instantiation, to represent all the selectable parts of the code. These contributions have been empirically evaluated on the Linux kernel

and Mozilla Firefox thanks to a toolchain implementing our model allowing the identification of anomalies and their instantiation in the proposed metamodel. The anomalies identified in the Linux kernel are comparable to those identified by another state-of-the-art approach. Concerning Mozilla Firefox, anomalies could be identified in and between each step of the build system and a detailed report of the anomalies and their origins was submitted to the Mozilla Foundation developers. This work will be submitted to the SoSyM (Software and Systems Modeling) journal (Q1 SCImagoJR, SJR 2021 0.83) in the coming weeks.

I also got interested, in collaboration with other Ph.D. students of my laboratory, in the application of feature identification techniques to determine the features implemented by existing functions and thus propose a catalog of reusable functions accessible on the semantic web. This work led to the publication of a vision paper entitled *Towards a Linked Open Code* [5] at the ESWC 2021 international conference (CORE A), a reference conference of the semantic web community.

Service

Participation to program committees

2020 **Systems and Software Product Line Conference – SPLC '20 (CORE B)**, Artifact Track committee member

Participation to editorial committees

- **IEEE Transactions on Software Engineering – TSE (Q1 SCImagoJR, SJR 2021 2.03)**, reviewer

Mentoring

I co-supervised several internships and final year projects related to the topic of my Ph.D.. The work done by these students consists mainly in the development of prototypes allowing to validate empirically the approaches presented in my thesis. I participated, with the other co-supervisors, in the definition of the subjects and in the regular follow-up of the students by accompanying them in their technological choices and by defining the realization stages of the prototypes. I also participated in the transmission of the scientific and technical knowledge of the prototypes. Finally, I ensured the integration and validation of the prototypes, both regarding their reusability and the relevance and reproducibility of the scientific results they allow to obtain. These properties are then evaluated by the program committee members of the conferences who award badges attesting to the reproducibility and reusability of the tools, such as those obtained by *symfinder* [15], *VariCity* [9] and *VariMetrics* [2].

2021 – 2022 **Yann Brault** – *Automatic Identification of variability implementations in JavaScript*, M.Sc. 4th year research project, Faculté des Sciences de Nice.

Martin Bruel – *SymfinderJS: Automatic Identification of Variability Implementations in JavaScript-like Functional Languages*, M.Sc. (*ingénieur*) final-year project, Polytech Nice Sophia.

Patrick Anagonou, Guillaume Savornin, Anton Van der Tuijn – *Metrics exentensions and configuration for VariCity* [4], M.Sc. (*ingénieur*) final-year project, Polytech Nice Sophia.

João Brilhante, Charly Ducrocq, Ludovic Marti – *Intégration VariCity - IDE*, M.Sc. (*ingénieur*) final-year project, Polytech Nice Sophia.

2020 – 2021 **Paul-Marie Djekinnou, Florian Focas, François Rigaut** – *VariCity : une visualisation sous forme de ville pour comprendre la variabilité du code* [10], M.Sc. (*ingénieur*) final-year project, Polytech Nice Sophia.

2019 – 2020 **Florian Ainadou, Paul-Marie Djekinnou, Djotiham Nabagou** – *symfinder-API* [12], M.Sc. (*ingénieur*) final-year internship, Polytech Nice Sophia.

Théo Foray, Grégoire Peltier, Nathan Strobbe – *Identification et visualisation d'implémentations de variabilité dans des bases de code en C++ et JavaScript*, M.Sc. (*ingénieur*) final-year project, Polytech Nice Sophia.

PUBLICATIONS

International peer-reviewed journals

[1] Xhevahire Tërnava, **Johann Mortara**, Philippe Collet, and Daniel Le Berre. “Identification and visualization of variability implementations in object-oriented variability-rich systems: a symmetry-based approach”. In: *Journal of Automated Software Engineering* 29 (Feb. 2022), pp. 1–51. DOI: 10.1007/s10515-022-00329-x. URL: https://hal.archives-ouvertes.fr/hal-03593967. CORE B, Q1 SCImagoJR, SJR 2021 0.94.

International peer-reviewed conferences – Full papers

[2] **Johann Mortara**, Philippe Collet, and Anne-Marie Dery-Pinna. “Customizable Visualization of Quality Metrics for Object-Oriented Variability Implementations”. In: *Proceedings of the 26th ACM International Systems and Software Product Line Conference - Volume A*. SPLC ’22. Graz, Austria: Association for Computing Machinery, Sept. 2022, pp. 43–54. ISBN: 978-1-4503-9443-7. DOI: 10.1145/3546932.3547073. URL: https://hal.archives-ouvertes.fr/hal-03717858. CORE B. **Obtained the ACM *Artifacts Evaluated - Reusable* badges.**

[6] **Johann Mortara** and Philippe Collet. “Capturing the diversity of analyses on the Linux kernel variability”. In: *Proceedings of the 25th ACM International Systems and Software Product Line Conference - Volume A*. SPLC ’21. Leicester, United Kingdom: Association for Computing Machinery, Sept. 2021, pp. 160–171. ISBN: 978-1-4503-8469-8. DOI: 10.1145/3461001.3471151. URL: https://hal.archives-ouvertes.fr/hal-03283627. CORE B. *Companion technical report: 10.5281/zenodo.4715969*.

[9] **Johann Mortara**, Philippe Collet, and Anne-Marie Dery-Pinna. “Visualization of Object-Oriented Variability Implementations as Cities”. In: *2021 Working Conference on Software Visualization (VISSOFT)*. Luxembourg (virtual), Luxembourg, Sept. 2021, pp. 76–87. ISBN: 978-1-6654-3144-6. DOI: 10.1109/VISSOFT52517.2021.00017. URL: https://hal.archives-ouvertes.fr/hal-03312487. CORE B. **Obtained the Best Artifact Award – Open Research Objects (ORO) and Research Objects Reviewed (ROR) badges**.

[14] **Johann Mortara**, Xhevahire Tërnava, and Philippe Collet. “Mapping Features to Automatically Identified Object-Oriented Variability Implementations: The Case of ArgoUML-SPL”. In: *Proceedings of the 14th International Working Conference on Variability Modelling of Software-Intensive Systems*. VaMoS ’20. Magdeburg, Germany: Association for Computing Machinery, Feb. 2020, pp. 1–9. ISBN: 978-1-4503-7501-6. DOI: 10.1145/3377024.3377037. URL: https://hal.archives-ouvertes.fr/hal-02421353. Acceptance rate: 40%.

[15] Xhevahire Tërnava, **Johann Mortara**, and Philippe Collet. “Identifying and Visualizing Variability in Object-Oriented Variability-Rich Systems”. In: *Proceedings of the 23rd International Systems and Software Product Line Conference - Volume A*. SPLC ’19. Paris, France: Association for Computing Machinery, Sept. 2019, pp. 231–243. ISBN: 978-1-4503-7138-4. DOI: 10.1145/3336294.3336311. URL: https://hal.archives-ouvertes.fr/hal-02339296. CORE B. **Obtained the ACM *Artifacts Evaluated - Reusable* badges.**

International peer-reviewed conferences – Short papers

[5] Ahmed El Amine Djebri, Antonia Ettorre, and **Johann Mortara**. “Towards a Linked Open Code”. In: *ESWC 2021 – The Semantic Web*. Ed. by Springer. Vol. 12731. ESWC: European Semantic Web Conference. Heraklion / Virtual, Greece, June 2021, pp. 497–505. ISBN: 978-3-030-77385-4. DOI: 10.1007/978-3-030-77385-4_29. URL: https://hal.archives-ouvertes.fr/hal-03190617. CORE A.

International peer-reviewed conferences – Tool demonstrations and Tutorials

[3] **Johann Mortara**, Philippe Collet, and Anne-Marie Dery-Pinna. “IDE-assisted visualization of indebted OO variability implementations”. In: *Proceedings of the 26th ACM International Systems and Software Product Line Conference - Volume B*. SPLC ’22. Graz, Austria: Association for Computing Machinery, Sept. 2022, pp. 74–77. ISBN: 978-1-4503-9206-8. DOI: 10.1145/3503229.3547066. URL: https://hal.archives-ouvertes.fr/hal-03717874. CORE B.

[8] **Johann Mortara** and Philippe Collet. “How I Met Your Implemented Variability: Identification in Object-Oriented Systems with symfinder”. In: *Proceedings of the 25th ACM International Systems and Software Product Line Conference - Volume A*. SPLC ’21. Leicester, United Kingdom: Association for Computing Machinery, Sept. 2021, p. 208. ISBN: 978-1-4503-8469-8. DOI: 10.1145/3461001.3472733. URL: https://hal.archives-ouvertes.fr/hal-03274636. CORE B.

[13] **Johann Mortara**, Philippe Collet, and Xhevahire Tërnava. "Identifying and Mapping Implemented Variabilities in Java and C++ Systems using symfinder". In: *Proceedings of the 24th ACM International Systems and Software Product Line Conference - Volume B*. SPLC '20. Montreal, QC, Canada: Association for Computing Machinery, Oct. 2020, pp. 9–12. ISBN: 978-1-4503-7570-2. DOI: 10.1145/3382026.3431251. URL: https://hal.archives-ouvertes.fr/hal-02908531. CORE B.

[16] **Johann Mortara**, Xhevahire Tërnava, and Philippe Collet. "symfinder: A Toolchain for the Identification and Visualization of Object-Oriented Variability Implementations". In: *Proceedings of the 23rd International Systems and Software Product Line Conference - Volume B*. SPLC '19. Paris, France: Association for Computing Machinery, Sept. 2019, pp. 5–8. ISBN: 978-1-4503-6668-7. DOI: 10.1145/3307630.3342394. URL: https://hal.archives-ouvertes.fr/hal-02342730. CORE B.

International peer-reviewed workshops – Full papers

[11] **Johann Mortara**, Xhevahire Tërnava, Philippe Collet, and Anne-Marie Dery-Pinna. "Extending the Identification of Object-Oriented Variability Implementations using Usage Relationships". In: *Proceedings of the 25th ACM International Systems and Software Product Line Conference - Volume B*. SPLC '21. Leicester, United Kingdom: Association for Computing Machinery, Sept. 2021, pp. 91–98. ISBN: 978-1-4503-8470-4. DOI: 10.1145/3461002.3473943. URL: https://hal.archives-ouvertes.fr/hal-03284626.

Software

[4] **Johann Mortara**, Philippe Collet, Anne-Marie Pinna-Dery, Patrick Anagonou, Guillaume Savornin, and Anton van der Tuijn. *Customizable Visualization of Quality Metrics for Object-Oriented Variability Implementations - Artifact*. June 2022. DOI: 10.5281/zenodo.6644634. URL: https://doi.org/10.5281/zenodo.6644634.

[10] **Johann Mortara**, Philippe Collet, Anne-Marie Dery-Pinna, Paul-Marie Djekinnou, Florian Focas, and François Rigaut. *Visualization of Object-Oriented Variability Implementations as Cities — Reproduction package*. **Core of the artifact submitted to the joint Artifact Track of the ICSME 2021 conference and that obtained the Best Artifact Award.** June 2021. DOI: 10.5281/zenodo.5198754. URL: https://doi.org/10.5281/zenodo.5198754.

[12] **Johann Mortara**, Xhevahire Tërnava, Philippe Collet, Anne-Marie Pinna-Dery, Florian Ainadou, Paul-Marie Djekinnou, and Djotiham Nabagou. *Extending the Identification of Object-Oriented Variability Implementations using Usage Relationships — Reproduction Package*. June 2021. DOI: 10.5281/zenodo.4946730. URL: https://doi.org/10.5281/zenodo.4946730.

AWARDS

Prix d'Excellence

Université Côte d'Azur

2021

Distinction awarded by Université Côte d'Azur for VariCity and its artifact that obtained the Best Artifact Award at ICSME 2021. This distinction has been awarded to 71 researchers and students being rewarded with prestigious prizes for their scientific contribution to the highest national and international level.

Best Artifact Award

ICSME / VISSOFT '21 conferences

2021

VariCity has been submitted as an Artifact in the Joint Artifact Evaluation Track and ROSE Festival at ICSME 2021 (evaluating artifacts from ICSME 2021, SCAM 2021, and VISSOFT 2021). These tools are reviewed to assess their public access, their capacity to easily reproduce the presented results, and their reusability in other contexts. VariCity obtained the Best Artifact Award. The CNRS also wrote an article about VariCity (in French): https://www.ins2i.cnrs.fr/fr/cnrsinfo/naviguer-dans-une-ville-pour-mieux-comprendre-les-grands-codes-variables

Prix d'Excellence

Polytech Nice Sophia

2019

Distinction awarded by Polytech Nice Sophia for representing the school regularly in multiple national and international programming contest such as the **SWERC 2018**, european qualification round for the ACM ICPC (International Collegiate Programming Contest). This was the first participation of Polytech Nice Sophia in an international programming contest. Ranked 51/89 participating teams.

PRESENTATIONS

I presented my published work in multiple international conferences and French national working groups. All the slides from these presentations are available on my personal website.

Conference presentations

2022 **Customizable Visualization of Quality Metrics for Object-Oriented Variability Implementations**, SPLC Research Track, Graz (Austria)

IDE-assisted visualization of indebted OO variability implementations, SPLC Tool Demo Track, Graz (Austria)

On Variability Debt in Object-Oriented Implementations, invited presentation at the TD4ViS workshop @ SPLC, Graz (Austria)

2021 **Visualization of Object-Oriented Variability Implementations as Cities**, VISSOFT Technical Track, Online

VariCity: Visualizing Object-Oriented Variability Implementations as Cities, VISSOFT Tool Demo Track, Online

You're an artifact, VariCity., invited presentation at the ROSE Festival @ ICSME, Online

Capturing the diversity of analyses on the Linux kernel variability, SPLC Research Track, Online

How I Met Your Implemented Variability: Identification in Object-Oriented Systems with symfinder, SPLC Tutorial, Online

2020 **Identifying and Mapping Implemented Variabilities in Java and C++ Systems using symfinder**, SPLC Tool Demo Track, Online

Mapping Features to Automatically Identified Object-Oriented Variability Implementations: The Case of ArgoUML-SPL, VaMoS, Magdeburg (Germany)

2019 **symfinder: A Toolchain for the Identification and Visualization of Object-Oriented Variability Implementations**, SPLC Tool Demo Track, Paris (France)

Working group presentations

2021 **Visualization of Object-Oriented Variability Implementations as Cities**, Groupe de Travail Génie Logiciel – IHM (*Software Engineering – Human Computer Interaction working group*), Online

Capturing the diversity of analyses on the Linux kernel variability, Groupe de Travail Vélocité Logicielle (*Software Velocity working group*), Paris (France)

2020 **Automatic identification of object-oriented variability implementations**, Groupe de Travail Vélocité Logicielle (*Software Velocity working group*), Online