Scalable Software Composition

My overall research goal is to properly understand "software composition" (*i.e.*, the opposite side of the "separation of concerns" coin), and how practitioners use it. Since the defense of my PhD in 2010, I obtained results dedicated to software composition, as well as its application to a given application domain, namely cyber-physical systems. I also contributed to software engineering in general, with a particular focus on modelling, and variability management at a large scale.

In this research statement, I will first summarize my previous research results in the last six years related to software composition (Section I), and then describe a research proposal that leverages these results with an objective of scalability (Section II). Finally, Section III describes how this proposal can integrate within the Department of Computing and Software at McMaster University. Publications referred as [K_i] (keynote) [J_i] (journal), [C_i] (conference) and [W_i] (workshop) refers to personal publications listed in my resume.

I. Contributions to the understanding of Software Composition (2014-2020)

My contributions fall into two categories: *(i)* research work done with HQPs and *(ii)* external collaborations. These collaborations can take several forms, such as dedicated work on a technical contribution (*e.g.*, collaboration with industrial partners) or wider vision papers (*e.g.*, the definition of a *Book of Knowledge* dedicated to Model-Driven Engineering). I firmly believe that research in software engineering must be applied to concrete problems. As a consequence, the work described in this section and done in collaboration with DataThing [J11, C28] is now implemented and deployed in the GreyCat database engine, commercialized by the company (with customers such as *Electricité du Luxembourg* and *Métropole de Lyon*). The domain-specific language designed for underwater floating devices [J7, C23] is now used by geoscientists from the GeoAzur research lab to design experimental campaigns.

A. Software composition general mechanisms

I focused my efforts on defining composition operators while measuring the benefits of introducing software composition in various application domains (*e.g.*, graph databases [C28], kernel development [J12], micro-services deployment [C18, C21, C26], business processes [C34]). These works have led to two invited presentations: a technical keynote presentation (250 attendees, [K1]), and an invitation by the French "*formal methods*" community to present the associated challenges [K2].

Where classical software composition approaches rely on a total ordering, I proposed approaches that free the software developer from identifying an application order when composing artifacts, relying on algebraic properties such as commutativity. Consider, for example, the Linux kernel: it contains 35 rewriting rules used to patch it and repair standard programming errors automatically. The application order is important here, and we demonstrated in B. Benni*'s Ph.D. [J12] that among the *35!* available orders several ones lead to situations where the kernel is not appropriately rewritten. We proposed a tooled approach that allows developers to apply the rules in parallel and identify conflicting situations. We applied the same model to code rewriters used to fix energy consumption anti-patterns in the Android ecosystem, and we identified issues that had not been detected before. We also performed an empirical validation of the approach to assess (with success) its scalability.

I started a collaboration with the DataThing company in Luxembourg [C28]. They develop a temporal middleware used in the context of ultra-large-scale sensors networks. In this context, I formalized the composition operator used to reconcile concurrent timelines, focusing on its commutativity to ensure determinism. The challenge here was to provide *(i)* a formal demonstration of the operator's commutativity and *(ii)* an implementation that scales to the number of operations required by the company. We multiplied by 20 the number of processable operations per second for the same deployment context (>20K ops/s, where state-of-the-art non-commutative approaches required a reconciliation bottleneck that does not scale to more than 1K ops/s). This contribution is now integrated inside the commercialized database engine (named GreyCat), as well as composable predictive models used to reduce the amount of collected data [J11].

In the micro-service deployment ecosystem, we analyzed with B. Benni* the Docker (*de facto* industrial standard for service deployment) composition model. We leveraged conflict detection mechanisms to exploit this reified composition [C26], and we analyzed a corpus of >20K deployment descriptors available on GitHub. This empirical analysis allowed us to identify that 75% of the deployment descriptors available as open source contain defects and that these defects cannot be detected without considering the compositional dimension of Docker.

B. Application domain: *Cyber-Physical Systems* (CPSs)

In this domain, I focused my effort on the definition of model-based composition operators dedicated to data collection policies and large-scale sensor networks [C31]. These operators allow one to reuse an existing network by deploying new applications on top of it, instead of building a new sensor network from scratch. This is very important in the context of Smart Cities, for example. Following this work, I was invited to give a keynote at a national event organized by the *Centre National de la Recherche Scientifique* (CNRS, France) to describe the research challenges crosscutting software engineering and CPSs [K2].

During C. Cecchinel*'s Ph.D. [C29], we supported the automatic deployment of multiple data collection policies on the same network, sharing resources to support constraints such as minimizing the energy consumption or reducing the consumed bandwidth. We also used machine learning algorithms to predict the required collection frequencies for the involved sensors, considering this data to extend applications' lifetime: a policy with a weekly scale lifetime was extended to a yearly scale one, thanks to an adaptive model trained on top of the composed policies [J11]. I also led the development of an experimental platform related to sensor networks: SmartCampus (as part of C. Cecchinel* MSc) [W53]. In I. Logre*'s Ph.D., we explored conflict detection mechanisms in a heterogeneous context [C33], using composition mechanisms to ensure the consistency of sensor-based dashboards by composing the different languages involved in such a development. In S. Lazreg*'s PhD, we worked with a car manufacturer (Visteon) to compose together *(i)* models reifying business features (what to display on the dashboard) and *(ii)* GPU pipelines able to realize such features. Using *Feature Transition Systems*, we automate the exploration of the design space to tame the variability of such realizations [C22]. This approach received a best paper in 2018 [C24], extended into a journal publication [J14]. In collaboration with the Geoscience Institute at *Université Côte d'Azur* and a company manufacturing underwater devices, in S. Bonnieux*'s PhD we explored the design of a domain-specific language that allows geoscientists to express experimental campaigns to be deployed on the devices (*e.g.*, whales' surveillance, earthquakes detection) [J7, C23].

C. Software Modelling and Variability Management

Considering software modelling, I applied *Model-Driven Engineering* (MDE) approaches to variability management in various domains such as modelling languages [W57, J15], business processes [J16, C34] or continuous integration [C18,C21,W48, W52]. Concerning continuous integration, the material designed for [C27, C48] is now used as a reference case study by three universities. My other contributions to variability management were focused on the management of large-scale software product lines [C32, W56, W58, C60], and lead to the registration of a French patent.

I have also dedicated efforts in my research contributions to work on the teaching of software modelling and variability management. This effort led to several communications in educator symposiums [W49,W50,W51] related to top conferences in the associated domains (e.g., MODELS, SPLC), as well as invited presentations in different universities [K3, K4] and industrial events [K1]. Since 2018, I am also involved in an initiative targeting the improvement of MDE teaching [J13, W48].

II. Research Proposal: Towards Scalable Composition Models (2021 – 2026)

This section describes how I plan to leverage my previous results to change the scale of systems that can be addressed by composition models. The key point here is to consider the developer as a first-class citizen and articulate the "scalability" of software composition in a developer-centric approach.

A. Context: Engineering Software at Large Scale

In a 2006 US Department of Defense report, Northrop *et al.* [1] coined the term "*Ultra-Large-Scale Systems*" (ULSSs) as systems with "*[…] ultra-large size on any imaginable dimension—number of lines of code; number of people employing the system for different purposes; amount of data stored, accessed, manipulated, and refined; number of connections and interdependencies among software components; number of hardware elements; etc.*". To tame their intrinsic complexity, the design, implementation and maintenance of ULSSs rely on decomposition approaches (*Separation of Concerns* – SoC [2]) and subsequent composition approaches to assemble the decomposed elements back together. To illustrate the design, implementation and maintenance of ULSSs, we can consider (among others) the micro-service paradigm [3] from the state-of-practice used by major companies (Amazon, Twitter, Netflix, eBay, PayPal or Shopify) to develop their ULSSs [4]. This paradigm involves several developers in various contexts [5], working on distributed elements orchestrated together [6] to address constantly evolving business-driven requirements [7] for hundreds of thousands of users simultaneously.

In 2013 and in 2019 [8,9], Northrop *et al.* published a retrospective analysis of the work done by the research community and concluded that the efforts made to develop tool support for software developers facing ULSSs development challenges were inadequate. **The work done so far does not consider software developers as first-class stakeholders of the overall development process, making it cumbersome and error-prone for a developer to contribute to such systems**. Developers must be involved in the design process of ULSSs (to introduce functional evolution or to deal with emergent behaviour [10]), as well as in upfront design [11]. ULSSs involves SoC at the requirements and code levels, but also at the technological level, including various communication modes (*e.g.*, asynchronous messaging, message brokers) and paradigms (*e.g.*, reactive programming) that must be consistently assembled in the source code. Developers must compose software components, cross-cutting concerns (*e.g.*, login) with different stacks (*e.g.*, LAMP) and paradigms (*e.g.*, object-oriented, functional programming) to obtain a correct system. They currently only have available approaches designed to operate at smaller scale (*e.g.*, function composition, class merging, component binding, service composition, aspect weaving) [12].

To take a concrete example, the current state-of-the-art composition approaches could easily determine that a developer delivered some artifacts (*e.g.*, code, models, tests) that conflicts with similar artifacts provided by another developer, in a specific subpart of the system where both modifications overlapped, *e.g.*, "*there is a git-merge conflict on line 32 of file X*" [13]. However, it is not yet possible to transform this information into something valuable at an ULSS scale, *e.g.*, "*the implementation provided by team T_1 of feature F_1 that derives from requirement R_A in project P_1 is not compatible at the code level with the implementation of the feature F_2 that derives from another requirement R_B and implemented as a part of project P_2 by team T_2"*. All the different information (*e.g.*, code, architectural models, requirements backlogs) available at small-scale needs to be composed at large-scale to provide such a feedback to developers, considering that a ULSS is in practice more than the simple union of its isolated subparts.

As Northrop *et al.* stated, "*scale changes everything*". **The research challenge of this proposal is in defining a composition model that allows developers to design ULSSs at the right level of abstraction,**

instead of constraining them to use approaches that were defined to solve problems at a smaller scale.

B. Research Objectives

The long-term objective of my research is now to investigate and measure how developers design ULSSs, by identifying the core elements that support developers. I target for the next five years the three short-terms objectives to focus on:

(O1) Identify the concepts necessary to model ULSSs from the developers' points of view. I will first define the concepts necessary to specifically support ULSSs developers, focusing on evolvability and maintainability. I will then make these concepts available to developers as frameworks or domain-specific languages and as a support for analysis to help them while designing ULSSs.

(O2) Define ULSSs composition models to support developers. I will study the needs for software composition at the developer's level and provide accurate compositional models that improve developers' productivity when working on ULSSs. These composition models are rooted in formal (*e.g.*, algebraic) properties to ensure that properties of the designed ULSSs can be proven.

(O3) Empirically analyse how ULSSs are developed. I rely on empirical analysis *(i)* to explore how software developers design ULSSs and then *(ii)* to measure the benefits of the compositional models defined in the project with respect to developers' productivity. I rely on open source-repositories for data collection related to these analyses.

C. Literature Overview

ULSSs architectural modelling. Since 2006, ULSSs have been studied both in academia and industry. For example, OASIS describes the TOSCA standard [14] to design ULS cloud applications. Much effort was put in defining architectural models dedicated to ULSSs [15,16], characterizing expected architecture properties and invariants. Other dimensions, like scalability [17], quality of service [18], or interoperability [19], were investigated independently. In all these works, the developers are never considered as first-class stakeholders, the targeted stakeholder being *(i)* architects with a holistic vision of the ULSSs or *(ii)* infrastructure engineers in charge of the operational scalability. The micro-service paradigm introduced more consideration for the developers to support evolution [20]. But it suffers from several flaws [21] when confronted to ULSSs, such as complexity of design and deployment, cost and security issues.

Evolving requirements modelling. Requirements modelling is critical in ULSSs, as they are characterized by their evolving and potentially conflicting requirements [22,23]. Where classical requirements modelling approaches are related to upfront elicitation, agile development considers the evolvability of user requirements [24]. Agile requirements are composable (*e.g.*, assembled into short development iteration cycles), but not as formalized as classical ones [25], making it difficult to formally reason on them. Recent advances in *natural language processing* (NLP) can automate the transformation of such requirements into domain models [26] that support reasoning. However, the specificity of ULSSs is not addressed by these works: how to measure the impact of a requirement on one another, and on the source code (the size of the requirements and the scattering of related code triggering scalability issues).

Source code analysis. In the last few years, many static analyses appeared to support developers: to improve compilers efficiency [27], identify security flaws [28], or address dependency issues [29]. Some of these analyses are integrated into tools used daily by millions of developers, such as LLVM or GitHub. Source code differencing [30, 31] and merging [32] are now moving to a new level (in terms of accuracy and expressiveness) by leveraging abstract syntax trees representations. However, it is still a major challenge to have several developers working on the same source code [33, 34], and non-polynomial analyses are unsuitable to address ULSSs codebases.

The originality of this research proposal is to advance the state of the art by considering the scalability challenges triggered by the ultra-large scale of the systems, from the developers' points of view.

D. Methodology

My research approach combines a theoretical approach (to model the identified concepts and demonstrate properties formally) with an empirical approach (to identify the concepts based on real-life artifacts, and to validate the added value of the obtained models for developers). The empirical dimension relies on reference open-source repositories (see methodology section for details), and I intend to validate my results through local industrial partnerships.

Considering that the domains of ULSSs are broad, I restrict the scope of my program to three families (F_j) of domains to experiment with it and define more precise research tasks. Developers interact mainly with source code ("*the code is the truth*" [35]), so I will investigate how ULSSs are designed and implemented by developers, considering their intrinsic characteristics and compositional requirements (**F_1**). ULSSs are also characterized by evolving business requirements on a large scale, influenced by agile methods [36]. I will then investigate the compositional support that exists at the requirements level to evolve potentially conflicting requirements, and their impact on the code (**F_2**). Finally, as developers are the cornerstone of this proposal, I will consider their role and involvement in the design of ULSSs [37], and how their productivity can be improved according to the field's best practices (**F_3**). These families of domain will provide a strong engineering background to HQPs that will be involved in the research.

The following table summarizes the tasks to be done when combining these three families (F_j) with the objectives described previously (O_i). The task identified as T_{ij} aims to address the objective O_i, according to the family F_j.

Table 1. Research tasks (T_{ij}) regarding objectives (O_i) and activity families (F_j)

Objectives \ Families	**(F_1) Source code dev.**	**(F_2) Agile Requirements**	**(F_3) Developers' involvement**
(O_1) Modelling	[**T_{11}**] Creating Multi-level ASTs	*Already covered by state-of-the-art [24, 26]*	[**T_{13}**] Defining a dev.-centric model
(O_2) Comp. Model	[**T_{21}**] Developing scalable AST-driven merge	[**T_{22}**] Composing agile artifacts	[**T_{23}**] Supporting evolution at large scale
(O_3) Emp. Analysis	[**T_{31}**] Characterizing code composition	[**T_{32}**] Mining of agile dev. practices	[**T_{33}**] Measuring the impact of ULSSs evolution

For objective O_1, the first task T_{11} consists in defining *Abstract syntax Trees* (ASTs) that are aligned with ULSSs composition requirements. State-of-the-art ASTs are designed from a compilation point of view, being comprehensive and containing all the details of the source code. However, such a detailed representation precludes the definition of efficient AST-based merge and diff algorithms, which are based on sub-graph isomorphism identification, an NP-complete problem. We will define a notion of multi-level AST, *i.e.*, ASTs specifically designed for ULSSs composition purpose, that can be zoomed in or out according to the level of details required by the composition algorithm. These ASTs will also consider the specificity of ULSSs (*e.g.*, event-driven communication) through annotations to reduce the search space and better identify source code modifications in the context of ULSSs. In task T_{13}, we will define an architectural model designed to support developers who work on ULSSs. Whereas the classical models were defined to support architects in a holistic approach, this architectural model will specifically target developers and will be ready to deal with the adaptability and evolvability issues faced by developers in their tasks. Typically, the model will support the definition of *"what-if?"* scenarios to help ULSSs developers decision-making process when confronted with business evolutions.

For objective O$_2$, we will target the definition of a compositional approach dedicated to the models identified in O$_1$. The first task (**T$_{21}$**) addresses the composition challenges that exist at the source code level, relying on state-of-the-art source code differencing approaches and the multi-level ASTs identified in **T$_{11}$**. We will here work on the definition of a family of composition algorithms used to merge ULSSs source code and requirements, considering the specificities of such systems (*e.g.*, remote communication, event-driven message exchange). With this task, we aim to support developers at large scale by providing an implementation of these algorithms integrated into the Git version control system. In task **T$_{22}$**, we will focus on the composition of agile requirements artifacts, such as user stories and backlogs. Thanks to state-of-the-art NLP approaches [38], we can transform such informal requirements into actionable models close to ontologies. We will leverage these models to define a compositional support for developers to deal with such requirements. Based on composition defined at the requirements level (*e.g.*, a sprint backlog that composes stories), the compositional support will propagate them to artifacts manipulated by developers, helping to tailor the effort estimation and risk management of a given sprint. Finally, task **T$_{23}$** will investigates how to support evolution at large scale. We selected this domain because of its relevance at both requirements (*i.e.*, lots of business-driven requirements) and source code (*i.e.*, decomposed source code) levels. We will define a traceability model tailored for this domain and validate how the compositional approach helps the evolvability of the system (an intrinsic property of micro-services architecture) for developers. The validation will rely on reference, open source ULSSs.

For objective O$_3$, the goal of the empirical analysis is twofold: *(i)* analyze existing open-source ULSSs, and *(ii)* evaluate the benefits of the languages and composition algorithms obtained in the tasks described above. Task **T$_{31}$** will address source code composition analysis, leveraging existing studies on merge conflicts and associated corpus [39]. This task will shape the composition algorithm defined by T$_{21}$, as well as measure its benefits for developers based on real-life applications. We will measure the distance that exists between the automatically composed code and the one manually designed in legacy projects as a metric of composition accuracy. Task **T$_{32}$** will target the agile development practices by mining the backlog of selected projects and their associated source code. It will involve the identification of best practices in agile development when applied to ULSSs (shaping **T$_{22}$**), as well as large-scale experiments with existing backlogs to validate the results obtained. Task **T$_{33}$** will consider the evolvability of ULSSs systems globally, considering developers as a crowd (based on a project's metadata available on public project trackers, *e.g.*, *ZenHub*) [40]. It will focus on measuring the impact that some evolution initiated by a given developer has on the work done by others, considering the socio-technical dimension of ULSSs. Version control systems and project management tools track the work of each developer and allow measuring such an impact in legacy ULSSs. The empirical analyses will rely on open-sources applications such as the ULSSs developed on top of the FIWARE framework [41], or references benchmark for micro-services [42]. As a proxy to measure developers' involvement in ULSSs design, we will leverage version control systems, project trackers (*e.g.*, ticketing systems containing requirements backlogs as user stories), and reference corpus (*e.g.*, Visual Narrator corpus [38]) to support automated quantitative analyses.

III. Integration into the Department of Computing and Software at McMaster University

Among the six *Areas of Specialisation* listed on the department website, my research interests are close to the topics investigated in the *Systems* and *Software Quality* areas (principally Dr Zheng, Lawford & Paige). The definition of the theoretical model used to model and support composition at large scale (such as proving properties or optimizing composition algorithms) could benefit from the expertise of the *Theory & Computing* area. With respect to research clusters, my research interests are close into the themes covered by the *Digital & Smart System* one. Considering that *Transportation* systems are ULSSs, this might also be considered as an axis of integration. The expertise of the *Centre for Software Verification* could also be highly beneficial to ensure properties on the composed systems.

Bibliography

[1] Northrop, L., Feiler, P., Gabriel, R. P., Goodenough, J., Linger, R., Longstaff, T., ... & Wallnau, K. (2006). *Ultra-large-scale systems: The software challenge of the future*. Carnegie Mellon University, Software Engineering Institute & Department of Defense technical report.

[2] Dijkstra, E. W. (1974). *On the role of scientific thought*. In Selected Writing on Comp. Collection.

[3] Rubin, J., Wang, Y., Kadiyala, H., Steinbacher, J., & Erwin, T. (2018, October). *Best practices and lessons learned in microservices*. In *Proceedings of the 28th Annual International Conference on Computer Science and Software Engineering*. IBM Corp

[4] Ahmadvand, M., & Ibrahim, A. (2016). *Requirements reconciliation for scalable and secure microservice (de-)composition*. In *24th International Requirements Engineering Conference Workshops (REW)* (pp. 68-73). IEEE.

[5] Jamshidi, P., Pahl, C., Mendonça, N. C., Lewis, J., & Tilkov, S. (2018). Microservices*: The journey so far and challenges ahead*. *IEEE Software*, *35*(3), 24-35.

[6] Newman, S. (2015). *Building microservices: designing fine-grained systems*. O'Reilly Media, Inc.

[7] Dragoni, N., Giallorenzo, S., Lafuente, A. L., Mazzara, M., Montesi, F., Mustafin, R., & Safina, L. (2017*). Microservices: Yesterday, Today, and Tomorrow.* In *Present and Ulterior Software Engineering*, 195–216. Springer.

[8] McGregor, J. D., Aoyama, M., Northrop, L. M., & Schmid, K. (2013). *Scale changes everything, but...* In *Proceedings of the 17th International Software Product Line Conference*. ACM.

[9] Northrop, L., Ozkaya, I., Fairbanks, G. & Keeling, M. (2019). *Designing the Software Systems of the Future*. *SIGSOFT Softw. Eng. Notes* 43, 4, 28-30.

[10] Alshuqayran, N., Ali, N., & Evans, R. (2018). *Towards Micro Service Architecture Recovery: An Empirical Study*. In *IEEE International Conference on Software Architecture* (ICSA).

[11] Vernon, V. (2016). *Domain-driven design distilled*. Addison-Wesley Professional.

[12] Pautasso, C., Zimmermann, O., Amundsen, M., Lewis, J., & Josuttis, N. (2017). *Microservices in practice, part 1: Reality check and service design*. *IEEE Software*, (1), 91-98.

[13] Leßenich, O., Siegmund, J., Apel, S., Kästner, C., & Hunsen, C. (2018*). Indicators for merge conflicts in the wild: Survey and empirical study*. *Automated Software Engineering*, *25*(2), 279-313

[14] Zimmermann, M., Breitenbücher, U., & Leymann, F. (2017, April). A Method and Programming Model for Developing Interacting Cloud Applications Based on the TOSCA Standard. In *International Conference on Enterprise Information Systems* (pp. 265-290). Springer, Cham.

[15] Issarny, V., Bouloukakis, G., Georgantas, N., & Billet, B. (2016, October). *Revisiting service-oriented architecture for the IoT: A middleware perspective*. In *International conference on service-oriented computing* (pp. 3-17). Springer, Cham.

[16] Chen, H. M., Kazman, R., & Haziyev, S. (2016). *Strategic prototyping for developing big data systems*. In *IEEE Software*, *33*(2), 36-43.

[17] Zhao, G., Ling, C., & Sun, D. (2015, May). *Sparksw: Scalable distributed computing system for large-scale biological sequence alignment*. In *2015 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing* (pp. 845-852). IEEE.

[18] Gan, Y., & Delimitrou, C. (2018). *The Architectural Implications of Cloud Microservices*. *IEEE Computer Architecture Letters*, *17*(2), 155-158.

[19] Spalazzese, R., Pelliccione, P., & Eklund, U. (2017). *INTERO: An interoperability model for large systems*. In *IEEE Software*.

[20] Sampaio, A. R., Kadiyala, H., Hu, B., Steinbacher, J., Erwin, T., Rosa, N., ... & Rubin, J. (2017). *Supporting microservice evolution*. In *2017 IEEE International Conference on Software Maintenance and Evolution (ICSME)* (pp. 539-543). IEEE.

[21] Jamshidi, P., Pahl, C., Mendonça, N. C., Lewis, J., & Tilkov, S. (2018). *Microservices: The journey so far and challenges ahead*. In *IEEE Software*, *35*(3), 24-35.

[22] Safwat, A., & Senousy, M. B. (2015). *Addressing challenges of ultra-large-scale system on requirements engineering*. In *Procedia Computer Science*, *65*, 442-449.
[23] Beidu, S., & Atlee, J. M. (2019). *Detecting Feature Interactions in FORML Models*. In *From Software Engineering to Formal Methods and Tools, and Back* (pp. 220-235). Springer, Cham.
[24] Schön, E. M., Thomaschewski, J., & Escalona, M. J. (2017). *Agile Requirements Engineering: A systematic literature review*. In *Computer Standards & Interfaces*, *49*, 79-91.
[25] Dick, J., Hull, E., & Jackson, K. (2017). *Requirements engineering, 4th edition*. Springer.
[26] Müter, L., Deoskar, T., Mathijssen, M., Brinkkemper, S., & Dalpiaz, F. (2019, March). *Refinement of User Stories into Backlog Items: Linguistic Structure and Action Verbs*. In *International Working Conference on Requirements Engineering: Foundation for Software Quality* (pp. 109-116). Springer.
[27] Asher, Y. B., Haber, G., & Stein, E. (2017). *A study of conflicting pairs of compiler optimizations*. In *2017 IEEE 11th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC)* (pp. 52-58). IEEE.
[28] Imtiaz, N., & Williams, L. (2019, April). *A synopsis of static analysis alerts on open-source software*. In *Proceedings of the 6th Annual Symposium on Hot Topics in the Science of Security* (p. 12). ACM.
[29] Zhang, B., Tenev, V., & Becker, M. (2016). *Android build dependency analysis*. In *24th International Conference on Program Comprehension (ICPC)* (pp. 1-4). IEEE.
[30] Falleri, J. R., Morandat, F., Blanc, X., Martinez, M., & Monperrus, M. (2014, September). *Fine-grained and accurate source code differencing*. In *Proceedings of the 29th ACM/IEEE international conference on Automated software engineering* (pp. 313-324). ACM.
[31] Rubin, J., & Chechik, M. (2012, September*). Locating distinguishing features using diff sets*. In *Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering* (pp. 242-245). IEEE.
[32] Leßenich, O., Apel, S., & Lengauer, C. (2015). *Balancing precision and performance in structured merge*. In *IEEE Automated Software Engineering*, *22*(3), 367-397.
[33] Owhadi-Kareshk, M., Nadi, S., & Rubin, J. (2019). *Predicting Merge Conflicts in Collaborative Software Development. arXiv preprint arXiv:1907.06274*.
[34] Cavalcanti, G., Borba, P., Seibt, G., & Apel, S. (2019) *The Impact of Structure on Software Merging: Semistructured versus Structured Merge*. In *Proceedings of the 29th IEEE/ACM International Conference on Automated Software Engineering.* IEEE.
[35] Martin, R. C. (2009). *Clean code: A handbook of agile software craftsmanship*. Pearson Education.
[36] Rezaei, R., Chiew, T. K., & Lee, S. P. (2014). *An interoperability model for ultra-large-scale systems*. In *Advances in Engineering Software*, *67*, 22-46.
[37] Galster, M., Weyns, D., Goedicke, M., Zdun, U., Cunha, J., & Chavarriaga, J. (2018). Variability and Complexity in Software Design: Towards Quality through Modeling and Testing. *ACM SIGSOFT Software Engineering Notes*, *42*(4), 35-37.
[38] Lucassen, G., Dalpiaz, F., van der Werf, J. M. E., & Brinkkemper, S. (2016). *Improving agile requirements: The quality user story framework and tool*. In *Requirements Engineering*, *21*, 383-403.
[39] Menezes, G. G. L., Murta, L. G. P., Barros, M. O., & Van Der Hoek, A. (2018). *On the Nature of Merge Conflicts: A Study of 2,731 Open-Source Java Projects Hosted by GitHub*. In *IEEE Transactions on Software Engineering* (DOI: 10.1109/TSE.2018.2871083)
[40] Viviani, G., Janik-Jones, C., Famelis, M., Xia, X., & Murphy, G. C. (2018). *What design topics do developers discuss?* In *Proceedings of the 26th Conference on Program Comprehension*. ACM.
[41] B. Cheng, S. Longo, F. Cirillo, M. Bauer and E. Kovacs. (2015) *Building a Big Data Platform for Smart Cities: Experience and Lessons from Santander*. In *IEEE Conference on Big Data*.
[42] J. von Kistowski, S. Eismann, N. Schmitt, A. Bauer, J. Grohmann and S. Kounev. (2018) *TeaStore: A Micro-Service Reference Application for Benchmarking, Modeling and Resource Management Research*. In *IEEE Proceedings of 26th International Symposium MASCOTS.*

Teaching Statement

This document briefly describes my teaching philosophy (section I), and then present the courses I have taught as Associate Professor at *Université du Québec à Montréal* (UQAM, 2019-…) and *Université Côte d'Azur* (UCA, 2012-2018).

I. Teaching Philosophy: "*Code is the truth*", but not the only truth.

Teaching software engineering is challenging. When working on a small scale, students consider that the methods and techniques presented are cannons used to kill flies. However, when working at a "real" scale, students spend more time understanding legacy code than properly assimilating new concepts. To address these issues, I heavily rely on code in my courses. As professors, we are supposed to provide students with the conceptual tools that will allow them to survive the unavoidable technological changes. However, software engineering students will not become achieved professionals without putting their hands in code and experimenting with what it means to design, implement, evaluate and measure software. However, this must not be done by forgetting the human dimension of software engineering, e.g., requirements engineering and user acceptance. A software engineer is not a geek living in his/her parent's basement and playing video games all day long but is an achieved professional able to cover all stages of a software lifecycle, from requirements engineering to software architecture, quality assessment and continuous delivery.

A. Project-based approach

To achieve this goal, all my courses come with *(i)* a project session assignment and *(ii)* a reference implementation that students can study. With this kind of setup, I can focus during the lectures on technologically independent concepts. Then hands-on labs and projects are used to acquire the technical skills associated with these concepts. To achieve the scalability issue mentioned in the introduction of this section, I was in charge of UCA's Innovation Projects, a collaboration with the local School of Management where students were asked to create a start-up in one month, working full time on their project. Management professors were accompanying them on business plans, ideation and marketing, while software engineering professors focused on agility and software quality. This model was implemented in other universities in France, and I was invited to advise 17 different programs concerning project-based teaching at the national level. I also led to two keynotes related to the teaching of software engineering in an agile context. I am currently leading at UQAM a reform to reshape our B.Sc. and include two project-based courses as part of the mandatory curriculum.

B. Synergy with Research

Being associated with research is what defines university teaching. As professors, we are not delivering technical training that will be outdated in few years. However, instead, we are providing students with the right abstractions to deliver high-quality software properly. I firmly believe in associating students with research teams, starting at the undergraduate level. During my courses, I always include paper readings as mandatory assignments, even at the undergraduate level. The goal here is to create a scientific reading culture instead of having students continually reinventing the wheel. To advertise for research, I interacted with the student association at UQAM and UCA to organize monthly seminars where young professors could pitch their research and advertise for their labs. These seminars were students' initiatives and attracted up to 100 students each month in Nice. Starting in 2014, I also started to dedicate a non-negligible amount of my time to publish papers related to teaching. As researchers, I believe it is essential for us to take a step back and have a scientific approach to what and why we are teaching some concepts and how we are teaching them.

C. Integrating with local industrial partners

Software engineering is all about engineering. I try to invite at least one industrial partner to give an invited presentation in each of my courses to put in perspective the material covered in the lectures with industrial experience. As I joined UQAM in 2019, I acted as liaison with the COOP internship office to better understand the local ecosystem. By co-founding the Computer Science Night in 2007 in France, I have participated for 12 years in a row to promote computer science and software engineering to students, from high school to Ph.D. This friendly competition brings together students from diverse origins and backgrounds once a year. They have to tackle challenges proposed by industrial partners, who share their expertise with the students. Professors are here to help beginners and support their achievements during the night, from sunset to sunrise. The last edition (before COVID-19) involved more than 4,000 students, located in 55 cities, with 52 industrial partners and more than 60k$ of prizes.

II. Courses Insights & Administrative duties

Since 2012, I am teaching courses related to software engineering. An extensive list of courses taught is available in my resume, and my teaching material is available on the GitHub profile and YouTube channel of my research group[1]. It is important to me to share my material with other professors or prospective students. As an Assistant and then Associate Professor at UCA in a French School of engineering, my courses focused on software architecture at the graduate level. At UQAM, I am still teaching software engineering topics, with a greater emphasis on software design, at the undergraduate level. Due to the COVID-19 pandemic, I redesigned my design course as a series of French-speaking podcasts, representing 11 hours of asynchronous video contents and synchronous case studies. This course is now reused in four universities. The content quality was remarked by UQAM, which is now funding studio recording and professional editing to do similar work at the graduate level. I am also coordinating a funded collaboration between UCA and UQAM to raise awareness of the challenges triggered by the ageing population from a software engineering perspective.

When I was elected as head of the Software Architecture M.Sc. in 2013, I also coordinated the "*Introduction to software engineering*" course at the undergraduate level to raise awareness of this topic in the cohorts. Under my direction, the M.Sc. tripled its number of students, and became the most selective M.Sc. in the School of Engineering, with an employment rate of 100% and the greatest number of students pursuing a Ph.D. after graduation. At UQAM, I am in charge of coordinating the courses related to software engineering, thus acting as deputy director for Software Engineering for the B.Sc. and M.Eng. programs (850 students).

III. Integration into the Department of Computing and Software at McMaster University

This section is based on the course portfolio described on McMaster website. At the undergraduate level, my main skills are related to courses related to software engineering and architecture, e.g., COMPSCI 1XD3, COMPSCI 2ME3, COMPSCI 3MI3, COMPSCI 4AR3 and COMPSCI 4HC3, as well as their SFWRENG counterparts when relevant. I can also contribute to courses related to distributed systems (e.g., COMPSCI 4DC3) and web development (e.g., COMPSCI 4WW3). According to the availability and need of the department, I would be happy to propose courses related to variability management and DevOps and micro-services development. I am not a registered engineer in Canada, but the CEAB recognizes my French engineering degree and I can commit to apply for a professional licence in Ontario if such a registration is required for some specific programs (e.g., B. Eng.).

In June 2020, Dr. Zheng received a CREATE grant to improve mobility for the aging population. This is related (at a different scale) to the work done in collaboration with UCA. I would be happy to share experience and feedback on this critical topic considering Canada's demographic profile.

[1] https://github.com/ace-lectures (course material) & https://www.youtube.com/AceResearchLab (podcasts)

Equity, Diversity and Inclusion Statement

Equity, Diversity and Inclusion (EDI) is a crucial issue in software engineering and computer science. According to Statistics Canada, only 15.8% of first-year students identifies as women. I was voiceless when I realized that my first engineering cohort in France contained only two women out of 120 students. Maintaining such disbalance is equivalent to shooting ourselves in the feet. This important topic is now addressed by initiatives like "Software Engineering Women in Research" led by Pr. Margaret Storey (http://margaretstorey.com/sewire/). Unfortunately, EDI is about gender disbalance in our field and about improving diversity and making minorities more visible. This topic starts to be addressed in our community, e.g., through the organization of LGBT lunches at major conferences such as ICSE or the work done by Pr. Alexander Serebrenik to advertise for "*Gender, Diversity and Inclusion and Software Engineering*" in a series of keynotes.

I. Concrete actions taken since 2012

When working at UCA, my recruitment pool for student supervision was driven by the School of Engineering admission process, which did not favour diversity and gender equality by design. As I had no control over this process, I tried to counterbalance it by presenting strong women acting as role models in the software engineering community. Moving to UQAM opened my recruitment pool, leading to a research team with more diversity. My research team now includes more members who identify as women, gay, as well as non-binary.

I am convinced that it is part of our job as professors to make students aware of EDI issues. To support this, I introduced in my undergraduate course a mandatory reading addressing this topic (Gender in Software Engineering, Carver & Serebrenik, IEEE Software 2019). In my graduate course related to software maintenance, I included readings related to gender and diversity issues in open-source software. Students were shocked when they realized that gender bias was existing under the hood on GitHub, for example. To be as inclusive as possible, I also refused to use any virtual proctoring system during the COVID-19 pandemic. I organized a community of practice with professors and lecturers involved in the Summer and Fall sessions of 2020. Again, the idea was to create a safe space to exchange and discuss our practices.

Working at UQAM, which has a large part of its student population accessing University for the first time, raised my awareness of the difficulty of studying while being a single mother and the hidden discrimination that can exist when non-binary students apply for jobs. Thus, I am interacting with *EllesCode*, the UQAM local chapter of the ACM-WS initiative, to advertise and encourage women to enroll in our B.Sc. program and create a safe space inside the University. Thanks to these interactions, I was involved in several research projects (*i.e.*, RELAI & Mentallys) related to mental disorder and e-health, which I am currently starting to discover through these projects.

II. Prospective actions

I want to emphasize my involvement in advertising software engineering to minorities. My previous efforts were essentially focused on my research team. However, I would like to address these issues on a larger scale by doing presentations or organizing hackathons in high schools.

The representation of LGBTQ2S+ is a topic that I have not consciously addressed in my group. I plan to be more involved with local associations of LGBTQ2S+ people in STEM at the local level to understand better the challenges they face in the industry and address them in my courses.

Sébastien Mosser

Professeur *of Software Engineering*

Université du Québec à Montréal
Comp. Science Department
Montréal, Canada
☎ *+1 514-987-3000 (poste 3904)*
✉ *mosser.sebastien@uqam.ca*
https://mosser.github.io

Education

2007 – 2010 **Ph.D. in Computer Science (*Doctorat*)**, *Université de Nice (FR).*
- Title : *Behavioral Compositions in Service-oriented Architectures*
- Supervisors : Michel Riveill & Mireille Blay–Fornarino
- Reviewers : Don Batory (UT Austin, USA) & Xavier Blanc (Univ. Bordeaux, FR)
- Examiners : Lionel Seinturier (Univ. Lille, FR) & Pierre–Alain Muller (Univ. Mulhouse, FR)

2004 – 2007 **Engineer degree (*Ingénieur CTI*)**, *Université de Nice (FR).*
- Major: Distributed systems, Minor: Software Engineering (valedictorian)
- *Engineer title recognized by the the CEAB to apply for professional license in Canada*

2002 – 2004 ***Diplôme d'Études Universitaires Générales* (DEUG)**, *Université de Nice (FR).*
- Mathematics & Computer Science applied to Science (MIAS)
- Specialization in Computer Science

Academic Employment

2019 – ... ***Professeur* of Software Engineering**, *Université du Québec à Montréal (CA).*
- Tenured, Category III (equivalent to Associate Professor)
- Teaching : Computer Science Department
  - Deputy director for soft. eng. of the Computer Science and Software Engineering B.Sc.
  - Deputy director of the Software Engineering M. Eng.
- Research : ACE research group
  - `http://ace-design.github.io`

2012 – 2018 ***Maître de Conférences* of Computer Science**, *Université Côte d'Azur (FR).*
- Tenured, equivalent to Associate Professor (leave-of-absence until 31/03/2022)
- Teaching: Polytech Nice – Sophia School of Engineering
  - Elected board member of the CS department (2014 - 2018)
  - Coordinating Software engineering programs (2014 - 2018)
  - Director of the "Software Architecture" M.Sc. (2013 - 2018)
  - Coordinating project-based teaching (2012 - 2018)
- Research : I3S Laboratory (CNRS), SPARKS team.
  - Board member of the lab (2018)
  - Member of the "*Scalable Software Systems*" (S^3) group.

2011 – 2012 **Research Scientist in Distributed Systems**, *SINTEF ICT (NO).*
- Department: *Network & Secured Systems*, MOD Team.

2010 – 2011 **Postdoctoral Fellow**, *Inria Lille–Nord Europe (FR).*
- ADAM (now SPIRALS) team, supervised by Pr. Laurence Duchien

2007 – 2010 ***Moniteur de l'enseignement supérieur***, *Polytech Nice – Sophia (FR).*
- A *moniteur* is recruited for three years after a selective process at the university level and is involved in a mentoring process to prepare for assistant professor positions.

Awards

2015 – 2019 ***Prime d'Encadrement Doctoral et de Recherche***, *National University Council.*
Distinction awarded to the best 20% of Associate professors (14K€), at the national level

Best Paper Award
- *"Assessing the Functional Feasibility of Variability-Intensive Data Flow-Oriented Systems"*, S. Lazreg *et al*, 33th Symposium on Applied Computing (SAC'18 [24])

Research Subventions

Ongoing projects

2020–2025 **Discovery Grant**, *NSERC*, 145K$.
- Role: Principal Investigator
- *A Compositional Approach to Support Developers in Developing Ultra-large-scale Systems*

2020–2023 **CAPESA**, *Inria Associated Team*, 50K$.
- Role: Principal Investigator
- Collaboration with the CASH team (ENS Lyon) on software compilers

2020–2023 **Mentalys**, *FRQNT Audace*, 100K$.
- Role: Co-researcher (part of funding: 10K$)
- Investigating software engineering methods to develop mental health application

2019–2021 ***RELAI***, *New Frontiers in Research*, 250K$.
- Rôle : Co-PI
- *Respectful and Explainable AI to Support Struggling People and Mental Health Practitioners*

2019–2021 ***ACE-SG***, *UQAM*, 15K$.
- Role: Principal Investigator
- UQAM starting grant (Faculty of science, amount regulated by collective agreement)

2019–2021 ***GL-AGE***, *FRQNT & Ministère des Affaires Étrangères Français*, 40K$.
- Role: Principal Investigator (Currently on hold because of COVID-19)
- *French-Québec* collaboration related to aging and software engineering

Past Projects, as PI or Co-PI (560K€)

2018–2020 ***Formalizing Scalable Composition Operators* (FiaSCO)**, *CNRS*, 5K€.
- Role: Principal Investigator
- Mobility award to fund interviews with experts, leading to the creation of the ACE research group

2017–2020 **Software Composition for the MERMAID**, *Regional grant*, 90K€.
- Role: Co-PI
- Definition of a domain-specific language to compose applications deployed on underwater floating buoy.

2016–2020 **Variability in Cyber-Physical Systems**, *Industrial Contract, VISTEON*, 77K€.
- Role: Co-PI
- Leveraging "*Feature Transition Systems*" to better design car dashboards

2016–2019 **Modelling Software Composition**, *UCA, School of Graduate Studies*, 100K€.
- Role: PI
- Suppoprt HQPs recruitement to explore software composition from a modelling point of view.

2016 ***Modeling for Scaling* (M4S)**, *CNRS new faculty*, 10K€.
- Role: PI
- Preliminary work to kick-start the *Modelling Software Composition* project

2015 – 2018 **Technological Transfer**, *EIT Digital*, 33K€.
- Rôle : Co-PI
- Transferring the DEPOSIT tool defined in the group to the DataThing company

2014 – 2017 ***Tailored Comp. for Large-scale Sensing Network***, *UCA, School of Graduate Studies*, 100K€.
- Rôle : Co-PI
- Support HQPs recruitement to explore software composition applied to large-scale sensor networks in smart cities

2013 – 2016 ***Model-based Sensor Data Visualizations***, *UCA, School of Graduate Studies*, 100K€.
- Role: PI
- Support HQPs recruitement to explore the definition of dashboard and data visualization in a composable way

2012 – 2014 **IDOL**, *EGIDE Aurora*, 20K€.
- Rôle : Co-PI
- Norwegian-France collaboration related to modelling application deployed on cloud-computing environment

2012 – 2014 **Mod4Cloud**, *Amazon Research Grant*, 25K€.
- Role : PI
- Experimental benchmarking of cloud deployment languages

Past Projects, as regular participant

2018–2020 ***I-WIN***, *Initative of Excellence UCA*, 36K€.
- Rôle: Work-package leader
- Sensors dynamic reconfiguration using artificial intelligence

2018–2020 ***SmartIoT for Mobility***, *Initative of Excellence UCA*, 25K€.
- Rôle: Work-package leader
- Smart contracts for cyber-physical systems

2012 – 2014 **YourCast**, *Technological Transfer, Agence Nationale de la Recherche*, 250K€.
- Role: Technical expert
- Starting fund to create a start-up company exploiting a piece of software defined in the group

2011 – 2012 **MODAClouds**, *EU FP7 program*, 8.7M€.
- Role: Work-package leader
- Definition of the CloudML language

2011 – 2012 **PaaSage**, *EU FP7 program*, 9.7M€.
- Role: Work-package leader
- Integrationg the CloudML language with existing cloud provider

2011 – 2012 **REMICS**, *Programme EU FP7*, 4.5M€.
- Role: Technical expert
- Model-driven engineering applied to the reverse engineering of legacy aplication into cloud-based ones

2011 – 2012 **ENVISION**, *Programme EU FP7*, 4.5M€.
- Role: Technical expert
- Defining a *Model as a service* environment

Graduate Students supervision

Summary (since 2011)
- Master students: 8 (2 ongoing)
- Doctoral students: 8 (2 ongoing)
  - In the French system, professors are not allowed to supervise PhD students before defending an habilitation thesis, excepting an exceptional allowance granted by the president of the university.

Ph.D. supervision

Starting Fall 2021 **Serge Dogny-Gagnely**, *UQAM*, Industrial collaboration.
- Developing data-driven applications in a composable way

Since 01/2021 **Alexandra Lapointe-Boisvert**, *UQAM*, Industrial collaboration.
- Composable requirements in an Agile & DevOps context
- Co-supervised with Pr. Sylvie Trudel

2017 – 2020 **Sébastien Bonnieux**, *Université Côte d'Azur*, Industrial collaboration.
- MDE-based approach to compose embedded applications
- Co-supervised with Pr. Mireille Blay-Fornarino
- Publications: [7, 23].

2016 – 2019 **Benjamin Benni**, *Université Côte d'Azur*.
- Composition black-boxes composition operators.
- Publications: [10, 8, 18, 21, 12, 25, 26, 48].

2016 – 2020 **Sami Lazreg**, *Université Côte d'Azur*, Industrial collaboration.
- Modelling software variability in embedded systems
- co-supervised with Pr. Philippe Collet
- Publications : [22, 14, 24].

2014 – 2017 **Cyril Cecchinel**, *Université Côte d'Azur*.
- Modelling composable aplications for sensor networks
- co-supervised with Pr. Philippe Collet
- Publications: [11, 29, 30, 31].

2013 – 2017 **Ivan Logre**, *Université Côte d'Azur*.
- Visualizing data collected at large scale by sensor networks
- Publications: [33, 57, 68].

2010 – 2014 **Alexandre Feugas**, *Université Lille 1*, Industrial collaboration.
- Business Processes Evolution
- co-supervised with Pr. Laurence Duchien
- Publications: [34, 72].

M.Sc. Supervision

Starting Fall 2021 **Corinne Pulgar**, *UQAM*.
- Using justification diagrams to express ethical requirements

Since 01/2020 **Jean-Philippe Caissy**, *UQAM*.
- Reverse-engineering of micro-service applications in a composable way
- Publications: [21, 18].

2017–2018 **Günther Jungbluth**, *Université Côte d'Azur*, (CNRS apprentice program).
- Developing scalable data-processing pipelines
- Publications: [52].

2016 **Benjamin Benni**, *Université Côte d'Azur*.
- A language-driven approach for model composition

2014 **Cyril Cecchinel**, *Université Nice – Sophia Antipolis*.
- Code generation applied to sensor networks
- Publications : [53].

2013 **Ivan Logre**, *Université Nice – Sophia Antipolis*.
- User-centered dashboards for data collected by large scale sensor networks

2011 **Eirik Brandtzæg**, *Universitetet i Oslo*.
- CloudML, A DSL for model-based realization of applications in the cloud
- Publications: [35, 61].

Teaching

Courses currently offered at Université du Québec à Montréal

MGL7361 **Principles of Software Design**, *Graduate*, M.Eng..
- Course currently refactored as podcasts with institutional support

MGL7460 **Developing & Maintaining Software**, *Graduate*, M.Eng.
- Course offered to regular students as well as Desjardins employees

INF5153 **Software Design**, *Undergrad*, B.Sc..
- Course developed as french-speaking podcasts due to COVID-19
- Podcasts reused by four universities since Fall 2020
- Preliminary webiste available here: `https://conception-objet.github.io/`

INF600G **Designing Software for Aging Population**, *Undergrad*, B.Sc., Elective.
- Course developed as part of a French-Quebec collaboration
- Webiste available here: `https://ace-design.github.io/champlain/`

Invited Lecturer

ENS Lyon **Software Engineering & Compilation**, *Graduate*.
- Course given in 2018 & 2019 (Fall session) as invited lecturer in Lyon
- *École Normale Supérieure de Lyon* is a highly selective & research-intensive University

CNRS **Model-driven Engineering**, *Graduate*.
- Course given in 2016 & 2017 during the national Summer School of Software Engineering

Courses previously offered at Université Côte d'Azur

Domain-specific Language Engineering, *Graduate*.

Service-oriented Architectures, *Graduate*.

Software Architecture & DevOps, *Undergrad & Graduate*, Industrial collaboration.

Business Process Modelling, *Graduate*.

Innovation projects & Entrepreneurship, *Graduate*, Industrial Collaboration.

Introduction to Software Engineering, *Undergrad.*

Professional Service

Steering Committee

- Co-president, *Int. Workshop on Modularity in Modeling*, 2016-…
- Co-president, *Int. Workshop on DevOps at MODELS*, 2019-…
- Co-president, *Int. Workshop on MDE Evaluation*, 2019-…

Conference Organization

- *Virtual conference chair*, MODELS (2020)
- *Accomodation Chair*, ICSE (2019)
- *Social Media Chair*, Modularity (2015)
- *Career development co-chair*, SERVICES (2012)
- *General logistics*, Journées nationales du GdR GPL (2011)
- *Demo chair*, *Benelux Software Evolution* (2010)

Program Committee

- *Int. Conf. on Software Engineering* Artefact Evaluation (ICSE), 2021
- *Int. Conf. on Model Driven Eng. Lang. and Systems* (MODELS), 2021
- *Int. Conf. on Systems and Software Product Line* (SPLC), 2020
- *Int. Conf. on Model Driven Eng. Lang. and Systems* Doc. Symp. (MODELS), 2020
- *Int. Workshop. on Software Engineering for the IoT* (SERP4IOT), 2020
- *Int. Conf. on Research Challenges in Information Science* Doc. Symp. (RCIS), 2019
- *Int. Conf. on Model-driven Engineering & Soft. Dev.* (MODELSWARD), 2019 - …
- *Int. Workshop on Modeling for Micro-services*, 2018 - …
- *Int. Conference on Big Data* (BigData), 2015 - …
- *Int. Workshop on Scalable Data Management* (SCDM), 2014 - …
- *Int. Workshop on Model-driven Cloud engineering*, 2014
- *Int. Conference on Web Services* (ICWS), 2013 - …
- *Nordic Workshop on Cloud computing*, 2013 & 2014

Ph.D. Committee

- External :
  - 2021: Alexandre Rio (Université de Rennes, reviewer)
  - 2021: Thibault Béziers la Fosse (Mines-Télécom Bretagne, examiner)
  - 2019: Hyacinth Ali (McGill, thesis committee)
  - 2016: Mai Anh Bui (Université Paris 6, reviewer)
- Internal :
  - Dimitri Prestat (UQAM, Projet de Thèse)

Reviewer

- *Journal of Object Tehcnology* (JOT)
- *Journal of Computer Language* (COLA)
- *Journal of Internet of Things* (IoT)
- *Transactions on Cloud Computing* (TCC)
- *Journal of Software and Systems* (JSS)
- *Software & System Modelling* (SoSym)
- *Software Quality Journal* (SQJ)
- *Empirical Software Engineering* (ESE)

Funding Evaluation

- *Agence Nationale de la Recherche* (2011, 2020)
- FRQNT doctoral grants (2020, 2021)
- NSERC USRA at Université du Québec à Montréal (2019, 2020)
- NSERC Discovery, external reviewer (2017)
- IMT Atlantique, team creation evaluation (2020))

Workgroup coordination

- SE@MTL: montly-based seminars (37 profs involved)
- GL/\CE (2015-2020): Software engineering applied to CPS (17 research groups involved)
- PING (2013): Software engineering teaching (13 research groups involved)

Publications (2007 – . . .)

Underlined names are students who were under my supervision at the time of the work described in the paper.

Invited Presentations (Keynotes)

[1] B. Benni and **S. Mosser**. Applying Software Composition to the Docker Ecosystem. Oct. 2018. Amadeus Global Tech Forum.

[2] **S. Mosser**. Les aspects génie logiciel pour les Systèmes Cyber-Physique. In *Journées IIoT du GDR MACS, CNRS*, France, July 2018.

[3] **S. Mosser**. Renforcer l'engagement étudiant en projet. July 2017. Journées sur la pédagogie active, Université Bretagne-Loire.

[4] **S. Mosser**. Projets, Agilité & École d'Ingénieur. Mar. 2017. Journées sur l'Innovation Pédagogique, Université du Maine.

[5] V. Aranega, A. Etien, and **S. Mosser**. Using Feature Model to build Model Transformation Chains. In *Journées 2013 du GDR GPL, CNRS*, France, Mar. 2013.

[6] **S. Mosser**, G. Mussbacher, M. Blay-Fornarino, and D. Amyot. Une approche orientée aspect allant du modèle d'exigences au modèle de conception. In *Journées du GDR GPL*, pages 37–38, Lille, France, June 2011.

International Journal Papers

[7] S. Bonnieux, D. Cazau, **S. Mosser**, M. Blay-Fornarino, Y. Hello, and G. Nolet. MeLa: A Programming Language for a New Multidisciplinary Oceanographic Float. *MDPI Sensors*, 2020.

[8] B. Benni, **S. Mosser**, M. Acher, and M. Paillart. Characterizing Black-box Composition Operators via Generated Tailored Benchmarks. *Journal of Object Technology (JOT): special issue ECMFA'20*, June 2020.

[9] G. Mussbacher, B. Combemale, J. Kienzle, S. Abrahão, H. Ali, N. Bencomo, M. Búr, L. Burgueño, G. Engels, P. Jeanjean, J.-M. Jézéquel, T. Kühn, **S. Mosser**, H. Sahraoui, E. Syriani, D. Varró, and M. Weyssow. Opportunities in Intelligent Modeling Assistance. *Software and Systems Modeling*, 2020.

[10] B. Combemale, J. Kienzle, G. Mussbacher, H. Ali, D. Amyot, M. Bagherzadeh, E. Batot, N. Bencomo, B. Benni, J.-M. Bruel, J. Cabot, B. H. C. Cheng, P. Collet, G. Engels, R. Heinrich, J.-M. Jézéquel, A. Koziolek, **S. Mosser**, R. Reussner, H. Sahraoui, R. Saini, J. Sallou, S. Stinckwich, E. Syriani, and M. Wimmer. A Hitchhiker's Guide to Model-Driven Engineering for Data-Centric Systems. *IEEE Software*, 2020.

[11] C. Cecchinel, F. Fouquet, **S. Mosser**, and P. Collet. Leveraging live machine learning and deep sleep to support a self-adaptive efficient configuration of battery powered sensors. *Future Generation Computer Systems (FGS)*, Mar. 2019.

[12] B. Benni, **S. Mosser**, N. Moha, and M. Riveill. A Delta-oriented Approach to Support the Safe Reuse of Black-box Code Rewriters. *Journal of Software: Evolution and Process (JSEP), ICSR special issue*, July 2019.

[13] L. Burgeno, F. Ciccozzi, M. Famelis, G. Kappel, L. Lambers, **S. Mosser**, R. Paige, A. Pierantonio, A. Rensink, R. Salay, G. Taentzer, A. Vallecillo, and M. Wimmer. Contents for a Model-Based Software Engineering Body of Knowledge. *Journal of Software and Systems Modeling*, June 2019.

[14] S. Lazreg, P. Collet, and **S. Mosser**. Functional Feasibility Analysis of Variability-Intensive Dataflow-oriented Applications over Highly-configurable Platforms. *ACM SIGAPP Applied Computing Review*, Sept. 2018.

[15] B. Combemale, J. Kienzle, G. Mussbacher, O. Barais, E. Bousse, W. Cazzola, P. Collet, T. Degueule, R. Heinrich, J.-M. Jézéquel, M. Leduc, T. Mayerhofer, **S. Mosser**, M. Schöttle, M. Strittmatter, and A. Wortmann. Concern-Oriented Language Development (COLD): Fostering Reuse in Language Engineering. *Computer Languages, Systems and Structures*, 2018.

[16] **S. Mosser** and M. Blay-Fornarino. ADORE, a Logical Meta-model Supporting Business Process Evolution. *Science of Computer Programming*, 78(8):1035 – 1054, 2013.

[17] **S. Mosser**, M. Blay-Fornarino, and R. France. Workflow Design using Fragment Composition (Crisis Management System Design through ADORE). *Transactions on Aspect-Oriented Software Development (TAOSD)*, Special issue on Aspect Oriented Modeling:1–34, 2010.

International Conference Papers

[18] **S. Mosser**, J.-P. Caissy, F. Juroszek, F. Vouters, and N. Moha. Charting Microservices to Support Services' Developers: the Anaximander Approach. In *International Conference on Service-Oriented Computing (ICSOC), short paper*, Dec. 2020.

[19] G. Mussbacher, B. Combemale, S. Abrahão, N. Bencomo, L. Burgueño, G. Engels, J. Kienzle, T. Kühn, **S. Mosser**, H. Sahraoui, and M. Weyssow. Towards an Assessment Grid for Intelligent Modeling Assistance. In *MDE Intelligence 2020 - 2nd Workshop on Artificial Intelligence and Model-driven Engineering*, Oct. 2020.

[20] D. Maupomé, M. D. Armstrong, R. M. Belbahar, J. Alezot, R. Balassanio, M. Queudot, **S. Mosser**, and M.-J. Meurs. Early mental health risk assessment through writing styles, topics and neural models. In *Working Notes of CLEF 2020 - Conference and Labs of the Evaluation Forum*, 2020.

[21] B. Benni, **S. Mosser**, J.-P. Caissy, and Y.-G. Guéhéneuc. Can Microservice-Based Online-Retailers be Used as an SPL? In *International System and Software Product Line Conference (SPLC)*, Dec. 2020.

[22] S. Lazreg, M. Cordy, P. Collet, P. Heymans, and **S. Mosser**. Multifaceted Automated Analyses for Variability-Intensive Embedded Systems. In *41st ACM/IEEE International Conference on Software Engineering*, ICSE, May 2019.

[23] S. Bonnieux, **S. Mosser**, B.-F. Mireille, Y. Hello, and G. Nolet. Model-driven Programming of Autonomous Floats for Multidisciplinary Monitoring of the Oceans. In *IEEE Oceanic Engineering Society & Marine Technology Society*, OCEANS, June 2019.

[24] S. Lazreg, P. Collet, and **S. Mosser**. Assessing the Functional Feasibility of Variability-Intensive Data Flow-Oriented Systems. In *Symposium on Applied Computing (**Best Paper Award**)*, Pau, France, Apr. 2018.

[25] B. Benni, **S. Mosser**, N. Moha, and M. Riveill. A Delta-oriented Approach to Support the Safe Reuse of Black-box Code Rewriters. In *17th International Conference on Software Reuse (ICSR'18)*, Madrid, France, May 2018.

[26] B. Benni, **S. Mosser**, P. Collet, and M. Riveill. Supporting Micro-services Deployment in a Safer Way: a Static Analysis and Automated Rewriting Approach. In *Symposium on applied Computing*, Pau, France, Apr. 2018.

[27] **S. Mosser** and J.-M. Bruel. Reconciling Requirements and Continuous Integration in an Agile Context (tutorial). In *International Requirements Engineering Conference*, RE, Aug. 2018.

[28] F. Fouquet, T. Hartmann, **S. Mosser**, and M. Cordy. Enabling lock-free concurrent workers over temporal graphs composed of multiple time-series. In *Symposium on Applied Computing*, volume 8, Pau, France, Apr. 2018.

[29] C. Cecchinel, **S. Mosser**, and P. Collet. Towards a (de)composable workflow architecture to define data collection policies. In ACM, editor, *Symposium on Applied Computing (SAC 2016)*, Pisa, Italy, Apr. 2016.

[30] C. Cecchinel, **S. Mosser**, and P. Collet. Automated Deployment of Data Collection Policies over Heterogeneous Shared Sensing Infrastructures. In *23rd Asia-Pacific Software Engineering Conference*, Hamilton, New Zealand, Dec. 2016.

[31] C. Cecchinel, **S. Mosser**, and P. Collet. Software Development Support for Shared Sensing Infrastructures: A Generative and Dynamic Approach. In *International Conference on Software Reuse (ICSR'15)*, Miami, United States, Jan. 2015. Springer.

[32] S. Urli, M. Blay-Fornarino, P. Collet, **S. Mosser**, and M. Riveill. Managing a Software Ecosystem Using a Multiple Software Product Line: a Case Study on Digital Signage Systems. In *Euromicro Conference series on Software Engineering and Advanced Applications(SEAA'14)*, Special issue: Software Product Lines and Software Ecosystems, pages 1–8, Verona, Italy, Aug. 2014. Elsevier.

[33] I. Logre, **S. Mosser**, P. Collet, and M. Riveill. Sensor Data Visualisation: A Composition-Based Approach to Support Domain Variability. In *European Conference on Modelling Foundations and Applications (ECMFA 2014)*, volume 8569, pages 101–116, York, United Kingdom, July 2014. Springer.

[34] A. Feugas, **S. Mosser**, and L. Duchien. A Causal Model to predict the Effect of Business Process Evolution on Quality of Service. In *Conference on the Quality of Software Architectures (QoSA)*, pages 143–152, Vancouver, Canada, June 2013. ACM.

[35] E. Brandtzæg, P. Mohagheghi, and **S. Mosser**. Towards a Domain-Specific Language to Deploy Applications in the Clouds. In *In 3rd International Conference on Cloud Computing, GRIDs, and Virtualization*, pages 213–218, 2012.

[36] **S. Mosser**, M. Blay-Fornarino, and L. Duchien. A Commutative Model Composition Operator to Support Software Adaptation. In A. Vallecillo, J.-P. Tolvanen, E. Kindler, H. Störrle, and D. Kolovos,

editors, *Modelling Foundations and Applications*, pages 4–19, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[37] C. A. Parra, D. Romero, **S. Mosser**, R. Rouvoy, L. Duchien, and L. Seinturier. Using Constraint-based Optimization and Variability to Support Continuous Self-Adaptation. In *27th ACM Symposium on Applied Computing (SAC'12), 7th Dependable and Adaptive Distributed Systems (DADS) Track*, pages 486–491, Trento, Italy, Mar. 2012.

[38] V. Aranega, A. Etien, and **S. Mosser**. Using Feature Model to Build Model Transformation Chains. In R. B. France, J. Kazmeier, R. Breu, and C. Atkinson, editors, *Model Driven Engineering Languages and Systems*, pages 562–578, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[39] F. D. G. Velásquez, M. Blay-Fornarino, and **S. Mosser**. Introducing Security Access Control Policies into Legacy Business Processes. In *Fifteenth International Enterprise Distributed Object Computing Conference (EDOC'11), short paper*, pages 42–49, Helsinki, Finland, Aug. 2011. IEEE.

[40] **S. Mosser**, G. Mussbacher, M. Blay-Fornarino, and D. Amyot. From Aspect-oriented Requirements Models to Aspect-oriented Business Process Design Models. In *10th international conference on Aspect Oriented Software Development (AOSD'11)*, pages 1–12, Porto de Galinhas, Brazil, Mar. 2011. ACM.

[41] **S. Mosser**, G. Hermosillo, A.-F. Le Meur, L. Seinturier, and L. Duchien. Undoing Event-Driven Adaptation of Business Processes. In *8th IEEE International Conference on Services Computing (SCC'11)*, pages 234–241, Washington DC, United States, July 2011. IEEE.

[42] M. Clavreul, **S. Mosser**, M. Blay-Fornarino, and R. B. France. Service-Oriented Architecture Modeling: Bridging the Gap between Structure and Behavior. In J. Whittle, T. Clark, and T. Kühne, editors, *Model Driven Engineering Languages and Systems (MODELS'11)*, volume 6981 of *Lecture Notes in Computer Science*, pages 289–303, Wellington, New Zealand, Oct. 2011. Springer Berlin / Heidelberg.

[43] M. Alférez, N. Amalio, S. Ciraci, F. Fleurey, J. Kienzle, J. Klein, M. Kramer, **S. Mosser**, G. Mussbacher, E. Roubstova, and G. Zhang. Aspect-Oriented Model Development at Different Levels of Abstraction. In *7th European Conference on Modelling Foundations and Applications (ECMFA'11)*, pages 1–16, Birmingham, United Kingdom, June 2011. Springer LNCS.

[44] **S. Mosser**, A. Bergel, and M. Blay-Fornarino. Visualizing and Assessing a Compositional Approach of Business Process Design. In *Software Composition 2010*, page Springer's Lecture Notes in Computer Science, Malaga, Spain, June 2010. ACM SIGPLAN and SIGSOFT.

[45] **S. Mosser**, M. Blay-Fornarino, and J. Montagnat. Orchestration Evolution Following Dataflow Concepts: Introducing Unanticipated Loops Inside a Legacy Workflow. In *International Conference on Internet and Web Applications and Services (ICIW)*, pages 1–6, Venice, Italy, May 2009. IEEE Computer Society.

[46] **S. Mosser**, F. Chauvel, M. Blay-Fornarino, and M. Riveill. Web Service Composition: Mashups Driven Orchestration Definition. In *International Conference on Itelligent Agents, Web Technologies and Internet Commerce (IAWTIC'08)*, pages 1–6, Vienna, Austria, Dec. 2008. IEEE Computer Society.

[47] **S. Mosser**, M. Blay-Fornarino, and M. Riveill. Web Services Orchestration Evolution : A Merge Process For Behavioral Evolution. In *2nd European Conference on Software Architecture(ECSA'08)*, pages 1–16, Paphos, Cyprus, Sept. 2008. Springer LNCS.

International Workshop Papers

[48] B. Benni, P. Collet, G. Molines, **S. Mosser**, and A.-M. Pinna-Dery. Teaching DevOps at the Graduate Level, a report from Polytech Nice Sophia (short paper). In *First International Workshop on Software Engineering Aspects of Continuous Development and New Paradigms of Software Production and Deployment*, Villebrumier, France, Mar. 2018. LASER Foundation, Springer.

[49] L. Gonnord and **S. Mosser**. Practicing Domain-Specific Languages: From Code to Models. In *14th Educators Symposium at MODELS 2018*, Oct. 2018.

[50] F. Ciccozzi, M. Famelis, G. Kappel, L. Lambers, **S. Mosser**, R. F. Paige, A. Pierantonio, A. Rensink, R. Salay, G. Taentzer, A. Vallecillo, and M. Wimmer. How do we teach Modelling and Model-Driven Engineering? A survey. In *14th Educators Symposium at MODELS 2018*, Oct. 2018.

[51] F. Ciccozzi, M. Famelis, G. Kappel, L. Lambers, **S. Mosser**, R. Paige, A. Pierantonio, A. Rensink, R. Salay, G. Taentzer, A. Vallecillo, and M. Wimmer. Towards a Body of Knowledge for Model-Based Software Engineering. In *14th Educators Symposium at MODELS 2018*, Oct. 2018.

[52] M. Blay-Fornarino, G. Jungbluth, and **S. Mosser**. Applying DevOps to Machine Learning, ROCK-Flows, a Story from the Trenches (short paper). In *First International Workshop on Software*

Engineering Aspects of Continuous Development and New Paradigms of Software Production and Deployment, Villebrumier, France, Mar. 2018. LASER Foundation, Springer.

[53] C. Cecchinel, M. Jimenez, **S. Mosser**, and M. Riveill. An Architecture to Support the Collection of Big Data in the Internet of Things. In *International Workshop on Ubiquitous Mobile cloud (co-located with SERVICES)*, Anchorage, United States, June 2014.

[54] **S. Mosser**, P. Collet, and M. Blay-Fornarino. Exploiting the internet of things to teach domain-specific languages and modeling: The arduinoml project. In *EduSymp@MoDELS*, 2014.

[55] P. Collet, **S. Mosser**, S. Urli, M. Blay-Fornarino, and P. Lahire. Experiences in Teaching Variability Modeling and Model-driven Generative Techniques. In *Proceedings of the 18th International Software Product Line Conference: Companion Volume for Workshops, Demonstrations and Tools - Volume 2*, SPLC '14, pages 26–29, New York, NY, USA, 2014. ACM.

[56] S. Urli, **S. Mosser**, M. Blay-Fornarino, and P. Collet. How to Exploit Domain Knowledge in Multiple Software Product Lines? In *Fourth International Workshop on Product LinE Approaches in Software Engineering at ICSE 2013 (PLEASE 2013)*, page 4 p., San Fransisco, United States, May 2013. ACM.

[57] **S. Mosser**, I. Logre, N. Ferry, and P. Collet. From Sensors to Visualization Dashboards: Need for Language Composition. In *Globalization of Modeling Languages workshop (GeMOC'13)*, Miami, United States, Sept. 2013.

[58] D. Romero, S. Urli, C. Quinton, M. Blay-Fornarino, P. Collet, L. Duchien, and **S. Mosser**. SPLEMMA: A Generic Framework for Controlled-Evolution of Software Product Lines. In *MAPLE/SCALE 2013*, volume 2, pages 59–66, Tokyo, Japan, Aug. 2013.

[59] B. Combemale, J. DeAntoni, R. B. France, F. Boulanger, **S. Mosser**, M. Pantel, B. Rumpe, R. Salay, and M. Schindler. Report on the First Workshop On the Globalization of Modeling Languages. *CoRR*, abs/1408.5703, 2013, 1408.5703.

[60] S. Urli, M. Blay-Fornarino, P. Collet, and **S. Mosser**. Using Composite Feature Models to Support Agile Software Product Line Evolution. In *International Workshop on Models and Evolution in MODELS Conference*, pages 1–6, Innsbruck, Austria, Sept. 2012.

[61] E. Brandtzæg, **S. Mosser**, and P. Mohagheghi. Towards CloudML, a Model-based Approach to Provision Resources in the Clouds. In *International Workshop on Cloud and MDE (co-loacted with ECNFA Conference)*, pages 1–6, 2012.

[62] **S. Mosser**, F. Fleurey, B. Morin, F. Chauvel, A. Solberg, and I. Goutier. SENSAPP As a Reference Platform to Support Cloud Experiments: From the Internet of Things to the Internet of Services. In *Proceedings of the 2012 14th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing*, SYNASC '12, pages 400–406, Washington, DC, USA, 2012. IEEE Computer Society.

[63] **S. Mosser**, L. Duchien, C. A. Parra, and M. Blay-Fornarino. Using Domain Features to Handle Feature Interactions. In A. C. P. Series, editor, *Variability Modelling Software-Intensive Systems (VAMOS)*, pages 101–110, Leipzig, Germany, Jan. 2012. Ulrich Eisenecker, University of Leipzig, DE.

[64] D. Ardagna, E. Di Nitto, G. Casale, D. Petcu, P. Mohagheghi, **S. Mosser**, P. Matthews, A. Gericke, C. Ballagny, F. D'Andria, C.-S. Nechifor, and C. Sheridan. MODAClouds: A Model-driven Approach for the Design and Execution of Applications on Multiple Clouds. In *Proceedings of the 4th International Workshop on Modeling in Software Engineering*, MiSE '12, pages 50–56, Piscataway, NJ, USA, 2012. IEEE Press.

[65] C. Quinton, **S. Mosser**, C. Parra, and L. Duchien. Using Multiple Feature Models to Design Applications for Mobile Phones. In *MAPLE / SCALE workshop, colocated with SPLC'11*, pages 1–8, Munich, Germany, Aug. 2011.

[66] **S. Mosser**, M. Blay-Fornarino, and M. Riveill. Service Oriented Architecture Definition Using Composition of Business-Driven Fragments. In *Models and Evolution (MODSE'09), MODELS'09 workshop*, pages 1–10, Denver, Colorado, United States, Oct. 2009.

[67] **S. Mosser**. Are Functional Languages a good way to represent productive meta-models ? In *4th European Lisp Workshop (ELW'07)*, pages 1–6, Berlin, Germany, France, July 2007.

Posters & Misc

[68] I. Logre, **S. Mosser**, and M. Riveill. Composition Challenges for Sensor Data Visualization (poster). In *International Conference on Modularity (MODULARITY 2015)*, Fort Collins, United States, Mar. 2015.

[69] **S. Mosser**. La Thèse ... (Seminar to new PhD Students). Feb. 2012.

National Journal Papers

[70] M. Blay-Fornarino, V. Hourdin, C. Joffroy, S. Lavirotte, **S. Mosser**, A.-M. Pinna Déry, P. Renevier, M. Riveill, and J.-Y. Tigli. Architecture pour l'adaptation de Systèmes d'Information Interactifs Orientés Services. *Revue des Sciences et Technologies de l'Information - Série L'Objet : logiciel, bases de données, réseaux*, pages 93–118, 2007.

National Conference & Workshop Papers

[71] F. Chauvel, **S. Mosser**, and A. Solberg. Reconsidering QoS Analysis in Dynamic and Open Systems. In *1ère conférence en ingénierie du logiciel(CIEL'12), short paper*, , Rennes, June 2012.

[72] A. Feugas, **S. Mosser**, A.-F. Le Meur, and L. Duchien. Déterminer l'impact d'une évolution dans les processus métiers. In *Journées sur l'Ingénierie Dirigée par les Modèles (IDM'11)*, pages 71–76, Lille, France, June 2011.

[73] C. Brel and **S. Mosser**. Vers une approche flot de données pour supporter la composition d'interfaces homme-machine. In *Journées sur l'Ingénierie Dirigée par les Modèles(IDM'11)*, pages 1–6, Lille, France, June 2011. CNRS.

[74] **S. Mosser** and M. Blay-Fornarino. Taming Orchestration Design Complexity through the ADORE Framework. In *Journées 2010 du GDR GPL, CNRS*, Pau, France, Mar. 2010.

[75] **S. Mosser** and M. Blay-Fornarino. Réflexions autour de la construction dirigée par les modèles d'un atelier de composition d'orchestrations. In *15ème conférence francophone sur les Langages et Modèles à Objets (LMO'09)*, pages 1–16, Nancy, France, Mar. 2009. Cépadues.

[76] **S. Mosser**, M. Blay-Fornarino, and M. Riveill. Un modèle d'évolution multi-vues des Architectures Orientées Services. In *Actes de l'Atelier Doctorant LMO'08(DOC LMO'08), workshop*, , page 6, Montréal, Mar. 2008. Université de Montréal -.

[77] **S. Mosser**, M. Blay-Fornarino, P. Collet, and P. Lahire. Vers l'intégration dynamique de contrats dans des architectures orientées services : une experience applicative du modèle au code. In *2ème Conférence sur les Architectures Logicielles (CAL'08)*, pages 1–15, Montréal, Canada, Mar. 2008.

[78] **S. Mosser**, M. Blay-Fornarino, and M. Riveill. Orchestrations de Services Web : Vers une évolution par composition. In *Atelier RIMEL (Rétro-Ingénierie, Maintenance et Evolution des Logiciels)*, , page 6, Toulouse, France, Mar. 2007. Dalila Tamzalit, Salah Sadou.

[79] C. Joffroy, **S. Mosser**, M. Blay-Fornarino, and C. Nemo. Des Orchestrations de Services Web aux Aspects. In U. d. T. EMN, INRIA, editor, *3ème Journée Francophone sur le Développement de Logiciels Par Aspects (JFLDPA'2007)*, pages 1–13, Toulouse, France, Mar. 2007.