Contributions à la production de logiciels de qualité

Olivier Zendra

Contributions à la production de logiciels de qualité

Habilitation à diriger des recherches

DE

Olivier ZENDRA

Université de Rennes.

Ecole doctorale MathSTIC.

Spécialité Informatique, Section 27.

Habilitation présentée et soutenue à Rennes le 20 décembre 2023.

Jury :

Rapporteurs :

Laure GONNORD	Professeure, GrenobleINP/UGA, France.
Marianne HUCHARD	Professeure, Université de Montpellier, France.
Jan VITEK	Professeur, Northeastern University, USA.

Examinateurs :

Olivier FESTOR	Professeur, Directeur de Lorraine INP, Université de Lorraine, France
Caroline FONTAINE	DR CNRS, ENS Cachan, France.
Jean-Marc JÉZÉQUEL	Professeur, Université de Rennes, France

Remerciements

Je remercie très sincèrement les membres de mon jury de m'avoir fait l'honneur d'accepter cette charge malgré leurs emplois du temps chargés, et pour le travail qu'elles et ils ont fourni pour cela, tout particulièrement les rapportrices et rapporteur.

Je remercie de tout coeur toutes les bonnes personnes avec qui j'ai eu la chance de travailler durant environ un quart de siècle de recherche, qui m'ont soutenu, et dont certaines sont devenues des amies.

Je remercie mes parents, pour leur support, qui a permis ma trajectoire, et sans qui je ne serais rien.

Je remercie ma famille, pour leur compréhension quand, trop souvent, je n'étais pas disponible pour cause professionnelle.

Avant-propos

Conformément aux consignes locales, qui préconisent pour les Habilitations à Diriger des Recherche (HDR) des documents de 30 à 40 pages, cette HDR est courte, sélective et surtout synthétique, bien que portant sur mon activité en tant que chercheur depuis environ un quart de siècle.

Conformément à la raison d'être de l'HDR, et à son nom, ce document n'est en aucun cas une "thèse de doctorat bis", mais au contraire explique les directions (axes) prises par mes recherches et met en avant les aspects liés à la direction (diriger) des recherches, les miennes et celles des personnes qui ont travaillé avec moi, en accordant leur juste place aux aspects managériaux et humains.

Version compilée le : 2024-01-22 09:52:06+01:00.

Plan

Chapitre 1

Synthèse de mon activité de formation

Informatique — 69 heures ETD — 1997-1998 — Public : D.E.U.G. SNV, UHP-Nancy 1. **Objectif :** Initier des débutants à l'informatique.

Algorithmique, programmation et structures de données (APSD) — 27 heures ETD — 1997-1999 — Public : Licence d'Informatique, UHP-Nancy 1. **Objectif :** Familiariser les étudiants avec la notion de type, les principales structures de données et les concepts de la programmation à objets (classes, instances, assertions, généricité, héritage, liaison dynamique). Des notions de base du génie logiciel (modularité, documentation, tests) étaient également mises en avant.

Informatique de Base (IB) — 118 heures ETD — 1997-2003 — Public : 1ère année de l'ESIAL. **Objectif :** Les principales notions abordées concernaient les types, les principales structures de données et les algorithmes de tri, ainsi que les concepts de la programmation à objets (classes, instances, assertions, généricité, héritage, liaison dynamique). L'accent était également mis sur des notions de base du génie logiciel (modularité, documentation, tests).

Algorithmique et programmation (AP) — 26 heures ETD — 1998-1999 — Public : 1ère année Informatique de l'ESIAL. **Objectif :** Aborder de façon plus poussée les concepts et techniques vus en IB, y ajouter d'autres notions importantes, telles que par exemple les types abstraits algébriques, la complexité algorithmique, les exceptions et la gestion de la mémoire.

Conception et développement en Java — 93 heures ETD — 1999-2000 — Publics : a) DESS IL et DESS IRS, UHP-Nancy 1. **b)** 2ème année de l'IUP RNC, UHP-Nancy 1. **c)** 2ème année de l'ESIAL. **Objectif :** Rendre les étudiants familiers avec les concepts fondamentaux des langages à objets (classe, objet, encapsulation, héritage, polymorphisme...) et le langage Java. Les mécanismes de base ont été étudiés, suivis par les aspects liés au multi-tâches, à la construction d'interfaces graphiques et la programmation d'applets.

School of Computer Science, McGill University, Montréal, Canada — 15 heures ETD — 2000-2001 — Public : *Master in Computer Science*. **Contexte :** Enseignements ponctuels portant sur mes domaines d'expertise personnels (gestion mémoire automatique, langages à objets, liaison dynamique et optimisation), dans le cadre des modules *CS308-303 - Advanced Programming Techniques*, *CS308-505 - High Performance Computer Architecture* et *CS308-764 : Run-time Support for Object-Oriented Programming Languages*.

Algorithmique et Langages à Objets (Algo) — 123 heures ETD — 2002-2004 — Public : 1ère année d'IUP GMI, UHP-Nancy 1. **Objectif :** Rendre les étudiants familiers avec les notions de base de l'algorithmique, ainsi que les concepts fondamentaux des langages à objets (classe, objet, encapsulation, héritage, polymorphisme...) et le langage Java.

Algorithmique, programmation et structures de données (APSD) — 34 heures ETD — 2003-2004 — Public : 2ème année d'IUP GMI, UHP-Nancy 1. **Objectif :** Suite du module Algo, permet de

présenter aux étudiants des notions de base du génie logiciel (spécification, contrats, preuves, motifs) et de les mettre en pratique à l'aide du langage Java.

Java — 3 heures ETD — 2004-2005 — Public : 1ère année École des Mines de Nancy, **Objectif :** Initiation à Java.

Gestion mémoire et basse énergie dans les systèmes embarqués — 14 heures ETD — 2005-2007 — Public : 2ème année du Master IS-EEAPR, UHP -Nancy 1 et 2ème année École des Mines de Nancy, **Objectif :** Concepts de base pour la gestion mémoire et la basse énergie dans les systèmes embarqués et temps-réel.

Projet de fin d'année — 12 heures ETD — 2007 — Public : Module ISA-SERTR, 3ème année de l'ENSEM, Nancy, **Objectif :** Intégration de l'outil Wattch d'analyse d'énergie à une plateforme expérimentale de recherche.

Mémoire — 6H CM — 2007-2008 — Public : 2ème année École des Mines de Nancy , **Objectif :** Gestion mémoire et basse consommation énergétique.

Mémoire et basse énergie — 9H CM — 2007-2008 — Public : Master CSSEA (Conception Sûre des Systèmes Embarqués et Ambiants), École des Mines de Nancy, **Objectif :** Gestion mémoire et basse consommation énergétique.

Basse énergie — 6H CM — 2007-2008 — Public : Système et Applications Réparties en 3ème année d'ENSEM, Nancy. **Objectif :** Gestion mémoire et basse consommation énergétique.

Mémoire et basse énergie — 9H CM — 2008-2009 — Public : Master CSSEA (Conception Sûre des Systèmes Embarqués et Ambiants), École des Mines de Nancy, **Objectif :** Gestion mémoire et basse consommation énergétique.

Java — 15H CM + 30H TD — 2008-2009 — Public : Formation continue (CUCES) de chômeurs en reconversion à l'Université de Nancy 2, UFR Math Info **Objectif :** concepts des langages à objets et programmation en Java.

Java — 15H CM + 30H TD — 2009-2010 — Public : Formation continue (CUCES) de chômeurs en reconversion à l'Université de Nancy 2, UFR Math Info **Objectif :** concepts des langages à objets et programmation en Java.

Mémoire — 6H CM — 2010-2011 — Public : 2ème année École des Mines de Nancy, **Objectif :** Gestion mémoire et basse consommation énergétique.

Java — 15H CM + 30H TD — 2011-2012 — Public : Formation continue (CUCES) de chômeurs en reconversion à l'Université de Lorraine, Nancy 2, UFR Math Info **Objectif :** Concepts des langages à objets et programmation en Java.

Chapitre 2

Synthèse de mon activité de recherche et de direction de la recherche

2.1 Introduction, motivation et chronologie

Mon domaine de recherche porte très globalement sur la production de logiciels, via les méthodes et outils de compilation, de build et de génie logiciel, et leur optimisation.

Il s'agit là d'un domaine crucial, puisque le logiciel est depuis de nombreuses années omniprésent dans nos sociétés, et l'est de plus en plus fortement. Les systèmes informatiques sont innombrables, dans tous les domaines du quotidien, des plus gros aux plus petits (embarqués, IoT...), des plus visibles aux plus discrets/enfouis. L'impact de la qualité (ou de la non-qualité...) des logiciels est donc considérable, pour des milliards d'utilisateurs, quasiment en permanence. Cet impact peu être modéré (qualité de service), ou vital, dans certains systèmes critiques, qui sont eux aussi de plus en plus nombreux.

La qualité de la production logicielle est donc un élément majeur, qui a animé l'ensemble de mes travaux de recherche depuis 1995-1997. Cependant ce problème est difficile, car les objectifs à atteindre sont nombreux : fiabilité fonctionnelle, avant tout, bien sûr, mais aussi les aspects dits non-fonctionnels, comme la vitesse, l'occupation mémoire, la consommation énergétique, la sécurité dans toutes ses facettes, etc. Les facteurs à prendre en compte pour atteindre ces objectifs sont encore plus variés et complexes : l'expressivité au niveau du langage ou des outils (pour une utilisabilité pratique), les algorithmes implantés bien sur, mais l'influence sur l'exécution du programme compilé de l'environnement d'exécution (architecture matérielle, système d'exploitation, machine virtuelle le cas échéant...), etc.

Ce sont précisément ces objectifs et ces facteurs qui ont guidés mes travaux au cours de ma carrière de chercheur, ce qui me permet d'esquisser maintenant leur **chronologie approximative** :

Après mon stage de DEA en apprentissage machine (cf. section 2.2) en 1995, j'ai en effet commencé par travailler sur la compilation des langages à objets (cf. 2.3) vers 1995, d'abord sur mon temps libre. J'ai démarré par le très beau langage Eiffel, pour lequel je me suis penché durant ma thèse (cf. 2.3.1) en premier lieu sur la correction de la compilation, puis sur les problèmes de performance, typiquement en vitesse puis en mémoire, qui ont naturellement suivi, étant souvent des facteurs limitants ou des préoccupations des utilisateurs.

J'ai ensuite décidé d'étendre mes travaux lors de mon post-doctorat en 2001 (cf. 2.3.2) à un langage plus utilisé qu'Eiffel, à savoir Java, en croissance exponentielle à cette époque. J'ai continué ces travaux en 2002 après mon postdoctorat en explorant les effets du matériel sur l'exécution de programmes Java (cf. 2.3.3),

car cette interface matériel-logiciel m'a toujours intéressé [1] et je pense que là est un gisement important d'efficacité et d'économie dans nos domaines.

J'ai également en 2001-2004 étendu mes travaux sur la gestion automatique de la mémoire, passant du nouveau ramasse-miettes pour Eiffel que j'avais fait pendant ma thèse à un nouveau ramasse-miette générationnel pour une bibliothèque fonctionnelle, celle des ATerms (cf. 2.3.4).

En plus de ces travaux sur la compilation et la mémoire, qui m'ont apporté une meilleure compréhension sur la mise en oeuvre technique concrète des langages, je suis revenu à l'étape amont, c'est à dire que je me suis intéressé à la conception des langages eux-mêmes et à leur expressivité. J'ai ainsi contribué sur 2001-2004 à plusieurs propositions dans ce domaine, pour Eiffel (cf. 2.3.5) et pour créer de nouveaux langages (cf. 2.3.6 et 2.3.7).

Je me suis focalisé de nouveau sur des aspects d'implantation, mais en m'intéressant cette fois tout spécifiquement à l'objectif de basse consommation énergétique (cf. 2.4) entre 2004 et 2016 [2]. J'ai donc dirigé des travaux l'abordant les économies d'énergie sous des angles complémentaires : l'ordonnancement (cf. 2.4.1), le placement en mémoire (cf. 2.4.2), les plateformes de support expérimental (cf. 2.4.3), les (protocoles de) réseaux de capteurs sans fil (cf. 2.4.4) [3].

En parallèle à ces travaux, je me suis rendu compte que les outils de génie logiciel disponibles ne me fournissaient pas une visualisation satisfaisante de ce que je voulais examiner dans les logiciels, notamment leur exécution. J'ai donc mené des travaux en 2008-2012 visant à faire avancer ces outils en m'intéressant aux visualisation avancée, non textuelles, des logiciels et de leur exécution. Pour cela j'ai d'abord dirigé des travaux sur l'instrumentation de l'exécution de programmes Java (cf. 2.3.8), puis sur la visualisation couplée du code du logiciel et de son exécution (cf. 2.5.).

Après m'être penché sur les objectifs de vitesse, mémoire, énergie, j'ai à partir de 2016 orienté mon attention sur l'objectif de sécurité dans les systèmes informatiques (cf. 2.6). La cybersécurité étant un très vaste domaine, j'ai donc dans ce virage thématique dirigé en parallèle des travaux portant sur plusieurs axes complémentaires.

Le premier axe de 2017 à 2020 a été l'analyse du protocole de sécurité IKEv2 (cf. 2.6.1).

Le second axe, lié à la conception de langages que j'avais abordée précédemment, porte depuis 2017 sur le problème crucial à mon avis de l'intégration explicite des propriétés de sécurité dans les logiciels (cf. 2.6.2).

Le troisième axe a porté de 2017 et 2022 sur l'analyse de binaires (que j'avais déjà abordée sur du bytecode Java, comme expliqué ci-dessus), mais cette fois pour détecter des maliciels (cf. 2.6.3), détecter et classifier des packers de maliciels (cf. 2.6.4), et classifier des maliciels (cf. 2.6.5).

Enfin, j'ai dirigé depuis 2018 des travaux portant sur les problèmes de sécurité liés à l'interface des mondes matériel et logiciel, et plus spécifiquement sur la défense contre des attaques par canaux auxiliaires, aussi bien sur le plan pratique que plus théorique (cf. 2.6.6).

Depuis 2021, j'ai pris part au co-encadrement de travaux (cf. 2.3.9) portant sur l'analyse du processus de build de variants logiciels, abordant l'exploration d'options de configuration, l'efficacité de la compilation en lien avec les mécanismes de cache de compilation et les stratégies incrémentales. Je considère ce "retour aux sources" pour moi en quelques sorte comme une première étape vers les travaux pour la sécurisation des chaines de build que je souhaite mener (cf. section 3.3).

Tout en menant ces travaux de recherche, j'ai pris part depuis 2016 de façon croissante aux charges spécifiquement liées à la direction de la recherche, en prenant la direction d'une équipe de cybersécurité (cf. 2.7), en montant et coordonnant le projet européen TeamPlay (cf. 2.8), et en contribuant aux réseaux scientifiques et à la prospective scientifique (cf. 2.9).

A ce moment-là, il m'est apparu qu'il était en 2023 plus que judicieux et logique que je capitalise sur tout cela et fasse le point en passant enfin mon Habilitation à Diriger les Recherches. D'où le présent document.

1. Mes planches d'audition de CR Inria à Nancy en 2000 mentionnaient déjà la co-conception matérielle-logicielle, sans utiliser ce terme proprement dit.
2. Une époque où les économies d'énergie n'étaient pas encore très à la mode. Les choses ont bien changé depuis.
3. A une époque où l'IoT - Internet of Things, n'était pas au devant de la scène. Là aussi les choses ont changé.

Les travaux qui sont présentés dans le reste de cette section le sont, comme je l'ai mentionné plus haut, dans un ordre chronologique non-strict, de nombreux recouvrements temporels existant.
Les travaux les plus anciens sont présentés plus succinctement.
Les travaux encore actifs, ie. qui se poursuivent actuellement, sont indiqués par ⇒ à la fin du titre de la section concernée.

2.2 Architectures connexionnistes et robotique.

Mots-clefs : IA, robotique mobile, navigation, évitement d'obstacles, connexionnisme, perceptron, apprentissage
Période : 1995

Mes premières recherches ont eu lieu en 1995 dans cadre de mon stage de D.E.A. (aujourd'hui on appellerait cela Master 2 Recherche) au sein de l'équipe RFIA du CRIN, sous la responsabilité de M. Pascal Blanchet, Maître de Conférences à l'Université de Nancy 2. Elles ont consisté à étudier l'application d'architectures connexionnistes simples pour l'apprentissage de l'évitement d'obstacles par un robot mobile. Il s'agissait là de navigation réactive, en couche basse, la planification ayant lieu à niveau plus élevé. Ce travail a eu pour résultats la réalisation de prototypes "GRP" (Guidage de Robot par Perceptron) écrits en C/C++, fonctionnant à la foi sur un simulateur de robot et sur le robot lui-même. J'ai rédigé un rapport de D.E.A. [Zen1995].

2.3 Langages, compilation, exécution de programmes ⇒

Mots-clefs : compilation, optimisation, classes, objets, spécialisation, liaison dynamique, gestion mémoire, Eiffel, SmallEiffel, SmartEiffel, prototypes ,instrumentation, Java, build, Linux
Période : 1997 ⇒

Une grande partie de mon activité de recherche, qui est aussi mon activité de recherche initiale (si on excepte mon DEA), et de direction de la recherche, située globalement entre les années 1997 et 2013, a porté sur plusieurs langages de programmation à objets, leur compilation et l'optimisation de leur exécution pour améliorer les performances en termes de vitesse et/ou de mémoire, ainsi que l'instrumentation dynamique (cf. sections 2.3.1 à 2.3.8). Mon activité sur ces domaines, jamais complètement arrêtée, redémarre plus fortement depuis 2021 avec les travaux autour du processus de build (cf. section 2.3.9).

2.3.1 Traduction et optimisation dans les langages à base de classes

Mots-clefs : langages à objets, traduction, compilation, inférence de types, analyse statique, contrôle de cohérence, optimisation, liaison dynamique, gestion mémoire, Eiffel.
Période : 1997-2000

Problématique

Ma Thèse de Doctorat s'inscrivait dans le cadre des recherches menées autour de la compilation des langages à objets, notamment Eiffel, et plus généralement des langages à typage statique. Les problèmes abordés touchaient à la traduction des langages à objets, l'analyse statique des programmes, l'implantation efficace de la liaison dynamique, l'utilisation de techniques de prédiction de type et d'évaluation partielle, la persistance des objets, la gestion de la mémoire et les ramasse-miettes ainsi que la validation.
De façon un peu lapidaire, on peut dire que le but de ce travail revenait à tenter de répondre à une question fondamentale : comment mieux compiler les langages à objets, c'est à dire comment obtenir des programmes plus rapides et plus sûrs ?

Approche

Le travail de recherche de ma thèse était basé en grande partie sur l'analyse statique, abordée via deux axes principaux.
Le premier consistait à pouvoir effectuer des contrôles de validité et de cohérence du programme, et ce non seulement sur les programmes finis, mais bien dès le début du développement, de façon à pouvoir assister les développeurs au maximum durant la phase de conception et d'implantation.
Le second considérait l'utilisation des informations apportées par l'analyse statique du système pour améliorer la qualité du code généré. En effet, ces informations offrent des possibilités importantes en terme d'optimisation du code généré, aussi bien par des optimisations liées aux algorithmes que par des optimisations sur les structures de données.
Ces recherches, bien que plus particulièrement validées sur le langage Eiffel, devaient bien entendu être applicables en général à la plupart des langages à base de classes (Java, C++ ...).

Résultats

J'ai proposé et expérimenté une approche basée sur la duplication et la spécialisation du code par analyse globale du système afin d'implanter de façon efficace la liaison dynamique et les structures de données du programme compilé [CCZ1997 ; ZCC1997 ; CZ1999]. J'ai ainsi introduit une méthode de liaison dynamique nouvelle et élégante, basée sur des arbres de branchement directs, dont les performances sont supérieures ou égales à celles des systèmes actuels classiques à base de tables d'indirection.

Cette approche a été étendue, à la génération, lors de la compilation, d'un ramasse-miettes [CCZ1998a ; CZC1998] et d'une gestion des exceptions automatiquement adaptés à l'application compilée.

J'ai également mené certaines études pour évaluer les optimisations permises par l'utilisation massive de l'aliasing dans un programme à objets, ainsi que les moyens de bien maîtriser cette technique [ZC1999b ; ZC2000 ; ZC2001].

Ces travaux ont été validés par le développement d'un compilateur Eiffel nommé SmallEiffel et de ses bibliothèques [ZC1999a], sur lequel j'avais d'ailleurs commencé à travailler dès 1994 sur mon temps libre. SmallEiffel a été mis à disposition du public et a constitué le premier et le seul compilateur Eiffel complet, entièrement gratuit, et au code source en libre accès. Ceci, ainsi que les performances de SmallEiffel, lui a valu d'être reconnu par la Free Software Foundation (FSF) comme "The GNU Eiffel Compiler". C'est aussi à ma connaissance le premier logiciel GNU entièrement réalisé et activement maintenu au sein de l'INRIA. A titre indicatif, sa diffusion, via Internet et les CDs des distributions Linux notamment, pouvait être estimée à au moins 200 000 exemplaires en 1998, 1 million d'exemplaires en 1999 et plusieurs millions chaque année après l'an 2000. Le nombre d'utilisateurs réels, bien que difficile à évaluer, était de l'ordre de plusieurs milliers. Sa visibilité internationale était donc considérable. Notons que SmallEiffel a été renommé SmartEiffel en septembre 2002.

Ces résultats ont également été la base d'un gros effort de vulgarisation au moyen d'articles dans des revues techniques moyen-grand public [CCZ1998b ; ZCC1998 ; CZC1999c ; CZC1999b ; ZCC1999a ; CZC1999a ; ZCC1999b].

Enfin, j'ai contribué en 1999 à des efforts pour (tenter de) standardiser les bibliothèques du langage Eiffel et leur standard ELKS (Eiffel Library Kernel Standard), controlé par le NICE (The Non-Profit International Consortium for Eiffel). En effet, un premier standard, ELKS'95, était très incomplet et n'avait jamais été mis à jour. Des discussions ont eu lieu à partir de notre proposition pour ELKS'2000 ([CSZ1999] et `https://eiffel-nice.org/standards/wip/other/prior1.html`), dont nous avons pu faire adopter certains éléments.

2.3.2 Détection de régularités dans les programmes pour l'optimisation en Java

Mots-clefs : langages à objets, compilation, inférence de types, analyse statique, optimisation, liaison dynamique, Java.
Période : 2000-2002

Contexte et problématique

J'ai mené ces travaux dans le cadre d'un postdoctorat à McGill University, Montréal, Canada, au sein du groupe ACL (Adaptive Computation Laboratory) de la School of Computer Science (SOCS), dirigé par Karel Driesen, Professeur à McGill University.
Le laboratoire ACL s'intéressait aux architectures matérielles et aux systèmes informatiques adaptatifs, notamment afin d'optimiser plus efficacement les programmes. Dans ce cadre, une étude quantitative et systématique des comportements des programmes peut permettre de mieux comprendre les interférences, les possibilités de prédiction et les opportunités d'adaptation tant au niveau du matériel (processeur) que du logiciel (compilateur). De nouvelles techniques d'optimisation peuvent ainsi être développées. L'ACL étudiait donc l'ensemble de ces problèmes, se focalisant sur les techniques permettant de détecter et prendre avantage de récurrences, éventuellement temporaires, dans les flux de données.
Les nombreux types de mécanismes de prédiction, en ligne ou hors ligne, permettant la détection de régularités dans l'exécution, doivent donc être évalués et caractérisés. De plus, une fois la prédictibilité d'un flux d'information établie, il est nécessaire de disposer de méthodes spécifiques au flux pour déterminer quelles actions préemptives peuvent être réellement utiles.

Travaux et résultats

Mon travail avait pour but d'améliorer la compilation et l'implantation efficace du langage Java. Dans cette optique, deux aspects principaux ont été explorés conjointement : la prédiction et l'optimisation de la liaison dynamique, d'une part, et l'impact des interactions entre logiciel et matériel sur les performances des programmes Java, d'autre part. Parmi les systèmes de liaison dynamique étudiés dans ce contexte se trouvait naturellement le nouveau système de liaison dynamique mis au point durant ma thèse
J'ai travaillé sur la détection de régularités au sein des comportements des programmes Java, afin d'adapter l'environnement d'exécution de façon à optimiser leur exécution. J'ai donc mis en place une expérimentation multi-environnements (JVM) et multi-plateformes (système d'exploitation et machine), dont les résultats montrent [ZD2002a ; ZD2002b] qu'il est possible de faire des optimisations de la liaison dynamique au niveau du bytecode Java (après compilation) qui permettent un gain quels que soient la JVM, le système d'exploitation et la machine sur lesquels le programme s'exécutera.
Une expérimentation plus fine était cependant nécessaire afin de mieux comprendre le rôle du schéma d'exécution (pattern de types) à un point de liaison dynamique donné, ainsi que pour mieux modéliser l'impact du matériel sur les résultats.

2.3.3 Optimisation de la liaison dynamique et prise en compte de l'architecture matérielle en Java

Mots-clefs : langages à objets, compilation, types, analyse statique, optimisation, liaison dynamique, prédiction de branchement, interactions matériel-logiciel, Java, bytecode.
Période : 2001-2002

Dans la continuité des travaux que j'avais débutés durant mon postdoctorat INRIA à McGill University, j'ai mené juste après mon recrutement au sein del'INRIA-Lorraine / LORIA des recherches portant sur l'optimisation des programmes Java. En effet, un des plus gros problèmes qui se posait aux nombreux

utilisateurs du langage Java est celui de la vitesse d'exécution des programmes. Il était donc capital de fournir des solutions à ce problème.
Après avoir travaillé sur la détection de régularités au sein des comportements des programmes Java, afin d'en tirer parti pour proposer des techniques d'optimisation portables, j'ai ensuite cherché à bien caractériser les interactions matériel-logiciel, pour pouvoir les exploiter et les modéliser. En effet, même si des optimisations portables sont possibles, elles peuvent être encore sensiblement améliorées par la prise en compte de l'environnement d'exécution (JVM et/ou matériel).
Pour cela, j'ai collaboré avec le laboratoire ACL de McGill University pour développer une expérimentation visant à compléter mes premiers travaux en simulant des prédicteurs de branchement de processeurs, avec différents patterns d'exécution. Les résultats de cette expérimentation [GZD2002 ; ZD2002b] confirment que certaines structures de contrôle permettent d'optimiser la liaison dynamique d'une façon qui soit robuste aux changements de processeur.Cependant, l'impact des patterns de types s'avère considérable sur les prédicteurs, et en cas de prédiction de branchement manquée, le coût sur la liaison dynamique peut être important.Néanmoins, lorsque le nombre de types est faible, quel que soit leur pattern, les méthodes classiques de liaison dynamique en Java peuvent être avantageusement remplacées par des structures alternatives, dont la liaison dynamique à arbre de branchement binaire que j'ai proposée dans ma Thèse de Doctorat.
Il apparaissait donc nécessaire de concevoir des optimisations qui prennent en compte les patterns d'exécution, soit hors ligne (*profiling*, comme dans mes travaux), soit au sein de l'environnement d'exécution (JVM ou processeur) lui-même.

2.3.4 GC^2 : Gestion mémoire pour la bibliothèque des ATerms

Mots-clefs : réécriture, termes, ATerms, gestion mémoire, ramasse-miette, marquage-balayage, générations
Période : 2001-2004

La bibliothèque des ATerms est bien connue dans le domaine de la réécriture de termes. Elle était utilisée intensivement par diverses équipes de recherche, et des centaines d'outils étaient basés sur elle, notamment l'environnement ASF+SDF, l'environnement ELAN, `Stratego`, `JJForester`, etc. Il s'agit d'un outil développé au CWI d'Amsterdam qui offre une API pratique pour manipuler des termes, un système gérant automatiquement l'aliasing maximal (ou partage maximal) des termes, ainsi qu'un système de gestion mémoire automatique.

C'est sur ce dernier point, la gestion mémoire, qu'une partie de mes travaux recherche se sont focalisés : j'ai cherché, en collaboration avec le projet Protheo de l'INRIA-Lorraine et le CWI d'Amsterdam, à améliorer les performances du ramasse-miettes intégré à la bibliothèque. Il semblait en effet que, du fait de la pression mémoire extrême exercée par certains programmes manipulant des termes, une grande partie du temps d'exécution soit passée à gérer la mémoire. J'ai ainsi pu compléter et largement réutiliser les connaissances que j'avais acquises dans le cadre de la conception et l'implantation du ramasse-miettes de SmallEiffel [CCZ1998a].

Nous avons donc souhaité concevoir un nouvel algorithme de gestion mémoire pour les ATerms qui prenne réellement en compte les spécificités de cette bibliothèque fonctionnelle, afin d'offrir les meilleures performances possibles. La principale caractéristiques que nous avons considérée est le fait que dans cette bibliothèque, un sous-terme n'est jamais modifié après la création du terme englobant. Ceci nous a permis d'envisager une stratégie de gestion mémoire améliorée basée sur le principe des générations. La mémoire y est divisée en deux (ou plus) sous-espaces correspondant chacun à une génération ("jeunes" ou "vieux" objets). Ceci permet de tirer avantage du *principe générationnel faible* d'Ungar qui indique que la plupart des objets meurent jeunes, et que seuls quelques-uns ont une longue durée de vie. En effet, en ne collectant qu'une partie de la mémoire (typiquement, en collectant plus souvent la jeune génération que la vieille), il devient possible de diminuer les pauses dues au ramasse-miettes. Cela permet de plus d'améliorer le taux de récupération d'objets et de diminuer le temps total passé dans le ramasse-miettes. Cependant, en pratique, la plupart des systèmes générationnels ont un coût de gestion des sous-espaces (le plus souvent par recopie) qui est assez élevé, et diminue le gain de temps total.

Dans le cadre de nos travaux, nous avons décidé de créer un nouvel algorithme générationnel non basé sur la copie, contrairement à la plupart des algorithmes générationnels existants. Nous avons ainsi conçu GC^2, un algorithme générationnel à marquage-balayage, qui évite le surcoût souvent imputables aux générations. Nous avons implanté une première version de GC^2 dans la bibliothèque des ATerms, et obtenons des résultats extrêmement positifs [MZ2002 ; MZ2004], avec de bons gains en mémoire et surtout en temps (environ 20%). Cette amélioration a été intégrée dans la version standard de la bibliothèque des ATerms, et est donc immédiatement applicable, de façon totalement transparente, à tous les outils utilisant cette bibliothèque.

Ces travaux pourraient être poursuivis en étudiant différents algorithmes de gestion mémoire parallèles, afin d'en proposer des versions tirant avantage des spécificités de la bibliothèque des ATerms. Le but serait de fournir une bibliothèque concurrente, qui pourrait être instanciée une seule fois et partagée par les différents composants inclus dans un environnement, alors qu'actuellement elle est instanciée une fois par composant l'utilisant. Ceci résulterait en des échanges entre composants en temps constant, et en un gain mémoire important.

2.3.5 Développement et intégration de nouveaux concepts à Eiffel et SmartEiffel

Mots-clefs : langages à objets, compilation, types, analyse statique, optimisation, tuples, gestion mémoire, weak references, versatile references, SCOOP, concurrence.
Période : 2001-2004

J'ai continué mes travaux sur le langage Eiffel et le compilateur SmallEiffel (renommé SmartEiffel en septembre 2002), et que j'avais dû interrompre durant mon postdoctorat, en proposant diverses améliorations au niveau langage et en effectuant un important travail de développement et maintenance autour du compilateur.
A l'époque, SmartEiffel est l'un des logiciels libres produits à l'INRIA les plus diffusés au monde, et l'un des rares à avoir le label GNU. Il est également une plateforme de recherche et d'expérimentation unique, permettant d'implanter et de tester les concepts et techniques d'optimisation que nous développons.
J'ai ainsi participé à l'intégration au sein de SmartEiffel de concepts nouveaux pour le langage Eiffel, à savoir la notion de *tuple*, qui offre la possibilité de manipuler des N-uplets de valeurs typés pour la prise en compte plus aisée de fonctions à nombre variable d'arguments par exemple, ainsi que le concept d'agent, qui est comparable à une fermeture, *closure* dans un langage fonctionnel, avec un aspect d'évaluation partielle, et dont un des aspects utiles majeurs consiste à pouvoir augmenter l'expressivité des assertions présentes dans les contrats. Nous nous sommes particulièrement intéressés aux problèmes de typage en Eiffel posés par ces nouveaux concepts, notamment pour les agents pour lesquels nous avons conçu une solution que nous avons proposé à la communauté travaillant autour du langage [Rib+2003 ; Rib+2004]. Nous nous sommes également penchés sur la compilation efficace des tuples et agents et l'optimisation de leur exécution.
J'ai également participé aux travaux visant à intégrer SCOOP dans SmartEiffel. SCOOP (Simple Concurrent Object-Oriented Programming) est un concept permettant de faire très simplement de la programmation concurrente à haut niveau dans un langage à objets. Bien que partiellement défini depuis de nombreuses années, SCOOP n'avait encore jamais été mis en oeuvre dans aucun compilateur Eiffel. Notre implantation était donc la première au monde. A ce titre, ces travaux ont permis d'avoir un retour sur la définition de SCOOP, et ainsi de l'améliorer et la préciser. Là encore, nous nous sommes tout particulièrement penchés sur les problèmes d'implantation efficace des concepts SCOOP, notamment pour les gardes et la distribution de l'exécution.
J'ai aussi, en collaboration avec Paul Zimmermann, étudié et commencé à développer une bibliothèque en Eiffel pour la gestion de polynômes, qui avait pour but de servir de base à l'écriture d'outils de calculs à la MuPAD en Eiffel.
Notons aussi que j'ai aussi mis en place une infrastructure de développement autour de SmartEiffel, afin d'améliorer les pratiques de développement et de mieux monter en charge : serveur CVS, liste de diffusion mél pour l'équipe qui travaillait autour de SmartEiffel, SmartZilla un serveur de gestion de rapport de bogues

pour SmartEiffel basé sur BugZilla, permettant donc aux développeurs comme aux utilisateurs d'interagir, signaler des bogues et de suivre leur évolution.
En ce qui concerne les techniques générales d'optimisation des langages à objets, suite directe de mes travaux de thèse, les résultats obtenus ont été très satisfaisants. Les directions explorées pouvaient cependant encore être approfondies, notamment en améliorant la finesse des informations fournies par l'analyse globale et la prédiction de type implantées dans SmartEiffel.
De nouvelles directions de recherche apparaissaient également souhaitables, comme l'étude de méthodes hybrides combinant ma technique de liaison dynamique et d'autres plus classiques, selon des critères automatiquement déterminés (heuristiques et/ou statistiques). A plus long terme, l'analyse dynamique du code aurait dû également être abordée, notamment en complément de l'analyse statique. Ces recherches auraient pour but à la fois d'améliorer les contrôles de validité et l'optimisation du code généré, mais également de permettre une montée en charge, une utilisation à plus grande échelle et dans des contextes plus distribués, de ces techniques.
J'ai de plus mené des travaux visant à améliorer la gestion mémoire dans les langages à objets, tant en termes de performances que d'expressivité pour le développeur. Dans ce cadre, j'ai travaillé, en encadrant un D.E.A., sur la notion de références faibles (*weak references*) et son intégration dans les langages à objets. Une implantation dans SmartEiffel a été effectuée et est diffusée [MZC2003]. Ce travail s'est prolongé avec la conception de différentes sortes de références adaptables (*versatile references* [MZC2004]) typées, qui généralisent la notion de références faibles mais typées et donnent au développeur plusieurs niveaux permettant de contrôler plus finement la mémoire d'une application, tout en gardant la sécurité apportée par un ramasse-miettes.

2.3.6 Définition et implantation de nouveaux langages : FunTalk et SmallTalk2K

Mots-clefs : langages à objets, langages à prototypes, typage, réflexivité
Période : 2002

Suite à mes travaux sur Eiffel et ses extensions, j'ai activement participé à plusieurs travaux sur la définition de nouveaux langages.
Les premiers furent FunTalk/SmallTalk2K.
L'objectif était de définir un nouveau langage à objets inspiré du langage Eiffel pour ce qui concerne les aspects génie logiciel, et des langages à prototypes pour ce qui concerne les aspect formels. En outre, les aspects dynamiques du langage SmallTalk semblaient appropriés pour augmenter l'expressivité du langage. Ainsi, l'absence de typage statique confère une très grande expressivité au langage SmallTalk en permettant la création dynamique de classes et la définition réflexive du système. Il aurait été souhaitable de conserver une grande partie de cette expressivité dans le langage SmallTalk2K tout en conservant les acquis du langage Eiffel avec un système de typage sûr. Le langage SmallTalk2K était vu comme une couche à classes au dessus du langage bas niveau à prototypes FunTalk. Il était ainsi envisagé de rendre disponibles dans SmallTalk2K quelques primitives propres au langage FunTalk (par exemple la modification ou l'extension dynamique des méthodes dans un objet).
Un travail important restait cependant à faire pour la définition, puis à plus long terme, l'implantation de FunTalk et/ou SmallTalk2K.
Les études faites ont cependant servi de base aux travaux suivants sur LIsaac.

2.3.7 LIsaac : définition et implantation d'un langage à prototypes pour système d'exploitation à objets

Mots-clefs : langages à objets, langages à prototypes, système d'exploitation, interpréteur, compilateur
Période : 2002-2004

Isaac est le nom d'un système d'exploitation à objets qui a été conçu dans l'équipe Design par B. Sonntag. Le travail autour d'Isaac auquel j'ai participé a eu pour pour but de "mettre de l'objet dans l'OS", c'est-à-dire d'intégrer *réellement* les concepts objets au sein des systèmes d'exploitation. L'objectif était ici d'avoir des notions de haut niveau (objets, encapsulation, partage, droits, héritage, etc.) intégrées et utilisées à très bas niveau (système d'exploitation, juste au-dessus du matériel) avec le même confort que dans un langage à objets [SCZ2002].
Un prototype de ce système a été réalisé sur plateforme PC Intel x86, ainsi que sur d'autres architectures de type PDA (Personal Digital Assistant), comme les Palms et iPaqs. Afin d'offrir le plus rapidement possible un environnement complet et de nombreuses applications, des travaux visant à porter la `glibc` sur Isaac ont été également été menés.
Pour pouvoir implanter le système d'exploitation Isaac, un langage nommé LIsaac (Langage pour Isaac) a été créé, à la conception et l'implantation duquel j'ai participé (`https://lisaac.org/manual/manual.pdf`). Plus précisément, il s'agissait d'un langage à prototypes, capable d'être utilisé pour toutes sortes d'usages, avec des spécificités facilitant l'écriture de systèmes d'exploitation.
Les phases de conception et d'implantation de LIsaac ont été menées en parallèle, afin d'avoir très tôt un retour maximal sur les aspects pratiques, d'expressivité et d'utilisabilité des choix faits au niveau conceptuel, pour pouvoir les valider. Nous avons ainsi disposé d'un *interpréteur*, développé dans le cadre d'un D.E.A. que j'ai encadré, qui nous a permis une grande flexibilité et une grande réactivité par rapport aux évolutions du langage LIsaac. Un *compilateur* LIsaac, qui avait pour but de fournir un code le plus optimisé possible, afin d'atteindre une performance maximale, a également été développé par B. Sonntag et J. Boutet que j'ai en partie supervisés.

2.3.8 Instrumentation dynamique de programmes Java

Mots-clefs : Java, analyse programmes, instrumentation, trace, profil, runtime
Période : 2008-2012

J'ai dirigé le doctorat de Pierre CASERTA, dont le premier axe portait sur l'analyse de programmes Java, via instrumentation et profilage à l'exécution.

Contrairement à de nombreux autres travaux approchants, notre approche avait l'originalité d'instrumenter le Java bytecode, au moment de son chargement en mémoire par le class loader. En effet, instrumenter le bytecode permet de fonctionner même quand on n'a pas accès au code source, ce qui est assez souvent le cas en Java, les "exécutables" étant distribués sans le source, sous forme de fichier .jar par exemple. Travailler au niveau du bytecode loader, à la volée, implique certes un coût à l'exécution, mais a le gros avantage de permettre de "voir" aisément toutes les bibliothèques qui seront appelées par un programme, y compris les bibliothèque système. Notre approche avait aussi l'originalité relative d'analyser l'exécution des programmes non seulement au niveau de la méthode mais bien au niveau du bloc de base, pour recueillir des données d'exécutions avec un maximum de précision, comme par exemple les différents types d'instances sur les sites d'appels. Nous avons ainsi introduit une nouvelle technique et un nouvel outil permettant de recueillir des informations encore non proposées par d'autres analyseurs.

Nous avons implanté notre technique d'instrumentation et profilage dans un prototype, JBinsTrace (Java Bytecode Instrumenter and Tracer), appelée ensuite VITRAIL Tracer. Les expérimentations ont montré que cette technique était suffisamment performante pour permettre une utilisation (bien que ralentie) de programmes graphiques interactifs, tout en fournissant une trace assez détaillée, au niveau des blocs de base [CZ2011a ; CZ2012].

Pierre CASERTA a obtenu son doctorat [Cas2012] le 07/12/2012.

2.3.9 Analyse et optimisation du processus de build des variants logiciels ⇒

Mots-clefs : variants logiciels, configurations, Linux, construction, *build*
Période : 2021 ⇒

La construction de logiciels (processus de *build*) est une activité essentielle qui implique la compilation, le test et le déploiement de systèmes logiciels, tout en maintenant une attention constante à l'assurance qualité. La configurabilité des logiciels ne cessant d'augmenter, la nécessité de *builder* différentes configurations devient de plus en plus pressante. Cependant, cette tâche reste à la fois coûteuse et difficile à exécuter efficacement. Il est d'usage de *builder* une application logicielle pour un ensemble spécifique de configurations via un processus de *builds* indépendants, qui recompilent tout à partir de zéro, appelés "clean builds".

En revanche, si le concept de *build* incrémental, lui, a été largement exploré dans le contexte de *l'évolution progressive* des logiciels et des mises à jour mineures du code source (ie. les "commit" des développeurs), cette approche n'a cependant pas été beaucoup étudiée dans le domaine des *différentes configurations* d'un logiciel.
Dans le cadre de l'encadrement de la thèse de George Aaron RANDRIANAINA, auquel je participe avec Mathieu ACHER et Djamel Eddine KHELLADI, nous avons étudié l'idée que l'utilisation de techniques de construction incrémentale (*incremental build*) peut effectivement réduire les coûts associés à l'exploration de l'espace de configuration des systèmes logiciels.

Les développeurs de logiciels ont une compréhension profonde de l'importance du *build* des logiciels, qui est une phase cruciale mais exigeante du cycle de vie global du développement des logiciels. Cela est particulièrement vrai lorsqu'il s'agit de construire des grands systèmes ou des systèmes très configurables, car la gamme très étendue d'options de configuration entraîne une prolifération de versions qui doivent être construites et évaluées. Linux est justement un exemple de système très grand et fortement configurable, doté d'un large éventail d'options pouvant être combinées.
Dans [Ran+2022] nous avons détaillé la mise en œuvre de la construction incrémentale dans deux scénarios d'application pratique, et avons fourni une évaluation initiale sur deux études de cas spécifiques, à savoir x264 et le noyau Linux. Dans le contexte de x264, il a été observé que la construction des différentes configurations peut être effectuée progressivement dans un certain ordre, ce qui permet de réduire la durée totale du processus de *build*. Nous avons ainsi fourni une preuve de concept de la possibilité d'exercer un contrôle sur le processus de génération de configurations, l'utilisation d'éléments communs permettant de réduire le temps de *build* de 66 % par rapport à l'utilisation exclusive de *clean builds*.
Cependant, pour la construction incrémentale de configurations du noyau Linux, il n'a pas été possible de trouver d'ordre idéal en raison de la disparité importante entre les configurations aléatoires, qui sont "perdues" au milieu d'un gigantesque espace de configurations possibles (15000 options de configurations). Nos preuves empiriques remettent donc en question l'efficacité de la compilation incrémentale pour les processus post-commit actuellement utilisés pour le code du système d'exploitation Linux.

Ces résultats montraient les avantages de l'utilisation d'une approche de construction incrémentale, bien que principalement dans le contexte de systèmes logiciels configurables à petite échelle, contrairement au système d'exploitation Linux. Nous avons donc poursuivi nos travaux pour analyser le *build* de Linux, notamment.
Dans [Ran+2023] nous présentons PyroBuildS, une nouvelle méthodologie que nous avons conçue visant à explorer efficacement l'espace de configuration étendu de Linux par le biais de constructions incrémentales. Comme les feux d'artifices, d'où son nom, PyroBuildS part d'une collection de configurations de départ (les "fusées") et génère ensuite des configurations mutantes (les "étincelles") qui sont générées à partir de chacune des configurations de base susmentionnées.
Cette approche facilite l'exploration de l'espace de configuration, grâce à une construction incrémentale efficace des mutants, tout en maintenant un niveau appréciable de diversité. Notre étude démontre que la stratégie PyroBuildS exploite efficacement les capacités de mise en cache de Make, ce qui se traduit par des réductions substantielles du temps de construction pouvant aller jusqu'à 85 %. Cette constatation est étayée par notre analyse de 2520 builds. De plus, nous avons observé une gamme variée d'options, représentant 33% du total des options, et avons appliqué avec succès notre approche à 15 sous-systèmes de Linux sur 17.
PyroBuildS pourrait ainsi être utilisé par les contributeurs individuels et par les services d'intégration continue pour améliorer efficacement le nombre de configurations *buildées* ou pour minimiser les dépenses associées à la construction de configurations multiples.

Ces travaux se poursuivent actuellement intensément. La thèse de George Aaron RANDRIANAINA est en cours, soutenance prévue vers 10/2024.

2.4 Optimisation basse consommation énergétique

Mots-clefs : énergie, basse consommation, optimisation
Période : 2004-2016

En lien avec le domaine des systèmes embarqués, je me suis intéressé à la basse consommation énergétique. En effet, un nombre important (et en forte croissance) de systèmes embarqués ne sont pas reliés à une alimentation électrique continue et doivent être autonomes. La minimisation de la consommation électrique est donc capitale pour de tels systèmes. A l'autre extrémité du spectre, les très gros systèmes (super-calculateurs, clusters...), eux aussi sont concernés par la basse consommation, notamment du fait des problèmes d'échauffement (qui en diminuent la fiabilité) et de dimensionnement et coût des alimentations électriques et des systèmes de refroidissement.

Ces problèmes de consommation énergétiques peuvent être abordés selon deux points de vue, puissance de crête (instantanée) ou énergie (puissance par temps) et traités à deux niveaux complémentaires, le niveau matériel (électronique) et le niveau logiciel. Dans ce domaine, ma recherche cible tout d'abord et principalement le côté logiciel [Zen2006], avec des travaux portant sur l'ordonnancement (section 2.4.1), le placement en mémoire (section 2.4.2), les plateformes de support expérimental (section 2.4.3), et les (protocoles de) réseaux de capteurs sans fil (section 2.4.4).

2.4.1 Ordonnancement basse énergie

Mots-clefs : énergie, basse consommation, optimisation, ordonnancement
Période : 2004-2012

J'ai ainsi commencé par aborder l'ordonnancement basse énergie dans un contexte multiprocesseur, en collaboration avec N. Navet de l'INRIA et J. Goossens de l'Université Libre de Bruxelles). Pour ces travaux, j'ai participé activement à la réalisation d'un système de simulation et d'ordonnancement multi-processeurs (supervision de M. Grenier). Ce logiciel graphique a été utilisé pour les recherches du projet TRIO en ordonnancement.
Les résultats de nos recherches dans ce domaine ont été un nouvel algorithme d'ordonnancement multiprocesseur basse énergie [NGZ2005].

J'ai repris et continué ces travaux avec Liliana Cucu-Grosjean et Cristian Maxim, avec qui j'ai proposé une méthode de réduction de l'énergie via la réduction des préemptions dans l'ordonnancement [MCZ2011].

2.4.2 Placement mémoire basse énergie

Mots-clefs : énergie, basse consommation, optimisation, mémoire, SPM, scratch-pad, algorithmes génétiques
Période : 2006-2012

J'ai également attaqué le problème de la basse consommation en me basant sur mes compétences en analyse et optimisation de programmes et en gestion mémoire, en travaillant sur des gestions mémoire prenant en compte l'énergie (*energy-aware memory placement and management*).
En effet, en ce qui concerne la consommation globale du système, la ressource mémoire peut avoir un impact considérable. La mémoire est souvent hiérarchisée et parallélisée pour fournir une bande passante adaptée aux traitements du processeur. Il me semble donc nécessaire de se focaliser d'une part sur la spécialisation (paramétrage) du système mémoire par rapport à un ensemble de tâches à exécuter et d'autre part sur l'optimisation de son utilisation pour améliorer la consommation d'énergie et si possible les temps d'accès.

Les optimisations classiques par placement des données dans la hiérarchie qui visent à réduire les temps d'accès conduisent en général à réduire l'énergie par accès mémoire. En effet, le coût en énergie et en temps d'un accès mémoire est d'autant plus grand que cet accès porte sur un niveau de la hiérarchie éloigné du processeur. Par ailleurs, la contribution de l'énergie statique, directement liée à la taille de la mémoire embarquée, devient de plus en plus importante. Pour la réduire, une piste consiste à paralléliser la mémoire en bancs, en essayant de maximiser le nombre de bancs mis au repos. Cette gestion remet en cause certaines stratégies classiques d'optimisation.

Il m'est donc apparu nécessaire d'étudier et de développer des techniques de placement de données tenant compte des caractéristiques et contraintes des tâches, de la structure de la mémoire et des possibilités de mise au repos de bancs mémoire. Ceci a par exemple pour but de concentrer les allocations nouvelles dans les bancs mémoires déjà actifs car partiellement utilisés, afin de faciliter la libération des autres bancs, qui pourront ainsi être temporairement désactivés. Le placement de certaines données (par exemple en fonction des priorités des tâches) dans des parties de la hiérarchie mémoire moins gourmandes en énergie (car moins rapides) offre également un fort potentiel pour réduire la consommation en énergie.
Ces travaux se sont appuyés sur SPECO (*Software Platform for the Evaluation of Compilation Optimizations*), un prototype de plateforme dont j'ai géré le développement en 2006 et qui avait pour but de faciliter les validations expérimentales en faisant du benchmarking automatisé, notamment au niveau énergétique.

Ils ont également donné également lieu en 2007 à une collaboration avec l'IRIT et le LIP6 au sein du projet ANR MORE (*Multicriteria Optimizations for Real-time Embedded systems*, cf. section 2.4.3), où j'étais responsable du critère de consommation énergétique lié au placement des données.
Ceci a constitué le coeur des travaux de Maha IDRISSI AOUAD, que j'ai encadrée en master puis en doctorat (co-direction). Nous avons étudié les mémoire scratch-pad (SPM) [IZ2007 ; IZ2008], qui présentent l'avantage de permettre de controller logiciellement le placement des données, contrairement au cache.
L'optimisation globale de fonctions multimodales étant également un problème délicat à résoudre du fait de la grande quantité d'optima locaux de ces fonctions, nous avons proposé et étudié différents nouveaux algorithmes hybrides et distribués afin de résoudre ces problèmes. Dans le cas de l'optimisation globale de fonctions multimodales, nos algorithmes hybrides convergent plus souvent vers la solution optimale globale. Nous avons également proposé des versions distribuées et coopératives de ces nouveaux algorithmes hybrides, qui sont plus rapides que leurs versions séquentielles respectives. Nous avons ainsi appliqué ces algorithmes pour en faire des heuristiques d'allocation optimale des données basées sur les algorithmes génétiques, sur la recherche tabou, et sur une hybridation des deux. Nos résultats expérimentaux montrent de bonnes performances, avec une réduction de la consommation d'énergie en mémoire de 76% à 98% sur nos programmes de tests [Idr+2010c ; Idr+2010a ; ISZ2010a ; Idr+2010b ; ISZ2010c ; ISZ2010b ; Idr+2010b].

Maha IDRISSI AOUAD a obtenu son doctorat [Idr2011] le 04/07/2011.

2.4.3 Basse énergie : collaborations et plateformes

Mots-clefs : basse consommation, plateforme, Open-PEOPLE, modélisation, simulation
Période : 2007-2013

Dans le cadre de ces recherches sur la basse consommation énergétique, j'ai participé à plusieurs groupes de chercheurs : AS Num106 Conception Faible Consommation du CNRS, axes Consommation et énergie et Logiciel Embarqués du GDR SoC/Sip (System On Chip - System In Package), Groupe de Travail StrQds (Systèmes Temps Réel et Qualité de Service) du GDR ARP (Architecture, Réseaux et systèmes, Parallélisme) et GDR ISIS (Information, Signal, Images et ViSion). En tant que membre du réseau HiPEAC (European Network of Excellence on High-Performance Embedded Architecture and Compilation), j'ai créé également et animé un cluster de chercheurs sur "Architecture-aware compiler solutions for energy issues in embedded systems."
Ces collaborations, ainsi que la charge considérables de développement de plateforme pour les expérimentations pour la basse consommation m'ont convaincu de la nécessité de mutualiser au maximum les développements et plateformes.

C'est donc dans ce contexte que j'ai dirigé des travaux, en tant que partenaire, sur le projet ANR MORE et sa plateforme expérimentale [Cas+2010 ; PZ2010] (qui ont également fourni le contexte des travaux liés à la thèse de Maha IDRISSI AOUAD, comme expliquée en section 2.4.2).
J'ai collaboré aussi avec des chercheurs plus spécialisés en électronique (au LESTER à Lorient) pour caractériser et modéliser les effets de l'architecture matérielle sur les optimisations énergétiques, notamment en mémoire. Cette caractérisation doit se faire au moins en partie grâce au développement de modèles, de simulateurs et d'outils de visualisation de ces artefacts architecturaux. Cette collaboration nous a amené à monter, en l'élargissant à sept autres partenaires académiques et industriels, une importante proposition de projet ANR de plateforme Open-PEOPLE qui visait à explicitement réaliser une plateforme fédératrice, matérielle et logicielle, de modélisation, d'expérimentation et de simulation pour l'architecture et la basse consommation [Chi+2010 ; Sen+2011b ; Sen+2012a ; Sen+2012b ; SBZ2012]. Dans Open-PEOPLE j'étais responsable de la partie logicielle de la plateforme.

2.4.4 Basse énergie dans les réseaux de capteurs sans fil

Mots-clefs : WSN, réseaux de capteurs sans fil, basse consommation, protocoles, RIOT OS, Contiki, Cooja.
Période : 2013-2016

J'ai co-dirigé dans le cadre du doctorat de Kévin ROUSSEL des travaux sur les réseaux de capteurs sans fil (WSN - Wireless Sensors Networks), visant à améliorer leurs performances temporelles et énergétiques.

Dans le domaine des réseaux de capteurs sans-fil, les piles réseau spécialisées avaient été l'objet de recherches très actives depuis de nombreuses années, mais très peu de ces études, notamment concernant les couches basses de ces piles réseau, avaient dépassé le stade de la théorie. Leurs implantations n'avaient généralement pas fait l'objet d'efforts poussés ou systématiques, surtout dans le cadre des systèmes d'exploitation spécialisés. Nous avons donc notamment travaillé à l'étude des protocoles de communication dans ces réseaux, et à l'analyse des interactions entre les protocoles des couches basses et les plates-formes logicielles dédiées, ie. les systèmes d'exploitation embarqués sur les noeuds.

Afin d'améliorer et d'optimiser ces couches basses des piles spécialisées, nous avons proposé S-CoSense un nouveau protocole MAC très basse consommation pour les WSN, amélioration de CoSense. Il permet d'obtenir de bonnes performance énergétiques, tout en conservant un débit approprié et évitant les phénomènes de congestion ou les pertes de paquets [RSZ2015b].

Avec l'implantation prototypale que nous avons faite de notre protocole S-CoSense, nous avons montré que nous pouvions amener des progrès notables dans la qualité de service des WSN, notamment avec un trafic réseau intense. Pour cette implantation, nous avons essayé le système d'exploitation Contiki OS, très connu de la communauté des OS embarqués, mais l'avons trouvé trop bogué. Nous nous sommes alors rabattus sur RIOT OS, développé à l'INRIA. Un peu plus récent, RIOT était moins complet, moins connu, mais plus solide que Contiki. Cependant, nous avons rencontré certains problèmes sur les noeuds du réseau qui nous ont amené à proposer des améliorations dans le système d'exploitation RIOT [RSZ2015b].

Nous avons également trouvé des problèmes de précision importants et inattendus dans les simulations / émulations effectuées par le framework Cooja/MPsim, couramment utilisé par la communauté pour simuler les WSN. Après avoir analysé les problèmes de fiabilité posés par l'utilisation de cet outil pour effectuer des évaluations de performances dans les WSN, nous avons proposé des solutions [RSZ2015a ; RSZ2016].

Kévin ROUSSEL a obtenu son doctorat [Rou2016] le 03/06/2016.

2.5 Visualisation avancée de programmes à objets

Mots-clefs : Java, trace, profil, runtime, visualisation, métaphore, 3D Hierarchical Edge Bundle, VITRAIL
Période : 2009-2012

Dans le doctorat de Pierre CASERTA, que j'ai dirigé, le second axe de recherche portait sur la visualisation avancée de la structure et du comportement de programmes Java, Ce second axe s'appuyait bien entendu sur les informations obtenues via le premier axe, celui d'instrumentation et profilage du bytecode Java décrit en section 2.3.8. dans le cadre de sa thèse et du projet que j'avais appelé VITRAIL (VIsualisation Temps Réel Avancée et Immersive de Logiciels). En effet, ces informations offrent des détails intéressants sur le fonctionnement des programmes et aident à expliquer le comportement des logiciels, aussi bien pour déceler les problèmes de performance que les problèmes de codage.

Nous avons basé la visualisation avancée que nous proposons [CZ2011b] sur la métaphore graphique de la ville, où la structure statique du logiciel est représentée par une ville, les bâtiments représentant les classes et les quartiers et rues représentant les packages. Nous avons dans notre prototype implanté deux types d'organisations spatiales (*layout*), à savoir les paquets imbriqués (*nested layout*) et les paquets distribués (*street layout*).
En nous basant sur les traces d'exécution fournies par JBinsTrace / VITRAIL Tracer (cf. section 2.3.8), nous avons proposé une nouvelle visualisation des appels de fonction dynamiques, superposée à la métaphore de la ville. Ceci permettait ainsi de visualiser aussi bien des éléments *statiques* que des éléments *dynamiques* du logiciel. Nous avons conçu et implanté un nouvel algorithme de groupement des liens proches, évitant un trop grand encombrement visuel, le 3D-Hierachical Edge Bundle. Celui-ci représente une généralisation du 2D-Hierarchical Bundle, et est utilisable pour toutes sortes de groupement de relations en 3D (non spécifiques au code, donc).
Dans le cadre de la visualisation de programmes, ceci nous a permis de proposer une nouvelle technique de visualisation, prototypée dans un outil, VITRAIL Visualizer, offrant via les 3D-Hierarchical Bundles une visualisation assez claire de graphes d'appels comportant des millions de relations, évitant un surencombrement visuel [CZB2011b ; CZB2011a].

Pierre CASERTA a obtenu son doctorat [Cas2012] le 07/12/2012.

2.6 Cybersécurité ⇒

Mots-clefs : cybersécurité
Période : 2016 ⇒

Depuis 2016, la cybersécurité est pour moi un domaine majeur de mes travaux de direction de la recherche. J'ai pour le moment abordé 4 axes complémentaires, portant sur l'analyse du protocole de sécurité IKEv2 (section 2.6.1), sur l'intégration explicite des propriétés de sécurité dans les logiciels (section 2.6.2), sur l'analyse de binaires détecter des maliciels (section 2.6.3), pour détecter et classifier des packers de maliciels (section 2.6.4), ou pour classifier des maliciels (section 2.6.5), et enfin sur la défense contre des attaques par canaux auxiliaires, aussi bien sur le plan pratique que plus théorique (cf. 2.6.6).

2.6.1 Analyse du protocole de sécurité IKEv2

Mots-clefs : IKEv2, protocole, DoS, deviation attack
Période : 2017-2020

J'ai co-dirigé dans le cadre de la thèse CIFRE de Tristan NINET, des travaux sur l'analyse du protocole d'authentification IKEv2, qui est notamment très utilisé dans les VPN.
Nos travaux ont consisté notamment à attaquer/vérifier le protocole IKEv2 et sa spécification à l'aide d'outil comme les outils d'exécution symbolique, de fuzzying, de model checking et de vérification de protocoles (Angr, SPIN, Proverif, Tamarin, etc.).

Dans des analyses précédentes, il avait été montré qu'IKEv2 possédait deux vulnérabilités d'authentification qui étaient considérées comme non exploitables. Dans nos nos travaux [Nin+2019a ; Nin+2019b] nous avons

prouvé qu'en fait, la première vulnérabilité n'existe pas. De plus, nous montrons que la deuxième vulnérabilité est exploitable, dans certaines conditions, pour mettre en œuvre une nouvelle attaque lente, furtive, par déni de service, que nous appelons l'attaque par déviation (*deviation attack*). Nous avons identifié les pré-requis de l'attaque, et avons proposé des contre-mesures possibles. Nous avons notamment proposé une modification simple du protocole IKEv2 qui élimine la vulnérabilité à cette attaque, et plus largement à celles du même type.

Tristan NINET a obtenu son doctorat [Nin2020] le 09/03/2020.

2.6.2 Intégration explicite de propriétés de sécurité dans les logiciels

Mots-clefs : propriété de sécurité, propriété non-fonctionnelle, expressivité, side channel, crypto
Période : 2017 ⇒

Je me suis intéressé depuis 2017 à l'intégration explicite des propriétés de sécurité dans les logiciels.
Le projet EU H2020 TeamPlay (Time, Energy and security Analysis for Multi/Many-core heterogenous PLAtforms) a notamment fourni un cadre et un support financier du 01/01/2018 au 30/06/2021 pour ces travaux. TeamPlay visait a intégrer dans les programmes des propriétés non fonctionnelles comme le temps, l'énergie et "la" sécurité comme « citoyens à part entière » (`https://www.teamplay-h2020.eu`).
Le but était de permettre aux programmes de faire de l'introspection sur leur propre temps, leur consommation d'énergie, leur sécurité, etc., et de permettre aux développeurs de raisonner et manipuler ces propriétés au niveau du code source, aussi aisément et explicitement que les propriétés fonctionnelles classiques.

En effet, les développeurs accordent de plus en plus d'importance aux propriétés des programmes en matière d'énergie, de temps et de sécurité (ETS), en particulier lorsque les applications s'exécutent sur des systèmes sensibles à l'ETS, tels que les dispositifs embarqués ou l'internet des objets. De plus, les développeurs manquent d'outils et de propriétés de langage leur permettant de raisonner sur énergie, temps et sécurité.

Dans le cadre des travaux de recherche que j'ai dirigé en tant que partenaire de TeamPlay, je me suis focalisé plus particulièrement sur les propriétés liées à la sécurité, au niveau de leur expressivité (comment les exprimer dans le source, avec quelle syntaxe pour le développeur et les outils), leur modélisation (sémantique), leur quantification théorique ou leur mesure pratique, et leur implantation effective (prise en compte au niveau du compilateur et/ou du runtime) .
En collaboration avec l'Université de Saint Andrews, nous avons proposé [Bro+2019] un nouveau framework, appelé Drive, qui permet à un développeur de raisonner sur l'énergie, le temps, ou la sécurité (ou sur d'autres propriétés non fonctionnelles) de leurs programmes en tant que propriétés explicites ("de première classe") du langage. Pour ce faire, nous avons conçu un langage de spécification de contrats (CSL - Contract Specification Language) permettant aux développeurs de raisonner sur ces propriétés explicites d'énergie, de temps et de sécurité en exprimant des contrats, qui sont ensuite prouvés corrects par un système de type formels sous-jacent. Nous avons expérimenté notre framework sur des exemples représentables, démontrant des propriétés de pire cas en ETS prouvables. Ces travaux prototypaux ont été intégrés dans la plateforme TeamPlay [Rou+2023].

Ces travaux ont été eux aussi perturbés par les difficultés, chronophages et énergivores, que j'explique en sections 2.7 et 2.8.

2.6.3 Détection de maliciels par analyse syntaxique de binaires

Mots-clefs : analyse syntaxique, maliciel, Yara, machine learning, apprentissage automatique
Période : 2017-2021

Je me suis intéressé à partir de 2017 à l'analyse de binaires pour la cybersécurité.

Un premier axe consistait à tenter de détecter des maliciels.

Dans ces travaux s'intégraient la thèse d'un doctorant, que j'appellerai "X" ici, démarrée en 2017, et le projet DGA RAPID MASSE (Modular Automated Syntactic Signature Extraction), démarré en 2018, en collaboration avec l'entreprise Teclib.
Les travaux du doctorant devaient porter sur l'analyse de programmes binaires afin d'extraire des signatures syntaxiques de maliciel. L'approche était d'utiliser des techniques d'apprentissage automatique pour gérer les règles Yara permettant de détecter les maliciels via leurs signatures.
En termes fonctionnels, le projet MASSE avait comme objectif de développer une architecture cloud de protection d'entreprises et d'organisations sensibles. Dans cette architecture, les machines ("clients") protégées intègrent un logiciel léger de type antivirus qui s'appuie sur des règles de détection syntaxique au format YARA pour scanner les binaires suspects. Lorsque des analyses plus poussées et requérant beaucoup de ressources sont nécessaires, elles s'exécutent sur un ou des serveurs dans un cloud. L'approche modulaire du cloud permet l'intégration de nombreux modules d'analyse et de génération de règles, à même de travailler de concert pour établir si un fichier est malicieux ou non, et pour générer, le cas échéant, les règles YARA qui vont permettre la détection du maliciel sur les machines clientes protégées par la solution MASSE.
Les travaux du doctorant "X" étaient censés fournir des modules intégrables à cette plateforme MASSE. Malheureusement, suite à sa psychologie particulière, "X" a été renvoyé par l'Université de Rennes 1 (UR1) en 2019 pour avoir proféré des menaces de mort en cours de sport de l'UR1. Ces travaux ont donc été quelque peu stoppés.

Les travaux sur MASSE, avec une forte composante d'ingénierie, ont cependant bien progressé de leur côté. Une plateforme fonctionnelle et fiable a été livrée à la fin du projet en 2020 [DLZ2019], sur laquelle viennent s'agréger des modules d'analyse et de génération de règles. Cette plateforme est le socle à partir duquel pourra évoluer MASSE dans le futur en intégrant de nouveaux modules qui pourront renforcer ses capacités de détection et de protection contre les maliciels.
Côté client un agent Linux et un agent Windows ont été développés, qui se chargent d'intercepter et scanner les interactions du système d'exploitation avec le système de fichiers. Les choix architecturaux et technologiques faits durant la phase d'analyse se sont révélés efficaces. Le découpage en couches selon le modèle de la Clean Architecture a facilité les tests et a produit un dispositif pérenne qu'il est aisé de modifier. L'API de communication avec les modules est simple et répond aux besoins identifiés initialement. De ce fait l'intégration de nouveaux modules ne requiert pas de gros efforts. Le protocole choisi pour le dialogue entre les modules et la plateforme, JSON-RPC, remplit le rôle qui lui est dévolu, à savoir une communication simple, agnostique et sécurisée en facilitant l'isolement du module du reste du serveur. Le choix de Nginx en tant que proxy inverse, load balancer et serveurs de fichiers, s'est avéré convaincant par sa rapidité, sa bonne résistance à la montée en charge et sa fiabilité. Le langage Go fût aussi un bon choix : langage (plus) sécurisé, code produit efficace, facilité du développement multi-plateformes.

Des axes d'amélioration et de poursuite de ces travaux ont été clairement identifiés.
Un premier axe serait la conception de modules supplémentaires, en particulier au niveau des modules de classification dont les exemplaires actuellement disponibles pêchent par l'absence de modules basés sur l'analyse du comportement dynamique des maliciels (exécution en sandbox, exécution symbolique) qui se montreraient utiles quand les analyses actuelles échouent.
Un second axe concerne les stratégies de prise de décision qui tranchent parmi les résultats de la classification menée par les différents modules : une première politique de prise de décision existe déjà, mais est trop simpliste. De nouvelles politiques de décision doivent être implantées, visant à atteindre un consensus parmi les voix potentiellement dissonantes des différents modules de classification. Le taux de faux positifs est actuellement trop haut. Deux vecteurs d'amélioration sont possibles. Le premier revient à améliorer le module NGrams de détection syntaxique actuel, par exemple en utilisant un algorithme génétique pour amender la qualité des règles qui en découlent. Le second est de profiter de l'approche modulaire de MASSE pour ajouter de nouveaux modules de génération de règles.
Un troisième axe concerne la qualité des règles, qui est améliorable, ce qui aurait un impact positif sur le taux de faux positifs. Il faudrait pour cela orienter la production de règles vers la qualité et diminuer la quantité de règles produites, en ayant par exemple un système qui à chaque nouvelle génération factorise le lot de nouvelles règles et extrait un facteur commun parmi ces nouveaux candidats. Implémenter une gestion

de la base de règles, visant à éviter la prolifération des règles, ce qui implique une sélection basée sur des tests avant publication des règles, est également une piste.
Une proposition de projet MASSE-2 faisant suite à MASSE a été élaborée avec Teclib, en prenant en compte ces axes de progrès, mais elle a été mise en attente suite à un changement des priorités de l'entreprise.

En plus du renvoi du doctorant "X", ces travaux ont été eux aussi perturbés par les difficultés, chronophages et énergivores, que j'explique en sections 2.7 et 2.8.

2.6.4 Détection et classification de packers de maliciels

Mots-clefs : maliciel, packing, classification, machine learning, packer, obfuscation, classification, clustering
Période : 2017-2021

Un second axe de mes travaux sur l'analyse de binaires pour la cybersécurité consistait à détecter et classifier des packers (empaqueteurs, compresseurs) de maliciels, afin de mieux pouvoir les traiter.

Les packers sont en effet des outils très répandus utilisés par les auteurs de maliciels pour entraver la détection et l'analyse statiques des maliciels. Il est essentiel d'identifier le packer utilisé pour emballer un maliciel afin de le décompacter et de l'analyser correctement, que ce soit manuellement ou automatiquement. Bien que de nombreux packers bien connus soient utilisés, on observe une tendance croissante à l'utilisation de nouveaux packers spécialisés qui compliquent l'analyse et la détection des maliciels.
Les travaux de recherche précédents ont été très efficaces dans l'identification des packers connus ou de leurs variantes, avec des techniques basées sur les signatures, l'apprentissage automatique supervisé ou la similarité. Toutefois, l'identification de nouvelles classes de packers reste un problème ouvert.

Dans les travaux de recherche que j'ai co-dirigés sur ce sujet, centrés notamment autour de la thèse de Lamine NOUREDDINE [Nou2021], nous avons proposé un classificateur de packers auto-évolutif qui fournit une solution efficace, incrémentale et robuste pour faire face à l'évolution rapide des packers [Nou+2021]. Nous avons conçu une mesure composite de la distance par paire combinant différents types de caractéristiques du packer. Nous avons dérivé une approche incrémentale de clustering capable d'identifier à la fois des (variantes de) classes de packers connues et de nouvelles classes, ainsi que de mettre à jour les clusters de manière automatique et efficace. Notre système est ainsi capable d'améliorer, intégrer, adapter et faire évoluer en permanence la connaissance du packer. De plus, afin d'optimiser les coûts de post-traitement des packers après clustering, nous avons introduit une nouvelle stratégie de sélection de petits sous-ensembles d'échantillons pertinents à partir des clusters.

Ces travaux ont été implantés dans le logiciel SE-PAC (Self-Evolving PAcker Classifier), logiciel de classification de packers auto-évolutif qui identifie la classe du packer utilisé pour packer un fichier exécutable Win32 (malveillant). Il est capable d'identifier des classes de packers connues et inconnues, et de mettre à jour ses connaissances sur les packers en fonction des classes de packers prédites. SE-PAC s'appuie sur un clustering incrémental semi-supervisé. Il se compose d'une phase hors ligne et d'une phase en ligne. Dans la phase hors ligne, un ensemble de caractéristiques (*features*) est extrait d'une collection de binaires *packés* fournis avec leurs étiquettes, puis un algorithme de clustering basé sur la densité (DBSCAN) est utilisé pour regrouper les packers similaires dans des clusters en fonction d'une mesure de distance. Au cours de cette étape, le seuil de similarité est ajusté afin de former les clusters qui correspondent le mieux à l'ensemble des étiquettes fournies. Dans la phase en ligne, SE-PAC reproduit les mêmes opérations d'extraction des caractéristiques et de calcul de la distance, puis utilise une version personnalisée de l'algorithme de clustering incrémental DBSCAN afin de classer les échantillons *packés* entrants, soit dans des classes de packers connues en étendant les clusters existants, soit dans de nouvelles classes de packers en formant de nouveaux clusters, soit en les laissant temporairement non classés (voir la notion de bruit de DBSCAN). Les clusters formés après chaque mise à jour servent de base à l'auto-évolution de SE-PAC. Enfin, SE-PAC comprend une stratégie de sélection post-clustering qui extrait un ensemble d'échantillons pertinents des clusters trouvés afin de réduire le coût du traitement du packer après clustering.

L'efficacité de notre approche et sa résistance au temps ont été évaluées à l'aide de : 1) un flux de maliciels issus du monde réel, composé de 16 000 binaires *packés*, comprenant 29 packers uniques, et 2) un ensemble de données synthétiques composé de 19 000 binaires *packés* élaborés manuellement, comprenant 31 packers uniques (y compris des packers personnalisés). Nos expériences montrent des résultats prometteurs en termes d'effectivité, d'efficacité et de robustesse pour la détection et la classification des packers.

Lamine NOUREDDINE a obtenu son doctorat [Nou2021] le 21/12/2021. Ces travaux ont été eux aussi perturbés par les difficultés, chronophages et énergivores, que j'explique en sections 2.7 et 2.8.

2.6.5 Classification de maliciels

Mots-clefs : maliciel, classification, machine learning
Période : 2019-2022

Un troisième axe de mes travaux sur l'analyse de binaires pour la cybersécurité consistait à classifier des maliciels, afin d'améliorer la faisabilité et la montée en charge de l'analyse de nouveaux maliciels.

Historiquement, l'analyse des maliciels a en effet largement fait appel au savoir-faire humain pour la création manuelle de signatures afin de détecter et de classer les maliciels. Or cette procédure est très coûteuse et prend beaucoup de temps, ce qui ne permet pas de faire face à l'explosion actuelle des cybermenaces. La solution consiste à automatiser largement l'analyse des maliciels. À cette fin, la classification des maliciels permet d'optimiser le traitement de vastes corpus de maliciels en identifiant les ressemblances entre des instances similaires. Par conséquent, la *classification des maliciels* est une activité clé pour l'analyse des maliciels, qui est primordiale dans le fonctionnement de la sécurité informatique dans son ensemble.

Dans les travaux de recherche que j'ai co-dirigé sur ce sujet, centrés notamment autour de la thèse de Cassius DE OLIVEIRA PUODZIUS [Puo2022], nous avons abordé le problème de la classification des maliciels en adoptant une approche dans laquelle l'intervention humaine est évitée autant que possible. Nous évitons ainsi la subjectivité inhérente à l'analyse humaine, en établissant la classification des maliciels uniquement à partir de la base de données directement extraites de l'analyse des maliciels. Avec cette approche centrée sur les données "brutes", notre objectif était d'améliorer l'automatisation de l'analyse des maliciels et de la combiner avec des méthodes d'apprentissage automatique capables de repérer et de révéler de manière autonome des points communs involontaires dans les données.

Ce travail s'est déroulé par étapes.
Dans un premier temps, nous nous sommes concentrés sur l'amélioration de l'analyse des maliciels et de son automatisation, en étudiant de nouvelles façons de tirer parti de l'exécution symbolique dans l'analyse de ces maliciels, avec une attention particulière portée à l'utilisation optimale des hyperparamètres et des tactiques des solveurs SMT. Nous avons également proposé des primitives de base et un nouveau paradigme d'évaluation pour les systèmes d'analyse des maliciels, afin de permettre la mise en place d'expériences plus fiables.
Nous avons en parallèle développé une plateforme distribuée pour augmenter notre puissance de calcul pour supporter ces travaux.
Nous nous sommes ensuite concentrés sur la représentation du comportement des maliciels, en accordant une attention particulière à sa précision et à sa robustesse. Nous avons ainsi proposé la formulation d'une représentation graphique compacte du comportement des programmes, les External Calls Dependency Graphs (ECDG) [Puo+2021], ainsi qu'une fonction correspondante pour le calcul de la similarité par paire, qui est précise et robuste. Notre expérimentation a montré en effet que cette fonction de similarité était très efficace, avec une accélération de x3,30 par rapport à radiff2 pour l'implémentation standard et x354,11 pour l'implémentation améliorée avec cache.
Enfin, nous nous sommes focalisés sur le clustering des maliciels, en concevant une nouvelle stratégie basée sur un clustering d'ensembles qui n'a aucune restriction en ce qui concerne la combinaison des caractéristiques syntaxiques et comportementales considérées, et qui reste extensible en pratique. Nos évaluations ont montré [Puo2022] que cette stratégie génère des clusters qui produisent des résultats presque infaillibles - score

d'homogénéité de 0,983 pour la phase de précision - et une perte d'information marginale pour un ensemble de données très pollué - score NMI de 0,974 entre les clusters initiaux et finaux de la phase de robustesse.
Dans l'ensemble, les ECDG et notre fonction de similarité permettent en pratique de créer des systèmes autonomes de recherche et de clustering de maliciels capables d'aider l'analyse humaine ou d'améliorer les modèles de classification pour l'analyse des maliciels.

Cassius DE OLIVEIRA PUODZIUS a obtenu son doctorat [Puo2022] le 19/12/2022. Ces travaux ont été eux aussi perturbés par les difficultés, chronophages et énergivores, que j'explique en sections 2.7 et 2.8.

2.6.6 Défense contre les attaques par canaux auxiliaires ⇒

Mots-clefs : canaux auxiliaires, fuite d'information, ladderisation , indiscernabilité,
Période : 2018 ⇒

J'ai, dans le cadre du projet TeamPlay (cf. 2.8), dirigé des travaux sur des méthodes et des techniques visant à améliorer la sécurité des logiciels, en supprimant certaines cyber-vulnérabilités présentes dans les codes sources, en particulier celles qui permettent des attaques par canaux auxiliaires (ou canaux cachés). Nous avons en effet abordé ce type spécifique de cyberattaques où les attaquants exploitent les fuites d'informations existantes lors de l'exécution physique d'un programme, par exemple les fuites liées au temps ou à l'énergie, pour découvrir des informations secrètes telles que des clés de chiffrement ou d'autres données sensibles.
La littérature existante présentait diverses tentatives pour résoudre le problème de la prévention des attaques par canaux auxiliaires, tentatives qui souvent s'appuient sur des mesures visant à réduire la discernabilité, c'est à dire les différences observables entre plusieurs variantes de code ou entre chemins de code.

La plupart de ces techniques exigent un haut degré d'expertise de la part du développeur, qui utilise souvent des correctifs de code ad hoc et artisanaux pour tenter de les rendre plus sûrs.
Dans un premier travail que nous avons effectué, en collaboration avec l'Université de Saint Andrews, nous avons adopté une approche différente, en nous appuyant sur le concept de *ladderisation*, inspiré des échelles de Montgomery. Nous avons proposé dans [Bro+2022] une technique et un outil semi-automatiques, pour générer des contre-mesures aux attaques par canaux auxiliaires. Notre technique, qui s'adresse aux développeurs non spécialisés, permet de refondre (une classe de) programmes C en des équivalents fonctionnels (et même équivalents algorithmiques) dotés de propriétés de sécurité améliorées. Notre approche propose des *refactorings* qui transforment le code source en son équivalent *ladderisé*, grâce à un système de réécriture vérifié sous-jacent, basé sur des types dépendants. Notre système de réécriture trouve automatiquement des réécritures d'expressions C sélectionnées, facilitant la production de leurs équivalents *ladderisés*, pour un sous-ensemble de C.
Nous avons expérimenté notre approche sur un certain nombre d'exemples représentatifs du domaine cryptographique, sur lesquels elle a effectivement amélioré la sécurité.

Les attaques par canal auxiliaire sont par définition rendues possibles par la fuite d'informations des systèmes informatiques par le biais de propriétés non fonctionnelles telles que le temps d'exécution, l'énergie consommée, les profils de puissance, etc. Il est particulièrement difficile de se protéger contre ces attaques, car elles reposent sur des mesures physiques qui ne sont généralement pas envisagées lors de la conception des propriétés fonctionnelles d'un programme. Les contre-mesures sont généralement dédiées à la protection d'un programme particulier contre une attaque particulière, ce qui manque d'universalité. Pour aider à combattre ces menaces, nous proposons avec Yoann MARQUER dans [MZH2022] la méthodologie de l'indiscernabilité, une nouvelle méthodologie pour quantifier sans connaissance préalable les fuites d'information des programmes, fournissant ainsi au développeur des mesures de sécurité précieuses.
Notre approche originale considère le code à analyser comme une boîte noire, seules les entrées publiques et les fuites étant observées. Elle peut être appliquée à différents types de fuites par canaux auxiliaires : temps, énergie, puissance, électromagnétisme, etc.
Notre méthodologie d'indiscernabilité a des fondements formels solides basés à la fois sur la sécurité topologique (avec des distances définies entre les observations dépendant du secret) et sur la théorie de l'information (quantifiant les informations secrètes restantes après l'observation par l'attaquant). Nous avons démontré

l'applicabilité de notre approche en fournissant des résultats expérimentaux pour les fuites de temps et de puissance, et en étudiant à la fois le cas moyen, le cas le plus défavorable et les métriques d'information indiscernables. Les travaux sur ce sujet sont à finaliser et à publier.

Ces travaux ont été eux aussi perturbés par les difficultés, chronophages et énergivores, que j'explique en sections 2.7 et 2.8.

2.7 REPI de TAMIS

Mots-clefs : management, direction d'équipe, direction de la recherche
Période : 10/2018 – 10/2020

J'ai été REPI (Responsable d'Equipe-Projet Inria) de l'équipe TAMIS du 12/10/2018 au jusqu'au 01/10/2020. Il s'est agi d'une activité managériale et humaine extrêmement intense et particulièrement chargée en défis.

J'ai été nommé en urgence REPI de TAMIS à la place du précédent REPI visé par des accusations de harcèlement. Mon travail a consisté à maintenir l'équipe à flots et lui permettre de fonctionner malgré la tempête, à protéger autant que possible ses membres, notamment les non permanents, notamment les femmes, et à gérer et stabiliser des situations individuelles professionnelles parfois très compliquées. J'étais en première ligne, tous les jours. Ceci s'est effectué en bonne coordination et avec le soutien des directions du centre Inria Rennes, de l'IRISA et la DG d'Inria.

Si être REPI à Inria n'est pas très original en soi, les conditions dans lesquelles j'ai eu à exercer cette charge ont, elles, été à la fois très originales et très difficiles. Conditions très originales car c'était la première fois en plus de 50 ans d'existence d'Inria qu'une telle situation se présentait. Conditions très difficiles car une équipe dont certains membres vivent une situation de grande souffrance n'est évidemment pas dans une atmosphère sereine. La prise en compte et l'accompagnement des situations humaines et psychologiques des membres de l'équipe a évidemment été également une part cruciale de mon travail de REPI. Le manque d'information et les conditions adverses ont également fortement compliqué mon travail au quotidien.

Néanmoins, la coopération avec des membres de l'équipe, des services du centre Inria Rennes, et les directions locales et nationales, a été sans faille. Malgré la situation très dégradée décrite ci-dessus, sous ma direction l'équipe a pu être maintenue à flots avec succès, la recherche dans TAMIS continuée, les contrats de recherche honorés, et 7 thèses soutenues.

Ce sont ces thèses finalement soutenues avec succès (dans TAMIS ou par la suite dans DiverSE et dans EMSEC/CAPSULE), qui me semblent être le marqueur de réussite le plus important de mon travail dans cette période. Je citerai donc les thèses de Tristan NINET, Lamine NOUREDDINE, Cassius DE OLIVEIRA PUODZIUS (dont j'ai été REPI TAMIS et que j'ai co-dirigées et co-encadrées), de Duy-Phuc PHAM (dont j'ai été REPI TAMIS et que j'ai co-encadrée), et de Delphine BEAULATON, Nisrine JAFRI, Christophe GENEVEY-METAT (dont j'ai été REPI TAMIS, mais aux travaux desquelles je n'ai pas participé). Ceci me semble être une validation claire de l'impact positif du travail que j'ai fait à ce poste, tout comme le fait que je suis resté en excellents termes avec la plupart des anciens membres de l'équipe.

Les publications des membres de l'équipe TAMIS (pendant qu'ils étaient dans TAMIS ou peu après dans DiverSE et dans EMSEC/CAPSULE) peuvent être retrouvées via les outils bibliométriques (HAL, ...), et constituent une diffusion de la recherche menée (entièrement ou en partie) pendant la période où j'ai été REPI de TAMIS.

Une fois la situation globale stabilisée, il a été décidé en commun accord avec la direction du centre de répartir les membres de TAMIS dans deux équipes (EMSEC puis CAPSULE pour les membres travaillant le plus avec Annelie HEUSER, DiverSE pour les membres travaillant le plus avec moi), et de fermer TAMIS au 01/10/2020, date de fin de ma fonction de REPI.

(On notera en ce qui me concerne personnellement que j'ai été arrêté pour épuisement du 12/12/2019 au 24/01/2020, ce qui est le plus long, et de loin, des très rares arrêts maladies que j'ai eu dans ma vie. Je pense donc avoir fait pas mal d'efforts en tant que REPI TAMIS pour mener l'équipe hors de la tourmente.).

Enfin, cette situation que j'ai contribué à résoudre comme indiqué ci-dessus aura donné lieu à la Décision N° 2019/084/00, Réf. Gedél n° 13463, du 06/03/2019, de révocation d'un ancien REPI, diffusée à tout Inria, et aura résulté en un renforcement très concret des mesures de prévention et de lutte contre le harcèlement (par exemple, à Rennes : campagne de communication, formations obligatoires).

2.8 Coordinateur du projet EU H2020 "TeamPlay"

Mots-clefs : management, direction de projet collaboratif, direction de la recherche
Période : 01/2018 – 06/2021

J'ai été responsable coordinateur du projet EU H2020 "TeamPlay" (Time, Energy and security Analysis for Multi/Many- core heterogenous PLAtforms), démarré le 01/01/2018 et terminé avec succès le 30/06/2021 (https://www.teamplay-h2020.eu/). Il s'est donc agi d'une activité managériale de coopération européenne.

Le projet comportait initialement 11 partenaires de 7 pays (Allemagne, Danemark, Espagne, France, Grèce, Pays-Bas, Royaume-Uni), dont 6 académiques et 5 industriels (1 grand groupe, 1 ETI, 3 PME), pour un financement global par l'UE d'environ 5,4 Meuros dont 621 Keuros pour Inria - EPI TAMIS.

J'ai mené la construction de la proposition de projet avec les partenaires, que pour la plupart je connaissais via le réseau HiPEAC mais avec la plupart desquels je n'avais jamais travaillé auparavant. J'ai notamment dû faire particulièrement attention à la cohérence globale et aux contributions de chacun. La proposition a été acceptée.

J'ai ensuite coordonné le projet pendant toute sa durée (3,5 ans), animant les réunions plénières mensuelles en visioconférence, les réunions plénières annuelles en présentiel (modulo le covid...), ainsi que les réunions plus restreintes se focalisant sur la cybersécurité.

En parallèle de ce travail de coordination, j'animais les travaux dans TAMIS portant sur les aspects cybersécurité, ainsi que plateforme/infrastructure, recrutant et encadrant 1 postdoctorant et 3 ingénieurs sur tout ou partie de la durée du projet.

Il s'agissait de la première coordination d'un projet européen pour moi.

Un gros défi a consisté à maintenir une cohésion de projet, puisqu'une fois le financement acquis, les partenaires ont eu une tendance à se refocaliser sur leurs propres travaux en interne, en oubliant un peu le projet et la coopération avec les autres. En imposant des réunions plénières fréquentes (plus que souhaité par les partenaires), en rappelant constamment le contexte du projet ("The big picture") et l'intérêt commun, et en insistant sur la nécessité d'être cohérents et cohésifs, en combattant les dérives hors du cadre du projet, j'ai pu obtenir une bonne cohésion de groupe et une ambiance agréable avec presque tous les partenaires après environ 6 mois. Ceci nous a permis de travailler dans de bonnes conditions, malgré le nombre de partenaires assez important et la présence de cultures et environnements très divers.
Une difficulté importante a consisté à obtenir que tous les partenaires fournissent ce à quoi ils s'étaient engagés. Pour un partenaire, il a fallu argumenter pour que les ingénieurs impliqués aient officiellement du temps libéré pour le projet. Pour un deuxième, il a fallu faire intervenir les mécanismes de régulation par le consortium prévus dans l'accord de consortium. Un partenaire n'a plus fait partie du consortium à partir du 13/10/2020. Ces aspects "extrêmes" de la gestion de projet ont eu retentissement négatif certain sur ma propre production scientifique (le reste du projet et des partenaires ayant été relativement préservés).

La validation de mes actions de management de ce projet vient en premier des services juridiques d'Inria, de la DG Inria et du Project Officer européen qui m'ont soutenu. Elle vient surtout au final de la clôture avec

une très bonne revue finale du projet TeamPlay, avec de nombreuses publications au global (qui continuent encore aujourd'hui en 2023, ce qui montre la solidité des liens de travail tissés dans le projet). Les partenaires de TeamPlay à qui j'ai proposé en 2022 de déposer un projet coordonné par moi (avec quelques nouveaux partenaires soigneusement choisis) ont tous accepté sans hésiter, ce qui est une validation supplémentaire de mon management. Comme nous avons échoué de peu en 2022, nous re-soumettons en 2023, avec les mêmes partenaires de TeamPlay (plus des nouveaux). Le réseau créé est donc solide.

J'ai ainsi appris énormément sur la coordination de projet, et sur comment choisir ses partenaires. Paradoxalement, les situations compliquées que j'ai eu à gérer ont également contribué à forger une relation encore plus solide entre les membres du consortium, d'où la solidité du lien de travail que j'indique ci-dessus.

Les résultats de ma coordination du projet TeamPlay sont concrètement visibles sur le site du projet, notamment par les publications scientifiques (visibles sur la page https://www.teamplay-h2020.eu/index.php?page=scientific-publications) et les outils mis à dispositions (https://www.teamplay-h2020.eu/index.php?page=documentation). Le fait que des résultats scientifiques et techniques du projet [Rou+2023] ont été intégrés dans les pratiques ou outils des partenaires industriels est un impact très important mais hélas moins visible à l'extérieur, à l'exception de l'EnergyAnalyser, intégré aux outils commercialisés par Absint, un peu plus visible (https://www.youtube.com/watch?v=VhK3WpTNfYk).

2.9 HiPEAC et HiPEAC Vision ⇒

Mots-clefs : projet collaboratif, direction de la recherche, prospective scientifique, prospective technologique, prospective sociétale
Période : 2016 ⇒

HiPEAC est un réseau d'excellence européen qui existe sans discontinuer depuis 2004 (https://www.hipeac.net/). Nous sommes depuis 2023 dans sa 7ème incarnation, HiPEAC-7. HiPEAC fut d'abord le European Network of Excellence on High-Performance Embedded Architecture and Compilation, puis plus récemment devint une CSA (Coordination and Support Action), puis fut renommé HiPEAC (High Performance, Edge And Cloud computing). Il s'agit cependant bien, malgré des évolutions dans la composition des partenaires du projet, de la même communauté. HiPEAC est coordonné depuis de nombreuses années par le Prof. Koen DE BOSSHCHERE, de l'Université de Gand (BE). HiPEAC a notamment crée une conférence annuelle HiPEAC (https://www.hipeac.net/events/#/conference/), d'excellent niveau scientifique, qui rassemble environ 550 personnes par an, et l'école d'été annuelle ACACES (https://www.hipeac.net/events/#/acaces/), limitée à 200 étudiants par an. Son rôle de structuration de communauté et de création de réseau de connaissances est majeur pour les scientifiques européens des domaines concernés. HiPEAC produit également la HiPEAC Vision (cf. ci-dessous).

Je suis membre d'HiPEAC depuis 2006, et suis le PI d'Inria pour HiPEAC et membre du Steering Committee d'HiPEAC depuis 2018. Je suis également membre du comité éditorial de la « HiPEAC Vision » (HiPEAC Vision Editorial Board, environ 6-8 personnes selon les années) depuis 2016. Je suis co-auteur de toutes les HiPEAC Visions produites depuis.

Une première forme de mon activité liée à HiPEAC est ma participation au Steering Committee de HiPEAC (réunions mensuelles), où je représente Inria avec assiduité, et participe aux décisions concernant la vie d'HiPEAC, ses actions, ses membres, ses orientations. Il s'agit là d'un travail important pour animer la communauté des chercheurs associés à HiPEAC.

Une seconde forme de mon activité, qui est clairement la plus chronophage mais aussi celle qui a le plus d'impact, concerne la Vision d'HiPEAC (https://www.hipeac.net/vision/). Il s'agit d'un document, fourni aussi bien sous forme imprimée que via un site web, de prospective scientifique, technique et sociétale, qu'HiPEAC livre à l'UE chaque année depuis 2023 (tous les deux ans avant). La HiPEAC Vision porte sur un spectre assez large de l'IT (HPC, cloud, edge, IoT, langages, compilation, sécurité, architecture logicielles

et matérielles, interfaces matériel-logiciel, technologies matérielles...). Il ne s'agit pas d'articles académiques proprement dits, mais d'articles devant faire l'état de l'art et surtout la prospective sur les domaines concernés, à plus large public (technique, grand public averti, politiques). Cette prospective doit être replacée dans un cadre à la fois mondial et européen, pour replacer l'UE dans son contexte, voir ses forces, ses faiblesses, les directions dans lesquelles elle devrait investir.

Une première partie de mon travail sur la HiPEAC Vision, liée à sa production, consiste avec le reste de l'Editorial Board de la HiPEAC Vision à définir la coloration du document global, son architecture et sa structure, à proposer des experts issus des communautés ciblées et à organiser des réunions de consultation (5-6 journées de réunions par an, typiquement), à trouver des auteurs pour traiter différents sujets que nous avons décelés comme étant d'intérêt (qui peuvent être les experts précités ou d'autres), à écrire moi-même certains articles, à participer au "reviewing" des articles, à participer à l'écriture des parties "communes" de la HiPEAC Vision.

Une seconde partie de mon travail sur la HiPEAC Vision, liée à son exploitation et sa diffusion, consiste avec le reste de l'Editorial Board à diffuser et expliquer la Vision dans les différents événements que nous organisons ou auxquels nous participons, à produire ou diriger la production des supports de diffusion (présentations, BDs, vidéos) et surtout et avant tout à interagir avec l'UE (responsables d'unités, responsables de DG, notamment la DG Connect qui chapeaute actuellement HiPEAC) pour lui expliquer la HiPEAC Vision et lui permettre de se l'approprier, pour alimenter sa réflexion sur les appels à projets de recherche futurs qu'elle émettra. En ce sens, il s'agit d'une contribution à la structuration de l'espace de recherche européen, tant académique qu'industriel. Depuis 2023, nous avons commencé une action d'interaction avec d'autres réseaux et organismes européens fédérant des communautés différentes mais liées (INSIDE, EPoSS, ETP4HPC, EUCloudEdgeIoT.eu, Destination Earth, AIOTI, BDVA, NESSI, ECSO, ...), afin de coordonner ces différents réseaux, d'éviter les duplications, et de participer à mieux structurer l'espace de recherche européen en informatique.

Une originalité de cette activité est liée à l'originalité d'HiPEAC, qui est le seul réseau d'excellence en informatique à ma connaissance à exister sans discontinuer depuis 19 ans. Le mérite ne m'en revient pas, bien sûr, puisque je suis dans le Steering Committee depuis 2018. Un autre point d'originalité, que je trouve particulièrement motivant, est la production d'éléments de prospectives pour guider l'UE dans ses choix scientifiques et technologiques pour le futur.

La difficulté de ce travail est intrinsèque à la prospective à 10 ans que nous visons, dans un domaine en constante et rapide évolution (parfois même très rapide voire disruptive), avec des acteurs privés ayant un fort impact. Ceci rejoint bien entendu le travail de recherche scientifique qui est le cœur de notre métier, mais se fait parfois à plus large grain, sur des domaines plus variés. Une autre difficulté consiste à trouve le bon niveau entre le scientifique et académique pur (article de recherche) et le niveau vulgarisation (être compréhensible par le plus grand nombre, notamment les politiques qui prennent les décisions pour les programmes de recherche à l'UE). Enfin, une difficulté consiste à produire une HiPEAC Vision qui aie une cohérence globale et une clarté (une vision, donc...) qu'on trouve moins fréquemment dans les articles scientifiques académiques qui sont souvent plus focalisés.

L'impact du réseau HiPEAC dans son ensemble est toujours difficile à mesurer dans son entièreté, mais le fait qu'il compte 891 membres individuels actifs issus de 518 institutions, que la conférence annuelle HiPEAC soit du meilleur niveau et attire environ 550 participants, tandis que l'école d'été ACACES reçoit depuis 2005 environ 280 candidatures pour 200 places, sont des indices factuels permettant d'apprécier son succès. Une validation de l'intérêt et de l'impact de la HiPEAC Vision vient du fait que l'UE continue à renouveler HiPEAC depuis 2004 (ma contribution jouant depuis 2016), avec un intérêt majeur pour la HiPEAC Vision. Un certain nombre d'appels à projets de l'UE ont correspondu à des recommandations faites antérieurement dans la HiPEAC Vision, quelques appels reprenant même des formulations très proches des nôtres.

En ce qui concerne ma contribution spécifiquement, une première validation de mon travail est le fait que je continue à contribuer à la HiPEAC Vision et à faire partie du HiPEAC Vision Editorial Board depuis 2016,

alors que sa composition a varié. Un autre témoignage de la solidité et de l'impact de ma contribution vient du fait que, lorsque que la proposition de projet européen HiPEAC-7 a été élaborée en 2022, le coordinateur d'HiPEAC depuis de longues années Koen DE BOSSCHERE a décidé de changer sensiblement les partenaires (5 nouveaux sur 11), pour amener du sang neuf et une énergie renouvelée, mais il a "decided to keep Inria because you Olivier have a very strong contribution in the HiPEAC Vision". Ce point a été de plus confirmé lorsqu'à la clôture d'HIPEAC-6 en 2023 [4], il a décidé d'attribuer à Inria une part plus importante du reliquat de budget qu'aux autres partenaires, considérant "That seems fair given the large voluntary contribution of INRIA [moi en pratique] to the vision work."

Le travail résultant d'HiPEAC, et donc du Steering Committee dont je fais partie, et notamment les événements organisés par le réseau se retrouvent sur https://www.hipeac.net, la HiPEAC Vision pouvant être naviguée ou téléchargée en un PDF global (ce que je recommande pour mesurer prendre l'ampleur de travail) à partir de https://www.hipeac.net/vision (+ archives en haut à droite).
HiPEAC Vision 2023 : https://www.hipeac.net/vision/2023.pdf
HiPEAC Vision 2021 : https://www.hipeac.net/vision/2021.pdf
HiPEAC Vision 2019 : https://www.hipeac.net/vision/2019.pdf
HiPEAC Vision 2017 : https://www.hipeac.net/vision/2017.pdf

2.10 Conclusion sur mon activité de recherche et de direction de la recherche

J'ai dans ce chapitre essayé de donner un aperçu clair de mes activités de recherche et de direction de la recherche jusqu'à présent, de leur chronologie (voir notamment le résumé d'activité en section 2.1.) et de la logique de leur enchainement.
Le tronc commun en a été la production de logiciel, au début via l'analyse de programmes et la compilation, avec l'exploration de nombreuses ramifications qui m'ont permis d'avoir une vision assez large : optimisation en vitesse, mémoire, énergie ; instrumentation ; visualisation avancée ; analyses de binaires pour la détection et la classification de packers et de maliciels ; etc.
J'ai également mis un certain accent sur les éléments permettant d'apprécier ma capacité à diriger les recherche, puisque l'objet de ce document est précisément ma candidature à une Habilitation à Diriger des Recherches.

Le chapitre suivant présente mon programme de recherches futures, avec deux branches toujours issues de ce tronc commun.

4. HiPEAC-6 et HiPEAC-7 se recouvrent un peu temporellement pour donner de la flexibilité pour l'utilisation des fonds, ce qui est une forme de « passation » de fonds restant d'HiPEAC-6 sur HiPEAC-7.

Chapitre 3

Synthèse de mon programme de recherche

Production de logiciels plus sécurisés

3.1 Introduction à mon programme de recherche

Après la synthèse de mes activités passées, dans le chapitre précédent, celui-ci présente mon programme de recherche, portant sur la production de logiciels plus sécurisés.

Mes travaux s'inscrivent dans les deux domaines : celui du génie logiciel au sens large, avec la production des programmes, et celui de la cybersécurité. Ces deux domaines sont, chacun pris indépendamment, très larges. Les travaux qu'il est possible d'y mener sont donc pléthoriques. De même, bien qu'évidemment moins nombreuses, les continuations possibles de travaux que j'ai menés sont assez nombreuses. Je ne présente cependant pas de projet pour tous celles-ci, principalement dans un souci d'utilisation optimale du temps de chacun, et pour faire preuve de réalisme sur le nombre de voies qu'il est possible de creuser.

Je présente donc ici deux axes de recherches futures qui me tiennent particulièrement à coeur et qui présentent une continuité avec des travaux que je dirige déjà ou auxquels je participe déjà. Le premier, en section 3.2 concerne la définition et intégration explicite des propriétés, non-fonctionnelles, de sécurité dans les logiciels et les outils de développement. Le second, en section 3.3 porte sur la sécurisation de la production des chaines de build.

3.2 Définition et intégration explicite des propriétés, non-fonctionnelles, de sécurité dans les logiciels et les outils de développement

3.2.1 Motivations, concepts et vision

Le premier projet de recherche futur que je présente m'est assez personnel, bien que commencé dans le cadre d'une coopération large et internationale, à savoir le projet H20202 TeamPlay que j'ai coordonné : La définition et l'intégration explicite des propriétés, non-fonctionnelles, de sécurité dans les logiciels et les outils de développement.

En effet, bien que la sécurité soit un problème de plus en plus important dans des domaines allant des capteurs à petite échelle aux centres de données, en passant par l'Internet des objets, les systèmes cyber-physiques intelligents et les appareils mobiles toujours plus nombreux, elle est généralement considérée comme une préoccupation secondaire. Ceci est dû au fait que la (bonne) sécurité étant difficile à voir et montrer, elle n'est que très difficilement "vendeuse". C'est seulement en cas de brèche de (mauvaise) sécurité que la notion de coût (et donc de bénéfice) apparait au grand jour, mais souvent trop tard.

Il y a donc un besoin clair et urgent d'**exposer la sécurité aux concepteurs et aux développeurs de systèmes informatiques**. L'idée fondamentale est qu'il est maintenant nécessaire de transformer la sécurité, sous ses aspects multiples, en citoyenne de première classe, c'est à dire visible, apparente, explicitement, pas implicitement ni auxiliairement. Cela permettra aux programmeurs (non experts) de les comprendre et de les manipuler directement. Ceci aidera à diminuer voire à supprimer l'utilisation complexe, longue et sujette aux erreurs des outils extrinsèques ainsi que les tests continus à grande échelle pour les aspects sécurité lors du processus d'ingénierie logicielle et, par conséquent, rendra ce dernier beaucoup plus simple, agile et productif. En exposant les propriétés de sécurité en tant que citoyens de première classe et en les rendant faciles à utiliser et à comprendre pour les développeurs, nous pourrons éliminer le besoin de double expertise : le développeur sera capable de manipuler et de raisonner avec précision sur les aspects de sécurité comme des valeurs normales du programme, en interaction directe avec des outils d'analyse et d'optimisation.

Pour créer des logiciels sécurisés, les programmeurs doivent pouvoir participer activement aux décisions concernant la sécurité, plutôt que d'être informés passivement des ses effets (ou plus souvent, des effets de l'absence de sécurité). Cela s'applique aussi bien à la conception qu'à la mise en œuvre et à l'exécution.

Je souhaite donc dans mes recherches travailler sur des idées nouvelles qui permettent d'intégrer des propriétés extra-fonctionnelles de sécurité dans les logiciels. Ainsi **les développeurs pourront être en mesure de traiter directement les propriétés de sécurité clés du système, explicitement**, et non plus comme un aspect annexe.

Cependant, si les propriétés fonctionnelles des programmes sont bien explorées, et si certaines propriétés non-fonctionnelles comme le temps et l'énergie sont relativement bien comprises, car elles correspondent à des réalités physiques connues, mesurables, il en est tout autrement des propriétés de sécurité. Il est en effet bien plus difficile de quantifier les performance en sécurité d'une application.

Il faudra pour cela d'abord définir la notion de sécurité, ou plus exactement, les différents aspects que recouvre la notion de sécurité, car celle-ci est vraiment plurielle.

3.2.2 Objectifs

Ce projet vise à développer de nouvelles techniques formellement motivées qui permettront de traiter efficacement les propriétés de sécurité importantes des logiciels, et de les traiter comme des citoyens de première classe, explicitement. Il faut permettre aux programmes de "réfléchir" directement sur leur propre sécurité, etc., tout en permettant au développeur de raisonner sur ces mêmes propriétés de sécurité au niveau de son code source. Cela permettra également des progrès significatifs pour maintenir un bon équilibre avec d'autres paramètres logiciels importants, y compris le temps, l'énergie, la taille du code, la complexité, etc.

Les objectifs concrets de mon projet de recherche sont :

Expressivité et Contrats : *Mettre au point de nouveaux mécanismes, ainsi que leurs fondements théoriques et pratiques, qui appuient le raisonnement direct sur la sécurité au niveau du langage.*
Les contrats sur la sécurité seront explicités directement via des constructions au niveau du code source, dans le but de pouvoir à terme vérifier ces contrats de manière entièrement automatique. Cela permettra aux programmeurs d'exprimer des contraintes et d'obtenir un retour d'information sur la sécurité directement au niveau du code source, les libérant ainsi de la nécessité de raisonner sur ces propriétés non fonctionnelles à des niveaux d'abstraction inférieurs comme le code machine. En même temps, la technologie de compilation garantira que tout raisonnement au niveau du langage sur les propriétés non fonctionnelles de sécurité coïncide effectivement avec le comportement réel à l'exécution.

Interrelations : *Déterminer les interrelations fondamentales entre les optimisations liées à la sécurité, et celles liées à d'autres propriété non-fonctionnelles comme le temps, l'énergie, etc.,*
Cela permettra de trouver quelles approches d'optimisation de la sécurité sont les plus efficaces en combinaison avec les autres critères, soit au niveau du pire cas, soit au niveau du cas moyen.

Modèles : *Développer des modèles des propriétés de sécurité permettant un raisonnement et une optimisation de haut niveau pendant le développement du système, et en cours d'exécution.*
Les modèles des propriétés de sécurité sont au cœur de l'approche car ils permettent la transparence sur l'ensemble du système. En capturant et en transportant les propriétés de sécurité aux différents niveaux d'abstraction au sein de la pile système, y compris potentiellement les compteurs matériels, l'architecture du jeu d'instructions et la représentation intermédiaire du compilateur, il deviendra possible de développer des techniques avancées d'analyse, d'optimisation, de gestion, de planification d'exécution et de configuration dynamique du matériel en fonction des caractéristiques et des besoins en sécurité.

Analyses : *Élaborer des analyses statiques et dynamiques capables de déterminer les informations sur la sécurité des fragments de code de manière à éclairer les programmes à haut niveau, afin d'assurer la transparence en matière de sécurité au niveau du code source.*
Les modèles des propriétés de sécurité qui seront développés dans mes recherches fourniront des données pour une analyse statique avancée qui permettront de prédire certaines propriétés de sécurité sans exécution sur le matériel cible. Ceci éliminera ou réduira considérablement le coût à l'exécution.

Fourniture d'outils : *Intégrer ces modèles, analyses et outils dans une boîte à outils basée sur l'analyse, capable de présenter au programmeur des informations pertinentes sur les aspects de sécurité.*
La boite à outils qui en résultera soutiendra l'ingénierie logicielle sensible à la sécurité en établissant des mécanismes permettant aux programmeurs d'interagir directement avec les contraintes liées à la sécurité au niveau du code source.

Pertinence industrielle : *Déterminer les exigences et les paramètres pertinents pour l'industrie et évaluer l'efficacité et le potentiel de ma recherche à l'aide de ces paramètres et exigences.*
Afin de garantir un véritable impact et une utilisabilité réelle de la boite à outils de ce projet, elle devra être intégrée dans un ou des IDEs et chaines de développements communément utilisées par l'industrie, ce qui impliquera un TRL assez élevé. Je serai donc attaché à favoriser les collaborations industrielles sur ce sujet.

3.2.3 Défis

Afin de réaliser la vision décrite ci-dessus, un certain nombre de défis techniques clés doivent être relevés.

Dans un premier temps, les défis à relever concerneront les propriétés de sécurité elles-mêmes :

- Concevoir des **mécanismes expressifs** capables d'exposer les aspects de sécurité comme des valeurs à part entière, d'une manière compréhensible pour les programmeurs, mais en même temps assez puissants et expressifs pour couvrir toutes les propriétés pertinentes. Quelles seront les propriétés, les métriques, de sécurité pertinentes (ie utiles) et accessibles (ie quantifiables, exprimables) ? Quelle sera la sémantique précise (ie compréhensible, et portable) de chaque propriété choisie ? La résolution de ces problèmes représente un défi intellectuel et technique majeur.
- Développer des **modèles** de propriétés de sécurité et les **analyses** qui permettent de les déterminer et/ou de les vérifier automatiquement. Les mesures lors de l'exécution prennent en général beaucoup de temps et nécessitent des compétences spécialisées que l'on ne trouve généralement pas chez les ingénieurs logiciels. Pour surmonter ce problème, des modèles fiables pour les différents aspects de sécurité sont nécessaires. Capturer l'utilisation des ressources matérielles et élaborer des analyses avancées qui utilisent ces modèles pour estimer avec fiabilité les propriétés de sécurité et les vérifier de façon fiable constituent des défis techniques majeurs pour la plupart des architectures matérielles utilisées dans la pratique.
- **Extraire les informations** permettant de quantifier la sécurité. Cela comprend l'information micro-architecturale, ainsi que l'exposition de l'information de haut en bas de la chaîne d'outils. Or depuis plusieurs décennies, le génie logiciel a fait abstraction de presque toute notion concrète de matériel, dans le but d'accroître la portabilité, la productivité et la réutilisabilité. Ceci a résulté en un empilement de nombreuses couches d'abstraction et d'indirection. L'inconvénient de cette abstraction

profonde est que la plupart des ingénieurs logiciels sont absolument inconscients de la façon dont les algorithmes, les données et leur encodage influencent la sécurité de leur code. L'informatique sécurisée ne peut devenir une réalité que si les différentes couches d'abstraction de la pile système sont brisées sans sacrifier les avantages de productivité, c'est-à-dire si il est possible de fournir une visibilité totale de la sécurité. Cet objectif ne peut être atteint que grâce à l'effort concerté avec des chercheurs de disciplines traditionnellement séparées : conception du matériel, logiciels système, modèles de programmation, langages, compilateurs, concepteurs d'outils et méthodologies.

A plus long terme, les défis à relever concerneront les propriétés de sécurité en interaction globale :

- Développer des techniques d'**optimisation multi-critères** novatrices qui permettent de prendre en compte la sécurité tout autant que les autres critères d'optimisation plus classiques, comme la vitesse, le WCET ou la consommation énergétique, en tenant compte des exigences du système. En effet, bien qu'il existe un grand nombre de travaux sur l'optimisation des logiciels, la prise en compte simultanée de critères non fonctionnels multiples lors de l'optimisation n'a pas encore été étudiée de manière systématique et complète. Les défis qui se présentent dans ce contexte sont donc de trouver des approches d'optimisation calculables, capables d'améliorer systématiquement des critères non contraignants tels que, par exemple, la performance moyenne ou la consommation d'énergie tout en garantissant le respect de contraintes strictes en termes de performance, de consommation d'énergie et de niveau de sécurité. Les optimisations multi-objectifs envisagées seront à la fois statiques et dynamiques : les optimisations statiques seront réalisées par le compilateur, les optimisations dynamiques par l'environnement d'exécution et par le système d'exploitation.
- Etudier des **mécanismes d'exécution** pour faire appliquer dynamiquement les règles de sécurité à l'échelle du système. Les systèmes modernes hébergent une multitude d'applications de différents développeurs, tous avec leur propre compréhension locale des propriétés de sécurité - ou même avec un manque total de compréhension de ces propriétés. Le défi est qu'un environnement d'exécution ne doit pas seulement être conscient des objectifs au niveau du système, mais doit également être capable de les appliquer en manipulant et en reconfigurant l'ordonnancement des applications individuelles du système en tenant compte de leurs besoins en sécurité.

3.2.4 Approches et avancées attendues

Notre approche vise à **permettre la manipulation directe des propriétés de sécurité, par les développeurs et par les outils**, et à donner des garanties statiquement vérifiables sur la sécurité. Elles apparaitront sous la forme d'annotations, ou DSL (Domain Specific Language), intégrées au langage hôte et à son système de types, et passeront donc ainsi à travers tous les outils de la chaine de développement.

La première question à résoudre sera celle des propriétés de sécurité elles-mêmes. En choisissant explicitement ce qui sera exprimable, avec quelle sémantique, elle ouvrira de nouveaux champs de possibilités. La définition d'un catalogue des propriétés de sécurité exprimables et disponibles sera sans doute une petite révolution culturelle, comme l'ont été en leur temps les design-patterns, par exemple. La définition de chacune de ces propriété pourra cependant être elle-même une avancée considérable, car il sera nécessaire d'en définir une sémantique précise, ce qui est actuellement loin d'être le cas.

Le simple fait de faire apparaitre la sécurité sous une forme explicite dans le code source des programmes est en soi une petite révolution. Cela implique que les développeurs vont être capables de consulter et manipuler explicitement, en écrivant des instructions dans le langage source, les valeurs des différentes propriétés de sécurité, pour prendre des décisions dans le programme. Ainsi, par exemple, un développeur pourra écrire que si le niveau de menace d'un environnement est trop important, on choisira des stocker des information plutôt que de transmettre, tandis que si le niveau est moins menaçant on pourra transmettre les informations avec un certain niveau de sécurité du chiffrage, etc. En coordination avec d'autres propriétés non-fonctionnelles, cela permettrait des décisions explicites de qualité de service au niveau du programme : s'il reste plus de E quantité d'énergie et T temps, alors prendre l'algorithme de chiffrement le plus solide du point de vue de la sécurité, mais le plus couteux en énergie et temps, sinon prendre un algorithme moins robuste mais moins couteux, etc.

Cela permettra également d'avoir des bibliothèques de composants réutilisables garantissant un certain niveau de sécurité dans un domaine particulier, facilitant ainsi la compositions d'applications sécurisées à partir de ces briques contractualisées. La valeur ajoutée en terme de génie logiciel sera alors considérable.

En ce qui concerne la sécurité prise "isolément", nous nous concentrerons particulièrement sur les attaques par canaux auxiliaires, en considérant l'énergie et le temps comme des sources exploitables d'information. Plus précisément, dans le cas d'applications sensibles du point de vue de la sécurité, les attaques par canaux auxiliaires sont capables de révéler des données traitées sensibles en mesurant par exemple l'énergie, le profil de puissance, ou le temps du système pendant son exécution. En développant une modélisation fiable de l'énergie et du temps, il sera possible de définir les faiblesses de sécurité *intrinsèques* au système à travers tous les niveaux d'abstraction. En particulier, la modélisation précise de chaque composant d'un système présente un intérêt particulier pour l'analyse des canaux auxiliaires. En effet, en raison de la grande quantité de bruit, les techniques extrinsèques d'évaluation de la sécurité par observation, qui ne reposent pas sur une modélisation suffisamment fine des informations fuitant hors du sytème, pourraient échouer alors même que de l'information fuite, et ainsi sous-estimer le risque de sécurité. Les techniques que nous développons dans nos travaux actuels, comme l'indiscernabilité, sont du plus grand intérêt ici.
En plus d'évaluer le risque de sécurité, nous considérerons des contre-mesures appropriées basées sur des techniques de dissimulation adaptées de manière automatisée.

Il existe peu de travaux sur le traitement de la sécurité comme une propriété intrinsèque d'un programme, un contrat, garantissant la sécurité par conception et par construction par rapport à des classes entières de menaces et d'attaques (par opposition à répondre à des menaces et attaques spécifiques une fois le programme construit). La nouveauté de l'approche que je vise ici est de chercher à exprimer formellement des contrats de sécurité vérifiables, générés à partir d'un programme d'entrée dans un langage de haut niveau. Il sera ensuite possible de vérifier ces contrats par des mécanismes normaux de vérification, non pas en vérifiant qu'un modèle du programme satisfait aux contraintes de sécurité, mais bien en vérifiant le programme source lui-même satisfait à ces contraintes, pour toutes les entrées possibles. Ces contraintes exprimées sous forme d'annotations/DSL sur le programme source pourront donc être vérifiées automatiquement et de manière transparente au moment de la compilation, à partir des modèles qui reflètent les réalités sous-jacentes de la sécurité, comme déterminés par notre modélisation préalable. De cette façon, les preuves mathématiques vérifiées par machine de l'exactitude du logiciel seront gérées, tout en fournissant une notation pratique et lisible pour les développeurs lambda. Une (continuation de ma) collaboration avec l'université de Saint Andrews est ici fortement envisagée.
Dans cette approche, les propriétés de sécurité sont délibérément apparentes, de première classe. L'implémentation et la spécification des propriétés de sécurité font donc nécessairement partie du même cadre ; tout changement dans la spécification doit se traduire par un changement correspondant dans l'implémentation, et vice versa. Contrairement à d'autres méthodes formelles et à d'autres approches de vérification des logiciels, tout lien entre un modèle formel d'un système et l'implantation elle-même sera préservé.

A plus long terme, et plus largement, ces travaux demandent à ce que des problèmes importants soient résolus pour l'optimisation multi-critère prenant en compte la sécurité.
Au-delà de la sécurité, un autre critère crucial, que j'ai déjà abordé dans les recherches passées, est la consommation énergétique. Or si on considère les compilateurs, et ceux qui tiennent compte de l'énergie, dans le passé, plusieurs auteurs ont montré que ces compilateurs peuvent exploiter un grand potentiel d'optimisation en termes de consommation d'énergie en utilisant des fonctionnalités matérielles dédiées. La réduction de la consommation d'énergie dynamique se fait généralement en exploitant le DVFS (Dynamic Voltage and Frequency Scaling), où la fréquence d'horloge d'un CPU est diminuée afin de permettre la réduction de la tension d'alimentation. Le DFS basée sur un compilateur a été abordée par plusieurs auteurs dans le passé. Des optimisations exploitant le DVFS et d'autres caractéristiques matérielles permettant d'économiser de l'énergie comme, par exemple, l'Adaptive Body Biasing (ABB) ont également été proposée. Ces techniques sont cependant peu discrètes, ce qui pose des problèmes de sécurité, par fuite d'information favorisant les attaques par canaux auxiliaires. Outre l'exploitation de telles fonctions matérielles dédiées d'économie d'énergie, de nombreux auteurs, dont moi, ont montré que les compilateurs peuvent exploiter un grand potentiel d'opti-

misation de consommation d'énergie en effectuant une allocation de mémoire appropriée. Malheureusement, toutes les approches pour la compilation sensible à l'énergie souffrent de certaines limitations graves : soit elles s'appuient sur la présence de fonctionnalités matérielles dédiées, soit elles sont incapables de prendre en compte à la fois la sécurité et l'énergie de manière simultanée, soit elles ne peuvent garantir que les contraintes dures sur la consommation d'énergie ou la criticité en temps réel sont respectées. De même, ces compilateurs sensibles à l'énergie ne tiennent pas compte des niveaux de sécurité lorsqu'ils prennent des décisions d'optimisation axées sur l'énergie. Ils ne considèrent essentiellement qu'un seul critère : l'énergie. Ainsi, les compilateurs actuels ne sont pas en mesure d'optimiser simultanément énergie et sécurité. Il en va de même pour les compilateurs ciblant le facteur temps qui ne supportent actuellement pas la sécurité.

Le principal problème est que les compilateurs actuels sont rarement en mesure de prendre en charge l'optimisation multicritères, ce qui empêche la génération de code optimisé à la foi pour la sécurité, l'énergie, le temps, voire d'autres critères. Les optimisations multicritères sont souvent réalisées en déterminant l'ensemble des solutions Pareto-optimales dans l'espace de solution multidimensionnel et en demandant au développeur de choisir l'une de ces solutions. Cependant, d'une part, la complexité du calcul de toutes les solutions Pareto-optimales est exponentielle au nombre de critères et, d'autre part, l'approche globale nécessite toujours l'intervention manuelle du concepteur humain.

Les travaux que j'ai coordonnés dans le passé ont montré qu'une alternative effective mais simplificatrice consiste à réduire le problème de l'optimisation multicritère à un problème monocritère avec des contraintes pour d'autres critères "durs". Par exemple, il est possible de modéliser dans le processus d'optimisation des limites strictes de sécurité, de bornes de consommation d'énergie, de temps réel, tout en minimisant les critères souples comme la performance moyenne ou la consommation énergétique. Une telle approche peut être modélisée à l'aide, par exemple, de l'ILP (Integer-Linear Programming), en faisant attention à formuler le problème de façon soigneusement restreinte pour éviter une complexité calculatoire trop forte.

Dans le domaine de l'optimisation multi-objectifs, les algorithmes évolutionnaires ont fait leurs preuves puisque leur durée d'exécution peut être limitée tout en fournissant des solutions de haute qualité. L'algorithme SPEA ou les optimisations Memetic en sont des exemples. En ce qui concerne l'état de l'art en matière de compilation et d'optimisation, on peut conclure que les quelques approches qui tentent d'équilibrer des exigences énergétiques avec des contraintes de performance n'exposent pas l'énergie au programmeur, mais la traitent comme un aspect purement système plutôt que comme un aspect intégré au langage/compilateur/système. En outre, il manque une évaluation systématique de la faisabilité d'algorithmes multi-objectifs pour l'optimisation simultanée de la sécurité, de l'énergie et du temps et de la sécurité. Cet axe de recherche implique donc d'aller bien au-delà de l'état de l'art, en recherchant des optimisations véritablement multicritères qui sont systématiquement capables de mettre en balance la sécurité avec la consommation d'énergie et le temps. A notre connaissance, une telle étude approfondie et complète de l'optimisation multicritères de la sécurité, l'énergie, et du temps est révolutionnaire et préparerait le terrain pour la génération de code à la fois sécurisé, basse consommation et rapide.

3.2.5 Conclusion

La définition et intégration explicite des propriétés, non-fonctionnelles, de sécurité dans les logiciels et les outils de développement est un axe de mes futures recherches qui est une continuité de certains de mes travaux récents (esquissés dans TeamPlay, cf. section 2.6.2), mais aussi plus largement de certains de mes travaux bien plus anciens (les aspects autours des langages, cf. section 2.3).

Cet axe de recherche amène de nombreux défis passionnants, à la croisée de plusieurs domaines. C'est également un axe qui ouvre vers des défis dans d'autres domaines, et donc a un potentiel, voire un besoin, de collaborations très élevé.

Il offre également la possibilité de grandes avancées ayant un impact fort , très concret et à large échelle dans les pratiques de production de logiciels, avec la promesse d'une amélioration sensible de la cybersécurité des logiciels, et donc une véritable utilité technique, économique et sociétale, ce qui est pour moi particulièrement enthousiasmant.

3.3 Sécurisation de la production des chaines de build

3.3.1 Motivations, concepts et vision

La numérisation et l'informatisation présentent de nombreux risques : les bogues et les failles de sécurité dans les systèmes logiciels peuvent accidentellement divulguer des données personnelles hautement privées, et les dommages causés aux infrastructures critiques peuvent tuer des personnes ou menacer la souveraineté d'un État. Les vulnérabilités des logiciels sont des défauts qui, lorsqu'ils sont exploités, peuvent entraîner des comportements indésirables et incorrects. Presque tous les logiciels présentent des problèmes de sécurité. Les attaques contournent la sécurité du logiciel hôte en exploitant ces failles [1].
Au cours des dernières décennies, de nombreuses méthodologies et outils ont été créés pour réduire les vulnérabilités ou éliminer certaines catégories de vulnérabilités. Ces méthodes comprennent l'analyse statique, l'interprétation abstraite, le développement avec compilation vérifiée, etc. Cependant, ces stratégies sont rarement employées et parfois impossibles à utiliser. Elles requièrent en effet des compétences rares et hautement spécialisées. En outre, il peut être difficile d'appliquer ces approches à des systèmes complexes. De ce fait, les grandes bases de code ne sont pas sécurisées. Il a été prouvé que la réutilisation de composants open-source dans le développement de logiciels modernes stimule l'innovation. Cependant, la diversification et la complexité des logiciels, de la chaîne de construction et de la chaîne d'approvisionnement, qui ont résulté de cette approche, ont introduit de nouveaux risques importants en matière de cybersécurité. En effet, les vulnérabilités d'une bibliothèque populaire peuvent affecter tous les systèmes qui l'utilisent.

Par conséquent, il est clairement *de la plus haute importance de trouver les failles de sécurité, ou vulnérabilités, dans le code source et d'y remédier* dans la mesure du possible.
L'objectif de ce projet est justement de développer de nouvelles méthodes innovantes pour trouver et corriger les vulnérabilités dans les codes sources des logiciels.

La cybersécurité est un problème croissant dans le développement et la distribution de logiciels. Les développeurs de logiciels non experts (et parfois inexpérimentés) en sécurité ont de grandes difficultés à mettre au point des logiciels sûrs. Les attaques possibles de la chaîne de production logicielle, où un code toxique est injecté dans les dépôts logiciels et se propage à travers le processus de développement et de distribution, exacerbent la menace en matière de sécurité. *Une technique améliorant radicalement les chaînes de production logicielle utilisées est donc nécessaire pour relever le défi de la sécurité, en donnant aux développeurs de meilleurs outils pour développer, valider et déployer des logiciels plus sûrs.* Je souhaite par ce projet relever ce défi en créant une approche de développement semi-automatique qui s'appuie sur des techniques telles que la modélisation de la sécurité, l'analyse de programmes, le test, la certification, le refactoring, la vérification etc. pour sécuriser les chaînes d'approvisionnement en logiciels.

3.3.2 Objectifs

Les objectifs de mon projet sont plus précisément :

Détection : *Développer des méthodes automatisées de détection des failles de sécurité qui permettent de repérer les sections de code vulnérables.*
Nous collecterons et modéliserons les vulnérabilités de sécurité et les classerons en fonction de leur gravité, de leur détectabilité et de leur capacité à être corrigées. Cela nous permettra d'effectuer des analyses de programme avancées pour i) détecter les failles de sécurité dans les codes sources en utilisant des méthodes statiques et/ou dynamiques et/ou l'IA et ii) détecter et qualifier la propagation des effets de la faille et l'impact sur la chaîne d'approvisionnement.
Ce travail répond directement au besoin d'outils permettant de s'assurer que les composants tiers et open-source sont exempts de vulnérabilités, de faiblesses et/ou de logiciels malveillants, en commençant par détecter les vulnérabilités.

1. Certaines de ces vulnérabilités étant dites "zero day", ce qui signifie qu'elles n'ont jamais été identifiées publiquement auparavant.

Réparation : *Développer de nouvelles techniques de refactoring pour permettre aux développeurs de corriger automatiquement les vulnérabilités de leur code source.*
La résolution manuelle des failles de sécurité exige des développeurs qu'ils comprennent la faille de sécurité, son impact sur la propagation et sa correction. Cela nécessite non seulement une grande expertise de la part des développeurs, mais aussi des connaissances hautement spécialisées sur le type de vulnérabilité et une connaissance approfondie du code source. Mon projet vise à offrir des outils (semi-)automatisés de transformation de code pour aider les développeurs non spécialisés à remédier aux vulnérabilités du code source en toute sécurité et avec un minimum d'effort. Des modèles spécifiques guideront ces outils de refactoring dans la correction des vulnérabilités du code source. Les outils de refactoring transformeront alors automatiquement le code source, supprimant ainsi la vulnérabilité. Nous fournirons également au développeur d'autres outils pour lui permettre de propager les effets de la correction d'une vulnérabilité à l'échelle d'un projet global.
Ce travail répond directement au besoin d'outils fournissant l'assurance que les composants tiers et open-source sont exempts de vulnérabilités, de faiblesses et/ou de logiciels malveillants, en corrigeant les vulnérabilités.

Certification : *Développer de nouvelles techniques de certification qui fournissent une validation solide que les vulnérabilités de sécurité ont bien été corrigées.*
Nous mettrons au point des instruments de certification pour garantir l'éradication des vulnérabilités en matière de sécurité du code source. Ces certificats fourniront des garanties solides que les outils de refactoring ont éliminé certaines vulnérabilités précises. Ce certificat sera transporté avec l'artefact logiciel et utilisé tout au long de sa durée de vie pour donner une solide confiance en sa sécurité.
Ceci répond au besoin d'un processus intégré pour le test, la vérification, la validation et la prise en compte des aspects liés à la certification, ainsi que sur la nécessité de procédures d'audit efficaces pour la cybersécurité.

Plateforme : *Intégrer ces modèles, techniques et outils dans une plateforme de développement capable de détecter et de réparer les failles de sécurité et de produire des certificats pour valider la suppression de la vulnérabilité.*
L'identification, la correction et la validation des failles de sécurité sont souvent des tâches distinctes. En outre, certaines techniques nécessitent des considérations particulières à différents niveaux de la chaîne d'approvisionnement, depuis les spécifications de haut niveau du code source, jusqu'au code source, aux bibliothèques et au déploiement, chaque couche nécessitant souvent des environnements de développement, des outils et des techniques différents. Ce projet vise à concevoir une plateforme de développement unifiée qui détecte, répare et valide les vulnérabilités de sécurité à différents niveaux de la chaîne d'approvisionnement, dépassant ainsi l'état de l'art actuel. Nous intégrerons nos outils dans des IDE, tels qu'Eclipse ou Visual Studio Code, et des chaînes CI/CD, répandus.
Ce travail répond au besoin d'outils permettant de détecter les vulnérabilités, les failles et les maliciels dans les composants tiers et open-source. Ces outils seront intégrés dans un processus de test, de vérification, de validation et de prise en compte des aspects de certification afin de fournir des méthodes et des procédures permettant de sécuriser les chaînes d'approvisionnement.

Evaluation : *Déterminer les besoins de l'industrie et évaluer l'efficacité de nos recherches et de nos outils en conséquence.*
Les activités de recherche et de développement de ce projet s'appuieront sur différents cas pratiques industriels. Ces cas d'utilisation fourniront des exigences différentes et complémentaires en termes de sécurité et de types de vulnérabilités nécessitant une détection et une remédiation. Nous évaluerons les méthodes et les outils de notre projet sur ces cas d'utilisation afin de démontrer que les vulnérabilités ont bel et bien été éradiquées.
Cet objectif est directement lié à l'applicabilité concrète et pratique de nos travaux et à leur impact économique dans l'industrie.

3.3.3 Défis

Afin d'atteindre les objectifs décrits ci-dessus, un certain nombre de défis techniques clés doivent être relevés.

Créer de nouveaux modèles complets des vulnérabilités et des correctifs, ainsi que des analyses et des refactorings capables de les exploiter de manière efficace et automatique.
Les vulnérabilités et leurs corrections sont généralement fournies de manière lisible mais informelle, pour une utilisation humaine. Des modèles formels entièrement structurés doivent capturer les éléments d'information pertinents et être utilisables dans l'ensemble de notre plateforme, pour automatiser à la fois la détection des vulnérabilités et les transformations de code nécessaires pour les corriger. Ces modèles doivent également être extraits automatiquement des catalogues existants, quelque peu informels. La conception des modèles formels entièrement structurés nécessaires au projet est donc un défi majeur.

Développer des analyses et des refactorings automatisés efficaces.
Nos analyses et nos corrections doivent être suffisamment rapides pour être pratiques et efficaces. Ainsi, leurs coûts doivent être évalués lors de la conception et contrôlés lors de la mise en œuvre afin de s'assurer avec précision (et rapidité) que les budgets temporels sont respectés. La rapidité de nos analyses et de nos refactorings est un défi technologique majeur dans ce projet.

Développer des certificats qui peuvent être attachés au code analysé et sécurisé.
Des certificats contenant des informations utiles et fiables sur les vulnérabilités présentes ou absentes dans une zone de code, et sur les correctifs appliqués, seront développés dans le cadre de ce projet. En outre, la capacité de transporter efficacement ces certificats avec le code associé est cruciale pour la création de valeur ajoutée grâce à la sécurité supplémentaire apportée par l'utilisation de nos techniques et de nos outils. Fournir cette capacité de manière simple est un défi que le projet devra relever.

Développer une plateforme qui s'intègre harmonieusement aux pratiques existantes.
Pour être acceptés par les développeurs, et donc avoir un impact maximal, nos méthodes et outils doivent être mis en œuvre dans une plateforme cohérente, utilisable et conviviale pour les développeurs, qui s'intègre sans effort dans leurs IDE et leurs chaînes CI/CD.

3.3.4 Approche

Le déroulement logique de nos travaux scientifiques et techniques comprendra les étapes suivantes, correspondant aux objectifs décrits ci-dessus :

Étape 1. Identifier et modéliser les vulnérabilités La première étape de notre vision est de pouvoir à la fois identifier et modéliser les vulnérabilités afin de pouvoir ensuite raisonner concrètement à leur sujet. Nous prévoyons de nous appuyer sur des catalogues structurés et bien établis de vulnérabilités existantes, connues et enregistrées, tels que MITRE CVE (Common Vulnerabilities and Exposures), qui répertorie dans un format lisible par un humain les vulnérabilités connues pour les composants logiciels identifiés par le CPE (Common Platform Enumeration). Sur la base de ces catalogues connus, nous développerons des modèles plus abstraits et formels de vulnérabilités, afin d'extraire automatiquement les vulnérabilités des ensembles de données susmentionnés et de les convertir en une base de données automatiquement exploitable dans notre plateforme, permettant ainsi les étapes suivantes.

Étape 2 : Trouver les vulnérabilités. L'étape 2 consiste à rechercher automatiquement les vulnérabilités dans les dépôts de code et les chaînes de build, en utilisant les modèles et la base de données de vulnérabilités de l'étape 1. Il existe divers outils commerciaux et gratuits capables de trouver des vulnérabilités dans le code source, qui mettent souvent en œuvre des méthodes d'analyse statique, tout en tenant compte des idiomes des langages de programmation. Ces instruments constituent la base du Static Application Security Testing (SAST). Les forces et les faiblesses de ces approches et technologies seront évaluées dans ce projet, et nous sélectionnerons les meilleures d'entre elles et améliorerons leurs performances pour pouvoir les intégrer dans notre plateforme. Nous pourrons également utiliser des tests dynamiques pour obtenir davantage de données sur le comportement des logiciels et, par exemple, repérer plus précisément les endroits où des vulnérabilités sont probables.
Notre plateforme aura la capacité de déterminer les dépendances des composants logiciels, afin de comprendre pleinement l'étendue d'une vulnérabilité. Nous établirons une liste des solutions open-source les

plus courantes et les plus pertinentes pour gérer les dépendances entre composants, et nous proposerons un modèle uniforme pour ces dépendances. Après avoir identifié les vulnérabilités dans le code source et connaissant les dépendances entre les composants logiciels, nous pouvons retracer de manière transitive l'impact de chaque vulnérabilité, en commençant par les composants impactés et en suivant les dépendances. Ce travail "ascendant" permettra de relier les vulnérabilités des composants logiciels au code employant ces composants.
L'analyse d'un grand nombre de lignes de code peut produire des milliers d'avertissements qui doivent être examinés manuellement pour réduire les faux positifs et comprendre les vrais positifs afin de fournir des solutions. Nous voulons automatiser ce processus autant que possible pour le rendre utilisable et évolutif. Il est donc essentiel d'améliorer la vitesse et la précision de la détection. Nos outils fourniront ensuite un rapport aux développeurs dans leurs IDEs habituels à propos des vulnérabilités (ou l'absence de vulnérabilités) dans leur dépôt de code afin qu'ils puissent évaluer avec précision la sécurité de leur code.

Étape 3 : Corriger les vulnérabilités. Après avoir trouvé des vulnérabilités et les avoir signalées aux développeurs, nous nous efforcerons de les supprimer afin de rendre le code plus sûr. De nombreuses vulnérabilités et dépendances du système à des codes vulnérables auront été révélées au cours des étapes précédentes. Cette étape 3 se concentre sur la correction aussi automatique que possible des vulnérabilités identifiées, en remaniant le code en adaptant les corrections de vulnérabilités disponibles aux variants présents dens le logiciel. Si aucun correctif n'est disponible pour une vulnérabilité, le code potentiellement dangereux peut être supprimé, mais la fonctionnalité du système sera altérée. Pour favoriser une plus grande acceptation de nos méthodologies et outils, cette ablation devrait être proposée au développeur lorsqu'il n'y a pas de correctif disponible. Le développeur décidera si la suppression de la fonctionnalité vulnérable vaut la protection accrue ou si elle mutilerait tellement le système qu'il est préférable de laisser la fonctionnalité, et donc la vulnérabilité, en place. Cette décision sera donc explicite et consignée dans le code.
Qu'il s'agisse d'une correction ou d'une ablation, le code est remanié localement, mais les dépendances du code peuvent nécessiter des changements globaux supplémentaires. Lorsqu'une correction implique l'utilisation d'une version plus récente de la bibliothèque avec un code de fonction mis à jour et une interface modifiée (paramètres), tous les appels système du logiciel vers la nouvelle version de la fonction doivent être mis à jour avec la nouvelle interface. Il s'agit de co-évolution de code. En raison de la difficulté de la co-évolution manuelle, certains projets ont du mal à suivre leurs dépendances ou à co-évoluer leur code pour des raisons de sécurité ou autres. Ainsi, tous les clients ne bénéficient pas de la correction des vulnérabilités dans la dernière version d'une bibliothèque. L'adoption et l'impact de la plateforme de notre projet dépendent donc de l'automatisation de ce processus de co-évolution lors de la correction ou de l'ablation d'une vulnérabilité.

Étape 4 : Certifier le travail effectué en associant des certificats au code. La sécurité est souvent négligée lors de la conception et du déploiement des systèmes. Nous pensons que c'est parce que la sécurité n'est pas fonctionnelle et n'est pas évidente (surtout lorsque tout fonctionne parfaitement ; elle est plus visible lorsque la sécurité est violée). La sécurité est donc rarement vendable (sauf dans les secteurs d'activité sensibles à la sécurité, comme le secteur militaire). Par conséquent, les développeurs et les gestionnaires ne sont guère incités à assurer une sécurité élevée.
Nos recherches antérieures ont mis l'accent sur le fait d'élever la sécurité des programmes au rang de citoyen de première classe et de rendre commercialisables des niveaux de sécurité plus élevés. La responsabilité et la certification sont des éléments essentiels pour rendre obligatoire la prise en compte des caractéristiques de sécurité dans les systèmes informatiques. Le pilier juridique de la responsabilité, qui apporte la crainte d'une sanction potentielle aux systèmes non sécurisés, motiverait les producteurs et les vendeurs de logiciel à investir du temps et de l'argent dans la sécurité de leur produits. En effet, la responsabilité rend rentable le temps supplémentaire consacré à la spécification et à la vérification La certification est le fondement technologique sur lequel la responsabilité légale peut être basée. Avec la certification, les efforts de sécurité des fabricants de systèmes peuvent être quantifiés, tarifés et reconnus légalement ; les utilisateurs peuvent exiger la sécurité sur la base d'une évaluation indépendante ; et les régulateurs peuvent interdire les systèmes à faible sécurité. La certification peut imposer le recours à des suites de tests, à des outils de vérification, à de meilleures pratiques en matière de développement, etc. La responsabilité et la certification encouragent les organisations à produire des logiciels sûrs ou à trouver, atténuer et corriger les failles dans les logiciels

existants.

Étape 5 : Avoir une plateforme utilisable. Pour mettre en œuvre les tâches exposées ci-dessus, nous avons besoin d'une plateforme pour combiner tous nos travaux et modules en un ensemble cohérent. Cette plateforme nous permettra d'effectuer des tests et d'évaluer la qualité et l'utilité de notre travail au sein du projet. Pour maximiser l'impact, cette plateforme sera incorporée dans les chaînes de construction populaires, les procédures CI/CD et les IDE des développeurs. Elle pourrait ainsi avoir un impact majeur et être adoptée par la communauté des développeurs et servir de preuve de concept pour des outils plus complexes dont l'objectif est d'être présent dans chaque environnement de développement afin d'aider les développeurs à améliorer la sécurité.

3.3.5 Avancées attendues

Ce projet fera progresser l'état de l'art dans plusieurs directions techniques et scientifiques clés, en apportant les contributions suivantes :

1. Nouvelles méthodes de détection automatique des failles de sécurité dans le code source.
Les failles de sécurité sont souvent découvertes par hasard, par l'examen manuel du code source ou par des tests spécifiques (fuzzing, tests de pénétration). Il existe des outils pour aider les développeurs à découvrir les failles de sécurité, mais ils utilisent soit les tests seuls pour ne couvrir qu'un sous-ensemble des comportements possibles, soit l'analyse syntaxique du code source pour trouver les fautes de programmation les plus courantes. Cependant, l'analyse syntaxique seule ne permet pas d'établir l'absence de vulnérabilité. Ce projet devrait donc intégrer des approches basées sur la sémantique afin de créer une méthodologie plus solide pour les programmes sécurisés.

2. Nouvelles techniques et outils permettant de détecter l'impact et la propagation des failles de sécurité dans le code source.
Les outils de gestion des dépendances étant courant dans les environnements de développement de logiciels pour de nombreux langages, ils pourraient être exploités dans le cadre de ce projet. Pour les langages où ils sont moins courants, par exemple C et C++, nous ferions progresser l'état de l'art dans deux directions. La première consisterait à récupérer automatiquement ces informations sur les dépendances, au moins pour les projets open-source, en se basant sur toutes les informations accessibles dans les grandes archives open-source, telles que Software Heritage. La seconde consisterait à intégrer des gestionnaires de paquets tels que CONAN (`https://conan.io/`) ou vcpkg (`https://vcpkg.io/en/index.html`) afin de pouvoir intégrer une analyse d'impact dès l'identification d'une vulnérabilité dans toutes les dépendances d'un projet.

4. Nouveaux modèles pour les vulnérabilités de sécurité connues et inconnues.
Pour répondre à nos exigences en matière de modèles exploitables pouvant être utilisés dans les outils de notre plateforme, nous étendrons les modèles de vulnérabilité et les méthodologies de modélisation existants afin de les adapter aux catalogues de vulnérabilités ouverts. Nous continuerons à développer ces modèles pour prendre en compte les vulnérabilités inconnues (de types connus), ce qui exige qu'ils soient généraux, de haut niveau, mais suffisamment spécifiques pour inclure toutes les informations essentielles sur les types de vulnérabilités.

5. Nouvelles techniques de transformation de programmes corrigeant les vulnérabilités de sécurité dans le code source.
L'état de l'art actuel en matière de refactoring pour corriger les vulnérabilités implique des transformations de code assez locales. Dans ce projet, nous adopterons une approche différente, en considérant un large éventail de types de vulnérabilités, des CVE connues aux vulnérabilités inconnues. Ce projet ne se concentrera pas non plus sur des microtransformations locales, mais sur le remaniement du code source d'une manière plus globale. Plutôt que d'appliquer des refactorings apparemment ad-hoc, notre approche de refactoring sera plus générique et guidée par des mécanismes de détection de vulnérabilités.

3.3.6 Conclusion

La sécurisation (des artefact issus) de la production des chaines de build est un axe de mes recherches futures qui constitue en quelques sorte l'extension à la fois de mes travaux de cybersécurité (cf. section 2.6) et de mes travaux actuels dans via le co-encadrement de la thèse de George Aaron RANDRIANAINA sur le processus de build (cf. section 2.3.9).
C'est également un axe que je mets actuellement concrètement en marche via ma participation au montage d'un projet SHW-Sec (Software Heritage Security) du CampusCyber (accepté en 2022), et via la coordination du montage d'un projet européen (refusé en 2022, re-soumis en 2023). Ces travaux seront donc effectués de façon collaborative.
La vision de ce projet, qui revient peu ou prou à créer un système "nettoyant" les dépôts logiciels des vulnérabilités, implique clairement une applicabilité très large et un impact technique, mais surtout économique et sociétal, fort, ce qui n'est que plus motivant pour moi.

3.4 Conclusion de mon programme de recherche

J'ai ici présenté deux axes majeurs des travaux que je veux mener dans le futur. Ces deux axes sont complémentaires, le premier (l'intégration explicite des propriétés de sécurité, section 3.2) étant plus amont, le second (sécurisation de la production des chaines de build, section 3.3) plus aval, dans le processus de développement de logiciels. Ils ont également en commun d'être pour moi enthousiasmants, et d'avoir un impact potentiel considérable, donc une utilité pratique, économique, sociétale, ce qui comme je l'ai déjà indiqué, est très important pour moi.

La quantité de travail à faire semble considérable, mais est gérable en avançant par étapes. Comme on dit familièrement : "Y'a plus qu'à !".

Bibliographie (extraits)

[Ach+2023] Mathieu Acher, Luc Lesoil, Georges Aaron Randrianaina, Xhevahire Tërnava et Olivier Zendra. "A Call for Removing Variability". In : *VaMoS 2023 - 17th International Working Conference on Variability Modelling of Software-Intensive Systems*. Odense, Denmark, jan. 2023. doi : 10.1145/3571788.3571801. url : https://hal.science/hal-03882594.

[APZ2011a] Sophie Alexandre, Jonathan Ponroy et Olivier Zendra. *Configuration d'Eclipse pour le portail de télécommande de la plateforme matérielle du projet ANR Open-PEOPLE.* Technical Report RT-0403. INRIA, mars 2011. 18 p. url : https://inria.hal.science/inria-00574956.

[APZ2011b] Sophie Alexandre, Jonathan Ponroy et Olivier Zendra. *Manuel d'administration de l'espace collaboratif wiki du projet ANR Open-PEOPLE.* Technical Report RT-0404. INRIA, mars 2011. 19 p. url : https://inria.hal.science/inria-00574961.

[APZ2011c] Sophie Alexandre, Jonathan Ponroy et Olivier Zendra. *Specification for the Open-PEOPLE software platform (user side).* Technical Report RT-0410. INRIA, juill. 2011. 46 p. url : https://inria.hal.science/inria-00599582.

[Blo+2012] Dominique Blouin, Eric Senn, Kevin Roussel et Olivier Zendra. "QAML : a multi-paradigm DSML for quantitative analysis of embedded system architecture models". In : *The 6th International Workshop on Multi-Paradigm Modeling.* MPM '12 : Proceedings of the 6th International Workshop on Multi-Paradigm Modeling. Innsbruck, Austria : ACM Press, oct. 2012, p. 37-42. doi : 10.1145/2508443.2508450. url : https://hal.science/hal-03219224.

[Bro+2019] Christopher Brown, Adam D Barwell, Yoann Marquer, Céline Minh et Olivier Zendra. "Type-Driven Verification of Non-functional Properties". In : *PPDP 2019 - 21st International Symposium on Principles and Practice of Declarative Programming.* Porto, Portugal : ACM Press, oct. 2019, p. 1-15. doi : 10.1145/3354166.3354171. url : https://inria.hal.science/hal-02314723.

[Bro+2022] Christopher Brown, Adam Barwell, Yoann Marquer, Olivier Zendra, Tania Richmond et Chen Gu. "Semi-automatic ladderisation : improving code security through rewriting and dependent types". In : *PEPM 2022 - ACM SIGPLAN International Workshop on Partial Evaluation and Program Manipulation.* Philadelphia PA, United States : ACM, jan. 2022, p. 14-27. doi : 10.1145/3498886.3502202. url : https://inria.hal.science/hal-03805561.

[BZ2009] Antoine Beugnard et Olivier Zendra. *Conférence francophone sur les Architectures Logicielles : CAL 2009.* T. L-4. RNTI. Cépaduès, 2009. 104 p. url : https://hal.science/hal-02141769.

[Cas+2010] Hugues Cassé, Karine Heydemann, Haluk Ozaktas, Jonathan Ponroy, Christine Rochange et Olivier Zendra. "A framework to experiment optimizations for real-time and embedded software". In : *International Conference on Embedded Real Time Software and Systems (ERTS2).* Toulouse, France, mai 2010. url : https://inria.hal.science/inria-00539973.

[Cas2012] Pierre Caserta. “Analyse statique et dynamique de code et visualisation des logiciels via la métaphore de la ville : contribution à l’aide à la compréhension des programmes”. 2012LORR0266. Thèse de doct. 2012. URL : http://www.theses.fr/2012LORR0266/document.

[CCZ1997] Suzanne Collin, Dominique Colnet et Olivier Zendra. “Type Inference for Late Binding. The SmallEiffel Compiler.” In : *Joint Modular Languages Conference (JMLC)*. T. 1204. Lecture Notes in Computer Sciences. Lintz, Austria : Springer Verlag, 1997, p. 67-81. URL : https://inria.hal.science/inria-00563353.

[CCZ1998a] Dominique Colnet, Philippe Coucaud et Olivier Zendra. “Compiler Support to Customize the Mark & Sweep Algorithm”. In : *ACM SIGPLAN International Symposium on Memory Management - ISMM’98*. ACM Special Interest Group on Programming Languages (SIGPLAN). Vancouver, British Columbia, Canada : ACM Press, 1998, p. 154-165. URL : https://inria.hal.science/inria-00098708.

[CCZ1998b] Dominique Colnet, Philippe Coucaud et Olivier Zendra. “Eiffel : la puissance de la recherche au service de vos programmes”. In : *Programmez !* 2 (1998), p. 82-83. URL : https://inria.hal.science/inria-00098441.

[Chi+2010] Daniel Chillet, Eric Senn, Olivier Zendra, Smail Niar, Cécile Belleudy, V. Tissier et Christian Samoyeau. *Open-PEOPLE ANR Project, Open Power and Energy Optimization Platform and Estimator*. In HiPEAC INFO 24. Oct. 2010. URL : https://hal.science/hal-00923804.

[Chi+2011] Daniel Chillet, Eric Senn, Cécile Belleudy, Rabie Ben Atitallah, Olivier Zendra et Agnes Fritsch. “Open power and energy optimization platform and estimator (open-people)”. In : *International Workshop on Power and Timing Modeling, Optimization and Simulation (PATMOS 2011)*. Madrid, Spain, sept. 2011. URL : https://inria.hal.science/hal-00650648.

[CSZ1999] Dominique Colnet, Emmanuel Stapf et Olivier Zendra. *NICE ELKS 2000 proposal*. Rapport interne. 99-R-359. 1999. 29 p. URL : https://inria.hal.science/inria-00107842.

[CZ1999] Dominique Colnet et Olivier Zendra. “Optimizations of Eiffel programs : SmallEiffel, The GNU Eiffel Compiler.” In : *29th conference on Technology of Object-Oriented Languages & Systems - TOOLS Europe 29’99*. 29th Technology of Object-Oriented Languages and Systems (TOOLS/EUROPE-29 ’99). Nancy, France : IEEE Computer Society, juin 1999, p. 341-350. DOI : 10.1109/TOOLS.1999.779065. URL : https://inria.hal.science/inria-00098754.

[CZ2000] Dominique Colnet et Olivier Zendra. *Targeting the Java Virtual Machine with Genericity, Multiple Inheritance, Assertions and Expanded Types*. Rapport interne. A00-R-137. 2000. 14 p. URL : https://inria.hal.science/inria-00099302.

[CZ2001] Dominique Colnet et Olivier Zendra. “Global system analysis at work”. In : *Journal of Object Oriented Programming* 14.1 (2001), p. 10-13. URL : https://inria.hal.science/inria-00108074.

[CZ2009] Bernard Carré et Olivier Zendra. *Langages et Modèles à Objets : LMO 2009*. Actes de la 15ème édition de la conférence LMO, Nancy, 24-27 mars 2009. Cepadues Editions, Revue RNTI L-3, mars 2009. 154 p. URL : https://hal.science/hal-00713975.

[CZ2011a] Pierre Caserta et Olivier Zendra. “A Tracing Technique using Dynamic Bytecode Instrumentation of Java Applications and Libraries at Basic Block Level”. In : *6th workshop on Implementation, Compilation, Optimization of Object-Oriented Languages, Programs and Systems 2011 (ICOOOLPS 2011)*. Ian Rogers. Lancaster, United Kingdom, juill. 2011. URL : https://inria.hal.science/inria-00613720.

[CZ2011b] Pierre Caserta et Olivier Zendra. “Visualization of the Static aspects of Software : a survey”. In : *IEEE Transactions on Visualization and Computer Graphics* 17.7 (juill. 2011), p. 913-933. DOI : 10.1109/TVCG.2010.110. URL : https://inria.hal.science/inria-00546158.

[CZ2012] Pierre Caserta et Olivier Zendra. “JBInsTrace : A Tracer of Java and JRE Classes at Basic-Block Granularity by Dynamically Instrumenting Bytecode”. In : *Science of Computer Programming* (fév. 2012). URL : https://inria.hal.science/hal-00672976.

[CZ2021] Bart Coppens et Olivier Zendra. “Privacy : whether you’re aware of it or not, it does matter !” In : *HiPEAC Vision 2021*. Jan. 2021, p. 1-6. DOI : 10.5281/zenodo.4719402. URL : https://inria.hal.science/hal-03362809.

[CZ2023] Bart Coppens et Olivier Zendra. “Is privacy possible in a digital world ?” In : *The HiPEAC Vision 2023*. Jan. 2023, p. 145-162. DOI : 10.5281/zenodo.7461921. URL : https://inria.hal.science/hal-04113319.

[CZB2011a] Pierre Caserta, Olivier Zendra et Damien Bodénès. “3D Hierarchical Edge Bundles to Visualize Relations in a Software City Metaphor”. In : *6th IEEE International Workshop on Visualizing Software for Understanding and Analysis (VISSOFT 2011)*. Williamsburg, United States, sept. 2011. URL : https://inria.hal.science/inria-00613725.

[CZB2011b] Pierre Caserta, Olivier Zendra et Damien Bodénès. *Analysis and Advanced Visualization of Software*. The 25th European Conference on Object-Oriented Programming (ECOOP 2011). ECOOP 2011 Best Poster Award. Juill. 2011. URL : https://inria.hal.science/inria-00612601.

[CZC1998] Dominique Colnet, Olivier Zendra et Philippe Coucaud. *Using Type Inference to Customize the Garbage Collector in an Object-Oriented Language. The SmallEiffel Compiler*. Rapport interne. 98-R-196. 1998. 20 p. URL : https://inria.hal.science/inria-00098438.

[CZC1999a] Philippe Coucaud, Olivier Zendra et Dominique Colnet. “Gestion mémoire : manuelle ou automatique ?” In : *Programmez !* 7 (1999), p. 54-57. URL : https://inria.hal.science/inria-00098894.

[CZC1999b] Philippe Coucaud, Olivier Zendra et Dominique Colnet. “La programmation à objets : Application au langage Eiffel. 2ème partie”. In : *Linux Magazine France* 9 (1999), p. 52-55. URL : https://inria.hal.science/inria-00098975.

[CZC1999c] Philippe Coucaud, Olivier Zendra et Dominique Colnet. “La programmation à objets. Application au langage Eiffel”. In : *Linux Magazine France* 8 (1999), p. 63-76. URL : https://inria.hal.science/inria-00098966.

[DLZ2019] François Déchelle, Bruno Lebon et Olivier Zendra. “MASSE : Modular Automated Syntactic Signature Extraction”. In : *RESSI 2019 - Rendez-vous de la Recherche et de l’Enseignement de la Sécurité des Systèmes d’Information*. Erquy, France, mai 2019. URL : https://inria.hal.science/hal-02159947.

[Duc+2007] Roland Ducournau, Etienne Gagnon, Chandra Krintz, Philippe Mulet, Jan Vitek et Olivier Zendra. “International Workshop on Implementation, Compilation, Optimization of Object-Oriented Languages, Programs and Systems - Report on the Workshop ICOOOLPS’2006 at ECOOP’06”. In : *ECOOP 2006 Workshop Reader - ECOOP 2006 Workshops, Nantes, France, July 3-7, 2006*. Sous la dir. de C. Consel et M. Südholt. T. 4379. Lecture Notes in Computer Science (LNCS) 4379. Springer Berlin / Heidelberg, 2007, p. 1-14. DOI : 10.1007/978-3-540-71774-4_1. URL : https://inria.hal.science/inria-00113516.

[Dur+2017] Marc Duranton, Koen de Bosschere, Christian Gamrat, Jonas Maebe, Harm Munk et Olivier Zendra. *The HiPEAC Vision 2017*. HiPEAC, jan. 2017. 138 p. URL : https://inria.hal.science/hal-01491758.

[Dur+2019] Marc Duranton, Koen de Bosschere, Bart Coppens, Christian Gamrat, Madeleine Gray, Harm Munk, Emre Ozer, Tullio Vardanega et Olivier Zendra. *The HiPEAC Vision 2019*. HiPEAC, jan. 2019. 178 p. URL : https://inria.hal.science/hal-02314184.

[Dur+2021] Marc Duranton, Koen de Bosschere, Bart Coppens, Christian Gamrat, Thomas Hoberg, Harm Munk, Catherine Roderick, Tullio Vardanega et Olivier Zendra. *HiPEAC Vision 2021*. HiPEAC., jan. 2021, p. 1-228. URL : https://inria.hal.science/hal-03359519.

[Dur+2023] Marc Duranton, Koen De Bosschere, Bart Coppens, Christian Gamrat, Madeleine Gray, Thomas Hoberg, Harm Munk, Charles Robinson, Tullio Vardanega et Olivier Zendra. *HiPEAC Vision 2023*. HiPEAC, jan. 2023, p. 1-238. URL : https://inria.hal.science/hal-04023794.

[FZ2018] Tim Felgentreff et Olivier Zendra. *Proceedings of the 13th Workshop on Implementation, Compilation, Optimization of Object-Oriented Languages, Programs and Systems*. Sous la dir. de Tim Felgentreff et Olivier Zendra. 13th Workshop on Implementation, Compilation, Optimization of Object-Oriented Languages, Programs and Systems, ICOOOLPS 2018. 2018. URL : https://inria.hal.science/hal-02314731.

[Giv+2018] Thomas Given-Wilson, Axel Legay, Sean Sedwards et Olivier Zendra. "Group Abstraction for Assisted Navigation of Social Activities in Intelligent Environments". In : *Journal of Reliable Intelligent Environments* 4.2 (mai 2018), p. 107-120. DOI : 10.1007/s40860-018-0058-1. URL : https://inria.hal.science/hal-01629137.

[GZD2002] Dayong Gu, Olivier Zendra et Karel Driesen. *The Impact of Branch Prediction on Control Structures for Dynamic Dispatch in Java*. Research Report RR-4547. INRIA, 2002. 12 p. URL : https://inria.hal.science/inria-00072041.

[Idr+2010a] Maha Idrissi Aouad, Lhassane Idoumghar, René Schott et Olivier Zendra. "Reduction of Energy Consumption in Embedded Systems : A Hybrid Evolutionary Algorithm". In : *META'10 - 3rd International Conference on Metaheuristics and Nature Inspired Computing*. T. 95. Djerba, Tunisia, oct. 2010. URL : https://inria.hal.science/inria-00524975.

[Idr+2010b] Maha Idrissi Aouad, Lhassane Idoumghar, René Schott et Olivier Zendra. "Sequential and Distributed Hybrid GA-SA Algorithms for Energy Optimization in Embedded Systems". In : *the IADIS International Conference Applied Computing 2010*. Timisoara, Romania, oct. 2010, p. 167-174. URL : https://inria.hal.science/inria-00524974.

[Idr+2010c] Maha Idrissi Aouad, Lhassane Idoumghar, René Schott et Olivier Zendra. "Sequential and Distributed SA-Type Algorithms for Energy Optimization in Embedded Systems". In : *IEEE-CiSE 2010 (International Conference on Computational Intelligence and Software Engineering)*. T. 1. Artificial Intelligence. Wuhan, China, déc. 2010. URL : https://inria.hal.science/inria-00541375.

[Idr2011] Maha Idrissi Aouad. "Conception d'algorithmes hybrides pour l'optimisation de l'énergie mémoire dans les systèmes embarqués et de fonctions multimodales". 2011NAN10029. Thèse de doct. 2011. URL : http://www.theses.fr/2011NAN10029/document.

[ISZ2010a] Maha Idrissi Aouad, René Schott et Olivier Zendra. "A Tabu Search Heuristic for Scratch-Pad Memory Management". In : *ICSET 2010 - International Conference on Software Engineering and Technology*. Sous la dir. de WASET. T. 64. WASET - World Academy of Science, Engineering and Technology. Rome, Italy : WASET, avr. 2010, p. 386-390. URL : https://inria.hal.science/inria-00491191.

[ISZ2010b] Maha Idrissi Aouad, René Schott et Olivier Zendra. "Genetic Heuristics for Reducing Memory Energy Consumption in Embedded Systems". In : *ICSOFT 2010 - 5th International Conference on Software Engineering and Data Technologies*. Sous la dir. d'ICSOFT. T. 2. Athens, Greece, juill. 2010, p. 394-402. URL : https://inria.hal.science/inria-00516396.

[ISZ2010c] Maha Idrissi Aouad, René Schott et Olivier Zendra. "Hybrid Heuristics for Optimizing Energy Consumption in Embedded Systems". In : *ISCIS 2010 - 25th International Symposium on Computer and Information Sciences*. T. 62. Londres, United Kingdom : Springer, sept. 2010, p. 409-414. URL : https://inria.hal.science/inria-00524320.

[IZ2007] Maha Idrissi Aouad et Olivier Zendra. "A Survey of Scratch-Pad Memory Management Techniques for low-power and -energy". In : *2nd ECOOP Workshop on Implementation, Compilation, Optimization of Object-Oriented Languages, Programs and Systems (ICOOOLPS'2007)*. Sous la dir. d'Olivier Zendra, Eric Jul et Michael Cebulla. ECOOP. Berlin, Germany, juill. 2007, p. 31-38. URL : https://inria.hal.science/inria-00170210.

[IZ2008] Maha Idrissi Aouad et Olivier Zendra. *Outils de caractérisation du comportement mémoire et d'estimation de la consommation énergétique*. Technical Report RT-0350. INRIA, 2008. 53 p. URL : https://inria.hal.science/inria-00256860.

[MCZ2011] Cristian Maxim, Liliana Cucu-Grosjean et Olivier Zendra. "Towards reducing preemptions to save energy". In : *the 5th Junior Researcher Workshop on Real-Time Computing (JRWRTC 2011)*. Nantes, France, sept. 2011. URL : https://inria.hal.science/hal-00646997.

[MZ2002] Pierre-Etienne Moreau et Olivier Zendra. *GC^2 : A Generational Conservative Garbage Collector for the ATerm Library*. Research Report RR-4548. INRIA, 2002, p. 28. URL : https://inria.hal.science/inria-00072040.

[MZ2004] Pierre-Etienne Moreau et Olivier Zendra. "GC2 : a generational conservative garbage collector for the ATerm library". In : *Journal of Logic and Algebraic Programming* 59.1-2 (avr. 2004), p. 5-34. DOI : 10.1016/j.jlap.2003.12.003. URL : https://inria.hal.science/hal-02314741.

[MZ2015] Floréal Morandat et Olivier Zendra. *Proceedings of the 10th Workshop on Implementation, Compilation, Optimization of Object-Oriented Languages, Programs and Systems, ICOOOLPS 2015, Prague, Czech Republic, July 6, 2015*. Sous la dir. de Floréal Morandat et Olivier Zendra. 2015. DOI : 10.1145/2843915. URL : https://inria.hal.science/hal-01546055.

[MZC2003] Frederic Merizen, Olivier Zendra et Dominique Colnet. *Designing efficient and safe weak references in Eiffel with parametric types*. Rapport interne. A03-R-517. 2003. 14 p. URL : https://inria.hal.science/inria-00107752.

[MZC2004] Frederic Merizen, Olivier Zendra et Dominique Colnet. *Designing efficient and safe non-strong references in Eiffel with parametric types*. Rapport interne. A04-R-149. 2004. 30 p. URL : https://inria.hal.science/inria-00107810.

[MZH2022] Yoann Marquer, Olivier Zendra et Annelie Heuser. "The Indiscernibility Methodology : quantifying information leakage from side-channels with no prior knowledge". Working paper or preprint. Sept. 2022. URL : https://inria.hal.science/hal-03793085.

[NGZ2005] Nicolas Navet, Joël Goossens et Olivier Zendra. *Power-Aware Real-Time Scheduling on Identical Multiprocessor Platforms*. Rapport interne. 2005. 8 p. URL : https://inria.hal.science/inria-00000616.

[Nin+2019a] Tristan Ninet, Axel Legay, Romaric Maillard, Louis-Marie Traonouez et Olivier Zendra. "Model Checking the IKEv2 Protocol Using Spin". In : *PST 2019 - 17th International Conference on Privacy, Security and Trust*. Fredericton, Canada, août 2019, p. 1-9. URL : https://inria.hal.science/hal-02062292.

[Nin+2019b] Tristan Ninet, Axel Legay, Romaric Maillard, Louis-Marie Traonouez et Olivier Zendra. "The Deviation Attack : A Novel Denial-of-Service Attack Against IKEv2". In : *TrustCom 2019 - 18th IEEE International Conference on Trust, Security and Privacy in Computing and Communications*. Rotorua, New Zealand : IEEE, août 2019, p. 1-8. URL : https://inria.hal.science/hal-01980276.

[Nin2020] Tristan Ninet. "Formal verification of the Internet Key Exchange (IKEv2) security protocol". 2020REN1S002. Thèse de doct. 2020. URL : http://www.theses.fr/2020REN1S002/document.

[Nou+2021] Lamine Noureddine, Annelie Heuser, Cassius Puodzius et Olivier Zendra. "SE-PAC : A Self-Evolving PAcker Classifier against rapid packers evolution". In : *CODASPY '21 - 11th ACM Conference on Data and Application Security and Privacy*. Virtual Event, United States : ACM, avr. 2021, p. 1-12. DOI : 10.1145/3422337.3447848. URL : https://inria.hal.science/hal-03149211.

[Nou2021] Lamine Noureddine. "Packing detection and classification relying on machine learning to stop malware propagation". 2021REN1S091. Thèse de doct. 2021. URL : http://www.theses.fr/2021REN1S091/document.

[Pon+2011a] Jonathan Ponroy, Kévin Roussel, Olivier Zendra et Dominique Blouin. *Open-PEOPLE : Architecture and Implementation*. Technical Report. Sept. 2011. 31 p. URL : https://inria.hal.science/inria-00623985.

[Pon+2011b] Jonathan Ponroy, Kévin Roussel, Olivier Zendra et Dominique Blouin. *Open-PEOPLE : Ergonomics*. Technical Report. Sept. 2011. 37 p. URL : https://inria.hal.science/inria-00624000.

[Puo+2021] Cassius Puodzius, Olivier Zendra, Annelie Heuser et Lamine Noureddine. "Accurate and Robust Malware Analysis through Similarity of External Calls Dependency Graphs (ECDG)". In : *ARES 2021 - The 16th International Conference on Availability, Reliability and Security*. This paper received the Best Paper Award for IWCC 2021. Virtual, Austria : ACM, août 2021, p. 1-12. DOI : 10.1145/3465481.3470115. URL : https://hal.science/hal-03328395.

[Puo2022] Cassius Puodzius. "Data-driven malware classification assisted by machine learning methods". 2022REN1S090. Thèse de doct. 2022. URL : http://www.theses.fr/2022REN1S090/document.

[PZ2010] Jonathan Ponroy et Olivier Zendra. *Projet ANR MORE : Documentation développeur sur l'estimation de la consommation énergétique*. Technical Report RT-0397. INRIA, nov. 2010. 32 p. URL : https://inria.hal.science/inria-00529677.

[Ran+2022] Georges Aaron Randrianaina, Djamel Eddine Khelladi, Olivier Zendra et Mathieu Acher. "Towards Incremental Build of Software Configurations". In : *ICSE-NIER 2022 - 44th International Conference on Software Engineering – New Ideas and Emerging Results*. Pittsburgh, PA, United States, mai 2022, p. 1-5. DOI : 10.1145/3510455.3512792. URL : https://hal.science/hal-03558479.

[Ran+2023] Georges Aaron Randrianaina, Djamel Eddine Khelladi, Olivier Zendra et Mathieu Acher. "PyroBuildS : Enabling Efficient Exploration of Linux Configuration Space with Incremental Build". Working paper or preprint. Juin 2023. URL : https://hal.science/hal-04130361.

[Rib+2003] Philippe Ribet, Cyril Adrian, Olivier Zendra et Dominique Colnet. *Conformance of agents in the Eiffel language*. Research Report RR-4927. INRIA, 2003. 19 p. URL : https://inria.hal.science/inria-00071652.

[Rib+2004] Philippe Ribet, Cyril Adrian, Olivier Zendra et Dominique Colnet. "Conformance of agents in the Eiffel language". In : *The Journal of Object Technology* 3.4 (2004), p. 125-143. URL : https://inria.hal.science/inria-00100273.

[Rou+2023] Benjamin Rouxel, Christopher Brown, Emad Ebeid, Kerstin Eder, Heiko Falk, Clemens Grelck, Jesper Holst, Shashank Jadhav, Yoann Marquer, Marcos Martinez De Alejandro, Kris Nikov, Ali Sahafi, Ulrik Pagh Schultz Lundquist, Adam Seewald, Vangelis Vassalos, Simon Wegener et Olivier Zendra. "The TeamPlay Project : Analysing and Optimising Time, Energy, and Security for Cyber-Physical Systems". In : *DATE 2023 - Design, Automation and Test in Europe Conference*. Antwerp, Belgium, avr. 2023, p. 1-6. URL : https://inria.hal.science/hal-04108237.

[Rou2016] Kévin Roussel. "Évaluation et amélioration des plates-formes logicielles pour réseaux de capteurs sans-fil, pour optimiser la qualité de service et l'énergie". 2016LORR0051. Thèse de doct. 2016. URL : http://www.theses.fr/2016LORR0051/document.

[RSZ2015a] Kévin Roussel, Ye-Qiong Song et Olivier Zendra. *Lessons Learned through Implementation and Performance Comparison of Two MAC/RDC Protocols on Different WSN OS*. Research Report RR-8777. INRIA Nancy, mars 2015. 25 p. URL : https://inria.hal.science/hal-01202664.

[RSZ2015b] Kévin Roussel, Ye-Qiong Song et Olivier Zendra. "RIOT OS Paves the Way for Implementation of High-Performance MAC Protocols". In : *SENSORNETS 2015*. Sous la dir. de SCITEPRESS. SensorNets 2015. INSTICC and ESEO Angers. Angers, France, fév. 2015, p. 5-14. DOI : 10.5220/0005237600050014. URL : https://hal.science/hal-01141496.

[RSZ2016] Kévin Roussel, Ye-Qiong Song et Olivier Zendra. "Using Cooja for WSN Simulations : Some New Uses and Limits". In : *EWSN 2016 - NextMote workshop*. Sous la dir. de Kay Roemer. EWSN 2016 - NextMote workshop. ACM. Graz, Austria : Junction Publishing, fév. 2016, p. 319-324. URL : https://inria.hal.science/hal-01240986.

[RZ2011] Kévin Roussel et Olivier Zendra. *Open-PEOPLE : Development Platform*. Technical Report. Sept. 2011. 16 p. URL : https://inria.hal.science/inria-00625996.

[SBZ2012] Eric Senn, Dominique Blouin et Olivier Zendra. "A Multi-Paradigm DSML for Quantitative Analysis of Embedded System Architecture Models". In : *ACM/IEEE 15th International Conference on Model Driven Engineering Languages & Systems - MODELS 2012*. Innsbruck, Austria, sept. 2012. URL : https://hal.science/hal-00760298.

[SCZ2002] Benoit Sonntag, Dominique Colnet et Olivier Zendra. "Dynamic inheritance : a powerful mechanism for operating system design". In : *5th ECOOP Workshop on Object-Orientation and Operating Systems - ECOOP-OOOSWS'2002*. Lecture Notes in Computer Science. Malaga, Espagne : Springer Verlag, juin 2002. URL : https://inria.hal.science/inria-00100788.

[Sen+2011a] Eric Senn, Daniel Chillet, Olivier Zendra, Cécile Belleudy, Sébastien Bilavarn, Rabie Ben Atitallah, Agnès Fritsch et Christian Samoyeau. "Open-People : Open-Power and Energy Optimization Platform and Estimator". In : *Forum SAME 2011 - Sophia Antipolis Microelectronics*. Sophia Antipolis, France, oct. 2011. URL : https://hal.science/hal-00664206.

[Sen+2011b] Eric Senn, Jérémie Guillot, Daniel Chillet, Cécile Belleudy, Smail Niar, Olivier Zendra et Christian Samoyeau. "Open Power and Energy Optimization Platform and Estimator (Open-People) ANR Project". In : *DATE 2011 - Design, Automation & Test in Europe*. Grenoble, France, mars 2011. URL : https://inria.hal.science/hal-00650649.

[Sen+2012a] Eric Senn, Daniel Chillet, Olivier Zendra, Cécile Belleudy, Rabie Ben Atitallah, Agnès Fritsch et Christian Samoyeau. "Open-People : an Open Platform for Estimation and Optimizations of energy consumption". In : *Design and Architectures for Signal and Image Processing Conference (DASIP 2012)*. Sous la dir. de Springer. Karlsruhe, Germany, oct. 2012. DOI : 10.1007/978-3-642-17752-1_26. URL : https://inria.hal.science/hal-00741609.

[Sen+2012b] Eric Senn, Daniel Chillet, Olivier Zendra, Cécile Belleudy, Sebastien Bilavarn, Rabie Ben Atitallah, Christian Samoyeau et Agnès Fritsch. "Open-People : Open Power and Energy Optimization PLatform and Estimator". In : *DSD 2012 - 15th Euromicro Conference on Digital System Design*. Sous la dir. d'IEEE. Çeşme, Izmir, Turkey, sept. 2012, p. 668-675. DOI : 10.1109/DSD.2012.98. URL : https://inria.hal.science/hal-00741610.

[ZB2021] Olivier Zendra et Koen de Bosschere. "Taming the IT systems complexity hydra". In : *HiPEAC Vision 2021*. Jan. 2021, p. 1-8. DOI : 10.5281/zenodo.4719574. URL : https://inria.hal.science/hal-03362810.

[ZC1999a] Olivier Zendra et Dominique Colnet. “Adding external iterators to an existing Eiffel class library”. In : *32th conference on Technology of Object-Oriented Languages & Systems - TOOLS Pacific'99*. Technology of Object-Oriented Languages and Systems, 1999. TOOLS 32. Proceedings. Melbourne, Australia : IEEE Computer Society, nov. 1999, p. 188-199. doi : 10.1109/TOOLS.1999.809425. url : https://inria.hal.science/inria-00098784.

[ZC1999b] Olivier Zendra et Dominique Colnet. “Towards safer aliasing with the Eiffel language”. In : *Intercontinental Workshop on Aliasing in Object-Oriented Systems , IWAOOS'99 - ECOOP'99 workshop reader*. T. 1743. Lecture Notes in Computer Science. Lisbonne, Portugal : Springer Berlin / Heidelberg, 1999, p. 153-. url : https://inria.hal.science/inria-00098844.

[ZC2000] Olivier Zendra et Dominique Colnet. “Vers un usage plus sûr de l'aliasing avec Eiffel”. In : *5ème Colloque Langages et Modèles à Objets - LMO'2000*. Sous la dir. de Sahraoui Houari A. Dony Christophe. Mont Saint-Hilaire, Québec, Canada : Hermès Science, jan. 2000, p. 183-194. url : https://inria.hal.science/inria-00099060.

[ZC2001] Olivier Zendra et Dominique Colnet. “Coping with aliasing in the GNU Eiffel Compiler implementation”. In : *Software : Practice and Experience*. Special Issue : Aliasing in object-oriented systems 31.6 (2001), p. 601-613. doi : 10.1002/spe.373. url : https://inria.hal.science/inria-00100929.

[ZC2021] Olivier Zendra et Bart Coppens. “Cybersecurity must come to IT systems now”. In : *HiPEAC Vision 2021*. Jan. 2021, p. 74-79. url : https://inria.hal.science/hal-03362808.

[ZC2023a] Olivier Zendra et Bart Coppens. “From cybercrime to cyberwarfare, nobody can overlook cybersecurity any more”. In : *The HiPEAC Vision 2023*. Sous la dir. d'HiPEAC. Jan. 2023, p. 130-144. doi : 10.5281/zenodo.7461910. url : https://inria.hal.science/hal-04113296.

[ZC2023b] Olivier Zendra et Bart Coppens. “THE RACE FOR CYBERSECURITY”. In : *The HiPEAC Vision 2023*. Jan. 2023, p. 127-129. url : https://inria.hal.science/hal-04113336.

[ZCC1997] Olivier Zendra, Dominique Colnet et Suzanne Collin. “Efficient Dynamic Dispatch without Virtual Function Tables. The SmallEiffel Compiler.” In : *12th Annual ACM SIGPLAN Conference on Object-Oriented Programming Systems, Languages and Applications (OOPSLA'97)*. T. 32. ACM SIGPLAN Notices. ACM SIGPLAN. Atlanta, United States : ACM Press, oct. 1997, p. 125-141. url : https://inria.hal.science/inria-00565627.

[ZCC1998] Olivier Zendra, Dominique Colnet et Philippe Coucaud. “SmallEiffel : l'Eiffel à Très Grande Vitesse”. In : *Programmez !* 3 (1998), p. 64-67. url : https://inria.hal.science/inria-00098461.

[ZCC1999a] Olivier Zendra, Dominique Colnet et Philippe Coucaud. “La programmation à objets - Application au langage Eiffel (3ème partie)”. In : *Linux Magazine France* 10 (1999). url : https://inria.hal.science/inria-00098990.

[ZCC1999b] Olivier Zendra, Dominique Colnet et Philippe Coucaud. “La programmation à objets. Application au langage Eiffel (4ème partie)”. In : *Linux Magazine France* 11 (1999). url : https://inria.hal.science/inria-00108054.

[ZD2002a] Olivier Zendra et Karel Driesen. *Evaluation of Control Structures for Dynamic Dispatch in Java*. Research Report RR-4370. INRIA, 2002. 59 p. url : https://inria.hal.science/inria-00072218.

[ZD2002b] Olivier Zendra et Karel Driesen. “Stress-testing Control Structures for Dynamic Dispatch in Java”. In : *2nd Java Virtual Machine Research and Technology Symposium (JVM'2002)*. Usenix - The Advanced Computing Systems Association. San Francisco, CA, USA, août 2002, p. 105-118. url : https://inria.hal.science/inria-00000111.

[Zen+2008] Olivier Zendra, Eric Jul, Roland Ducournau, Etienne Gagnon, Richard E. Jones, Chandra Krintz, Philippe Mulet et Jan Vitek. "International Workshop on Implementation, Compilation, Optimization of Object-Oriented Languages, Programs and Systems - Report on the Workshop ICOOOLPS'2007 at ECOOP'07". In : *ECOOP 2007 Workshop Reader*. Sous la dir. de Michael Cebulla. T. 4906. LNCS. Springer, 2008. doi : 10.1007/978-3-540-78195-0. url : https://inria.hal.science/inria-00194953.

[Zen1995] Olivier Zendra. *Application d'architectures connexionnistes à l'apprentissage de l'évitement d'obstacles par un robot mobile.* Rapport de DEA. 1995. 72 p. url : https://inria.hal.science/inria-00565602.

[Zen2000] Olivier Zendra. "Translation and global optimization in class-based languages". Defended and granted with Highest Honors in October 2000. Theses. Université Henri Poincaré - Nancy I, oct. 2000. url : https://theses.hal.science/tel-00565633.

[Zen2006] Olivier Zendra. "Memory and compiler optimizations for low-power and -energy." In : *1st ECOOP Workshop on Implementation, Compilation, Optimization of Object-Oriented Languages, Programs and Systems (ICOOOLPS'2006).* Nantes, France, juill. 2006. url : https://inria.hal.science/inria-00104146.

[ZJ2013] Olivier Zendra et Eric Jul. *Proceedings of the 8th Workshop on Implementation, Compilation, Optimization of Object-Oriented Languages, Programs and Systems, ICOOOLPS 2013, Montpellier, France, July 2, 2013.* Sous la dir. d'Olivier Zendra et Eric Jul. 8th Workshop on Implementation, Compilation, Optimization of Object-Oriented Languages, Programs and Systems, ICOOOLPS 2013. 2013. doi : 10.1145/2491404. url : https://inria.hal.science/hal-01546070.

[ZS2012] Olivier Zendra et Markku Sakkinen. "ICOOOLPS 2010 and MASPEGHI 2010." In : *The Journal of Object Technology* 11.3 (2012). doi : 10.5381/jot.2012.11.3.e2. url : https://inria.hal.science/hal-02314734.