Analyse multi-facettes et opérationnelle pour la transformation des systèmes d'information

Multi-Facet Actionable Analytics for Information System Rejuvenation

A. Rock - CIM &
S. Ducasse - N. Anquetil - RMOD/Inria

June 21, 2019

Mots Clefs : qualité logicielle, runtime-analysis, rétro-ingénierie, régle automatique, modélisation, actionable analysis.

Encadrants Inria : Stéphane Ducasse, Directeur de recherche Inria, (spécialiste en maintenance, outillage et évolution logicielle) et Nicolas Anquetil, Maître de conférences HDR, (spécialiste en analyse de code et évolution logicielle).

Encadrants CIM : Nicolas Dias – Architecte applicatif (Expert en informatique et systèmes d'information) et Jérôme Sudich (Référent technique Logiciel Izy Protect Ingénieur EIGIP, 19 ans d'expérience chez CIM sur le logiciel Izy Protect)

1 Contexte industriel

CIM est une SAS au capital social de 200k détenu à 100% par DL Software. CIM est éditeur, intégrateur, hébergeur et infogéreur de solutions pour l'assurance de personnes en santé, prévoyance. Elle offre une expertise Santé et Prévoyance acquise après plus de 30 ans auprès de ses clients. CIM est hébergeur de ses solutions pour 90% de ses clients et plus de 1000 utilisateurs. Toutes les thématiques d'infrastructure et de surveillance des flux sont intégrées à cette offre. CIM est propriétaire de ses infrastructures serveurs, tous les éléments actifs des systèmes et tous les éléments de stockage sont achetés par CIM, gérés et supervisés par les équipes de CIM. Aucun sous-traitant n'intervient dans les opérations quotidiennes d'hébergement, d'exploitation des solutions et des données hébergées.

CIM est certifiée Microsoft GOLD Partner. Elle est l'éditeur des progiciels de la gamme Izy Links et assure l'intégration de l'ensemble des briques de cette gamme ainsi que des briques partenaires nécessaires à la bonne réussite du projet. Cette solution est développée en Power Builder sur base de données DB2. L'équipe de développement vient d'upgrader en Power Builder 2017 (été 2018) et est en cours de passage sur DB2 v11 (avec l'aide d'un DBA IBM – prestation 2018).

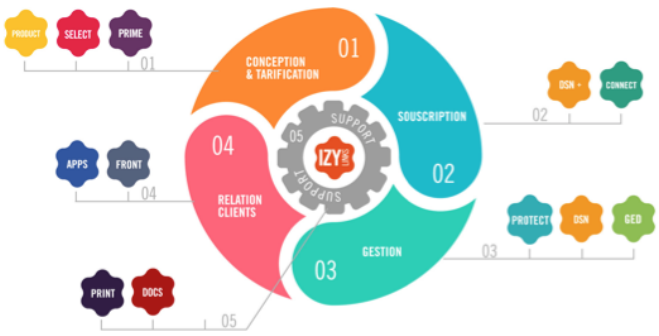

Figure 1: Yzi Protect : le coeur d'un eco système

Le système de gestion est centré sur le *back office* Izy Protect, autour duquel gravite l'ensemble des briques complémentaires répondant à l'ensemble des besoins, et pouvant être activées ou non comme le montre la figure 1.

Voici quelques fonctionalités clefs d'Izy Protect.

- Izy Protect est le *back office* de gestion santé et prévoyance, individuelle et collective. Il permet de suivre la production, les cotisations et les prestations / sinistres. C'est un outil complétement paramétrable qui s'adapte aux spécificités de gestion de chacun de nos clients.
- Izy Protect peut se connecter à l'ensemble des opérateurs de Tiers Payant, la liaison avec Almerys est déjà existante en standard.
- Izy Protect est interfacé avec un logiciel de transfert de fichiers (CFT : *Cross File Transfer*) pour échanger avec l'ensemble des concentrateurs et l'ensemble des caisses. Il permet également de communiquer avec les banques pour transmettre les flux de virement de façon sécurisé par exemple.
- Au sein d'Izy Protect est gérée une comptabilité auxiliaire pour générer l'ensemble des flux nécessaires à destination de la comptabilité générale (Sage 1000). Les flux de déversement comptable sont natifs dans la solutions.
- Un requêteur et un assistant requêteur sont également disponibles en natif. Le requêteur pour interroger directement la donnée depuis l'interface Izy Protect par les utilisateurs ayant les compétences nécessaires à cet exercice. Pour les autres, l'assistant requêteur met à disposition un ensemble de requêtes pré-formatées par les personnes habilitées et mises à dispositions des utilisateurs cibles avec un usage cadré et limité (compatible avec les obligations de la GDPR).

La société CIM a effectué une analyse de risque pour son évolution et croissance en 2017 d'où il ressort que Izy Protect souffre des problèmes de son âge (voir ci-dessous). Suite à cette analyse, une concertation avec des experts Inria en maintenance logicielle a débouché sur cette proposition de contrat CIFRE assortie de l'embauche d'un ingénieur de développement pour aider le doctorant.

2 Contexte et verrous scientifiques

Les systèmes d'information sont les éléments clefs et les exo-squelettes des sociétés et leurs acteurs économiques. Ils sont omniprésents et gèrent les données de nos vies et activités : assurances, salaires, gestion de clients, mutuelles, hopital, RH, grande distribution, commerce... Ils sont la clef de voute des organisations et de leurs *business*.

En 2014, Deloitte[1] a placé la rétro-ingénierie, maintenance et évolution des logiciels tels que les systemèmes d'information dans la liste des 10 thèmes qui permettront d'avoir un impact sur les organisations dans les technologies de l'information.

La situation vécue par CIM autour de son logiciel central s'inscrit dans cette problématique générale.

Problèmes usuels

Les organisations gérant des systèmes d'information font face aux problèmes difficiles suivants [8] :

Vieux langages. La durée de vie d'un système d'information s'étend sur plusieurs décades. Ces systèmes centraux pour les sociétés survivent aux modes technologiques. Alors que leur age offre une grande stabilité, la contre partie est qu'ils sont développés dans des vielles technologies souvent propriétaires. Ces technologies ne sont pas à la pointes des dernières avancées méthologiques et de l'outillage moderne (devops, intégration continue, ...). Elles ne favorisent donc pas, et parfois contrarient, l'adoption de telles avancées et/ou outils.

Logiciel vieillissant. La longue période de croissance et d'évolution de ces systèmes, ainsi que les changements de personnels favorisent l'émergence de problèmes tels que du code mort, du code dupliqué, l'absence de tests, et une documentation obsolète. Très souvent les développeurs originaux ne sont plus présents. Ainsi une grande partie du savoir (et ce à différents niveaux de granularité) est dispersée et diffuse.

Manque d'outils. Les vieux langages ne vivent pas dans des écosystèmes offrant de nombreux outils (*refactorings*, *unit framework*, couverture de tests, analyse de performance, profileur, débogueurs, analyse de qualité, code smells, cartes logicielles, ...). Il est difficile d'extraire des informations exactes permettant des prises de décisions. Le contrôle de l'évolution et la reprise en main du système d'information sont délicats.

Perte de savoir. Comme les systèmes patrinomiaux ont une durée de vie de plusieurs décades, les décisions prises au début du développement et leur évolution au cours de la vie du logiciel sont perdues. Les informations plus fines sont elles aussi perdues par manque de documentation vérifiables, *turn over* des développeurs.

Au cours des années, ces systèmes doivent interagir avec diverse technologies (REST, webservices,...). Ces technologies ont un impact sur l'architecture du

[1] https://www2.deloitte.com/content/dam/Deloitte/global/Documents/Technology/gx-cons-tech-trends-2014-inspiring-disruption.pdf, p.67–76

système. Il est difficile de comprendre les flôts d'information et les processus embarqués dans le logiciel.

Changements à haut risque. Le manque de la connaissance fine du code du système couplé au fait du manque de tests rend chaque changement risqué. Les développeurs ont du mal à faire plus que des fixes ou de traiter les besoins immédiats des clients. Ce risque contribue à la sclérose de tels systèmes.

3 Défis industriels et scientifiques pour CIM

CIM veut améliorer son offre tout en garantissant l'évolution de son logiciel pour les dix prochaines années. Or le logiciel central sur lequel repose la société souffre des problèmes mentionnés ci-dessus. Les deux principaux défis pour CIM.

- Extraction de connaissances afin de permettre la prise de décisions permettant au logiciel de continuer à pouvoir s'adapter aux demandes des clients.
- Support pour la montée en charge afin de permettre à la société de gérer des demandes plus nombreuses et de nouveaux clients.

Ces deux points sont cruciaux pour la société et cette thèse CIFRE est un effort pour une reprise en main et une montée en compétence. CIM projette d'embaucher le thésard une fois la thèse finie.

Verrous Scientifiques

Les problématiques énoncées ci-dessus (Section 2) ne sont pas nouvelles, cependant elles restent d'actualité tant par la particularité des situations que par l'absence d'outils applicables (produits par la communauté scientifique) dédiés pour ces systèmes [21].

Cette thèse et ce document s'inscrivent dans la vision d'*Actionable Analytics* proposée par des chercheurs éminents [3, 18] : *les analyses doivent prendre en compte le contexte local afin d'obtenir des résultats qui permettent aux utilisateurs de prendre des décisions*.

C'est dans ce contexte que nous mentionnons les trois verrous identifiés dans le cadre de la thèse : rétro-ingénierie, qualité logicielle et analyse de charge d'execution.

Retro-ingénierie. La rétro-ingénieurie logicielle consiste à appliquer des analyses spécialisées sur les logiciels pour en extraire des informations plus abstraites [4, 16]. Par exemple ces analyses permettent de redéfinir l'architecture du logiciel, extraire des informations sur l'évolution d'API des bibliothèques, identifier les tests pertinents à une portion du code, ...

Ces analyses sont dépendantes de l'objectif recherché et de la technologie utilisée dans le système analysé. On constate malheureusement que le cas des langages 4GL[2] tels que PowerBuilder a été peu étudié [20]. L'essentiel des travaux ce

[2]Langages dit de quatrième génération, v. par exemple: "Langage de programmation de quatrième génération" dans wikipedia

concentrant sur la technologie reine des systèmes d'information, Cobol, ou sur les langages à objets (principalement Java) et plus récemment sur les langages liés au développement web (ex: JavaScript).

Parce qu'ils offrent de nombreuses instructions très spécialisées (gestion de l'interface utilisateur, gestion de la base de données), des langages comme PowerBuilder ouvrent la porte à des nouvelles possibilités d'analyses, plus spécialisées que ce que l'on peut faire avec des langages généralistes comme Cobol ou Java. Ces possibilités doivent être explorées et mises à profit.

Visualization. La rétro-ingénierie s'appuie aussi sur la production de vues révélant des aspects différents d'un système logiciel [4, 16]. Extraire les vues clefs qui permettent de prendre des décisions reste un défi scientifique. En effet, ces vues dépendent du contexte local (*business*, domaine, processus, contraintes liées à la technologie utilisée) et doivent prendre en compte différentes sources d'information telles que l'information structurelle [1, 10], flots de données entre composants, rapports de bugs [5] ... Les ClassBlueprints sont une visualisation permettant de comprendre rapidement des classes [12, 15].

La visualisation et la construction de vues pertinentes qui répondent à des problèmes précis reste une technologie avancée, difficilement accessible à un utilisateur non spécialiste de la visualisation logiciel tels les développeurs d'application chez CIM. Il y a nécessité à définir un outillage plus accessible qui guide les utilisateurs dans la création d'une bonne visualisation leur permettant de répondre à leur besoin spécifique à un instant t.

Qualité logicielle actionable. La définition d'analyses logicielles qui produisent des rapports permettant aux utilisateurs de prendre de réelles décisions en est encore à ses débuts. Les outils industriels aggrègent souvent les résultats d'analyses très génériques qui ne donnent que peu d'information exploitable au jour le jour dans la gestion d'une équipe de développement. Comme toute les entreprises, CIM a exprimé le besoin de définir des indicateurs fins qui permettent de controler la pertinence de ses processus et suivre la qualité de ses systèmes à court et moyen terme. Une autre part du défi est de permettre aux utilisateurs avancés de définir leurs propres indicateurs de qualités et de les inclurent dans les rapports existants.

Analyses et charge d'exécution. Les traces ou *profilers* traditionnels permettent de comprendre l'exécution de programmes [2, 6, 7]. L'application de ses techniques n'est pas immédiate en présence de triggers, de différents langages (PowerBuilder, SQL, HTML, XML, JavaScript). Le point important de l'instrumentation du système est à prendre en compte. En effet, cette instrumentation n'est pas possible pour toutes les technologies, par exemple des technologies propriétaires et avec un marché relativement petit comparé aux langages généralistes tels que Java.

Pistes de solution

Ducasse et Lanza ont proposé une méthodologie pour la compréhension des logiciels objets [11]. Cette approche doit être adaptée aux langages procéduraux (*i.e* non objets)

et/ou 4GL. Lanza a proposé la visualisation *System Complexity* [17]. Elle prend en compte la hiérarchie d'héritage pour apporter une information centrée sur un lien fort de la programmation à objets. Cette visualisation n'est donc pas non plus adaptée aux les systèmes patrimoniaux non objets et devra être repensée.

Mordal-Manet *et al.* [19] présente un modèle de qualité basée sur l'agrégation d'indicateurs de haut niveau validés dans l'industrie. Une solution d'assistance à la paramétrisation du modèle doit être créée pour autoriser l'ajout de nouveux indicateurs de qualité dans le modèle.

Bertuli a proposé d'utiliser la visualisation *System Complexity* avec de l'information dynamique [2, 13]. Greevy a proposé d'enrichir les analyses avec des *features* [9].

4 Objectifs

L'objectif de la thèse est de proposer des modèles et des mécanismes permettant d'assurer une *"Regénération des systèmes d'information"* ("Rejuvenation of Information Systems").

Les expériences et validation des prototypes se feront dans le contexte de l'application du système d'information écrit en PowerBuilder de la société CIM. Il est important de noter que les travaux de recherche effectués seront reversés sous license MIT au sein de la plateforme Moose (http://moosetechnology.org) [14]. Les résultats pressentis sont largement adaptables et applicables à d'autres systèmes d'informations (non objets).

Afin de permettre au thésard de pouvoir pleinement se focaliser sur les objectifs de la thèse, la société CIM va engager un ingénieur expert pour construire une infrastructure (parser, metamodel) dédiée à PowerBuilder au dessus de la plateforme open-source d'analyse Moose. L'extension PowerBuilder sera accessible aux scientifiques utilisant la plateforme Moose. Le thésard pourra utiliser et étendre cette infrastructure pour la géneration de nouveaux outils.

Tâches préssenties

Le doctorant travaillera incrémentalement par cycles de 3 à 4 mois sur les tâches :

- Apprentissage de Powerbuilder, intégration dans l'equipe de développement de CIM, prise de contact avec l'application cible, IziProtect ;
- Apprentissage de la plateforme Moose, compréhension de la méta-modélisation, revue de la littérature ;
- Définition d'indicateurs de progrès fins adaptés au contexte de l'entreprise.
- Identification d'information contextuelle (discussion avec les équipes de développement CIM, compréhension des contraintes, processus métier ou de développement des équipes). Identification d'anti-patterns ;
- Définition des expériences à mener, identification des contraintes ;

- Définition de métriques et de requêtes *actionable*. Par exemple, des cartes logicielles identifiant le code mort, dupliqué, des métriques spécifiques pourront être développées ;
- Réalisation des expériences ;
- Réalisation des différentes expériences ;
- Evaluation des résultats, validation ;
- Identification de points de goulot d'étranglement, dépendances entre flots de données ;
- Définition d'outils pour le profilage et flots de données.

Verrous techniques

Les principaux verrous techniques à résoudre sont les suivants :

- Importeurs (parseur, meta-modèle) pour les technologies utilisées chez CIM (PowerBuilder, DB2)
- Instrumentation du code pour les analyses de performances.

Ces verrous techniques seront appréhendés par l'ingénieur embauché pour épauler le thésard. Cet ingénieur, embauché sur un contrat de 6 mois par CIM dans le cadre du projet, aura pour tâche exclusive de développer l'infrastructure de base qui permettra de faire des analyses plus poussées. Le résultat de son travail sera *open-source* pour être rendu disponible à la communauté Moose.

5 Organisation des travaux de recherche

5.1 Approche Méthodologique

L'appropriation de la pile technologique (meta-modeling avec Moose, PowerBuilder) et l'état de l'art restent des étapes importantes. La connaissance fine du contexte industriel autant au niveau de la sémantique du langage que celui des us et coutumes des équipes de développement sont des points importants à aborder afin d'être pertinent et de produire des résultats actionables de la part de la société.

- La thèse débutera par un état de l'art dans le domaine du reverse engineering, analyse de l'exécution, modèles de qualité. Cependant la thèse suivra une approche agile avec des cycles expérimentaux courts. Ainsi l'état de l'art se fera tout au long de la thèse et sera aiguillonné par les résultats des expériences.
- La définition d'indicateurs de progression est une étape importante dans la thèse. Une fois, la prise en main des plateformes et problèmes identifiés, un des premiers travaux sera l'identification et la définition d'indicateurs de différents niveaux permettant de valider la progression industriel et académique du projet.

- La thèse débutera par la définition de deux analyses et expérimentation autour de la définition du modèle de qualité et visualizations associées et de l'analyse d'exécution.
- Les résultats de ces deux expériences seront repris et mis en perspective pour alimenter les expériences suivantes.

5.2 Environnement de travail

Au niveau du déroulement de la thèse, le doctorant sera encadré par ses directeurs de thèse académique, S. Ducasse et N. Anquetil et ses encadrants industriels, N. Dias et J. Sudich. La société CIM se situe à moins de 200 mètres du laboratoire Inria Lille Nord Europe. Cette proximité va favoriser les échanges et l'encadrement du doctorant.

Il est à noter que l'équipe RMOD a une grande expérience d'interaction et travail avec les entreprises : plusieurs thèses financées par des entreprises ont eu lieu dans le passé ; des membres de l'équipe ont crées une société éditrice de logiciels (http://synectique.eu) ; Finalement l'équipe a créé un consortium industriel autour du logiciel Pharo (http://consortium.pharo.org).

S. Ducasse directeur de recherche Inria première classe, dirige l'équipe RMOD. De 2011 à 2014 il a été le délégué scientifique du centre Inria Lille Nord Europe (300 chercheurs/17 équipes). Il est expert en conception objet, conception de langages à objets, programmation réflexive ainsi que maintenance et évolution de large applications (visualisation, métriques, meta modelisation). Ses travaux sur les traits ont été introduits dans AmbientTalk, Slate, Pharo, Perl-6, PHP 5.4 et Squeak. Ils sont été portés sur JavaScript. Ils ont influencés les langages Scala et Fortress. Il est un des développeurs de Pharo (http://www.pharo.project.org/) un nouveau langage objet pure open-source inspiré de Smalltalk. Il est un des développeurs de Moose une plate-forme d'analyses (http://www.moosetechnology.org/). Il a été un des fondateurs de Synectique, une société proposant des outils d'analyses dédiées. Il a encadré 29 thèses soutenues et de trés nombreux masters. Trois de ses doctorants ont gagné des prix pour leur thèse. Plusieurs de ses doctorants sont ingénieurs chez Google.

Il a écrit une dizaine de livres. D'après Google Scholar son h-index est 53 avec 12000 citations.

Nicolas Anquetil est Habilité à Diriger des Recherches en informatique à l'Université de Lille depuis mai 2014 et effectue son activité de recherche au sein de l'équipe RMod. Il a un doctorat en informatique de l'Université de Montréal (Canada, en 1996), a travaillé successivement à l'Université d'Ottawa (Canada, 1997-1999), l'Université Fédérale de Rio de Janeiro (Brésil, 2000), l'Université Catholique de Brasilia (Brésil, 2001-2007), l'école des Mines de Nantes (France, 2008-2009), l'Université de Lille (France, 2009-actuel). Ses travaux de recherche se concentrent sur l'évolution logicielle : gestion de l'évolution, gestion des connaissances liées au génie logiciel, qualité logiciel, remodularisation d'architecture. Il a co-orienté 6 thèses de doctorat dans ce domaine et a de nombreuses publications dans les principales conférences et revues internationales du domaine.

Chez CIM, le doctorant integrera l'équipe et sera supervisé par

Nicolas Dias Architecte applicatif. BTS SN Lycée Les Eucalyptus – Nice. VAE EPSI, Expert en informatique et systèmes d'information. Nicolas est avant tout un chercheur de solutions adaptées au contexte et aux besoins. Très engagé dans la veille technologique, il est responsable de la trajectoire que CIM engage pour son SI à 2021.

Jérôme Sudich Référent technique Logiciel Izy Protect. Ingénieur EIGIP, 19 ans d'expérience chez CIM sur le logiciel Izy Protect. Concepteur développeur expert Power Builder, et DBA DB2. Jérôme accompagne les nouveaux collaborateurs dans leur montée en compétences et intervient principalement sur des sujets complexes pour lesquels sa grande expertise est requise. Sa connaissance du logiciel est un atout, ainsi que sa grande capacité d'analyse

Au niveau académique, le doctorant sera accueilli dans l'équipe de recherche RMOD dont les thématiques de recherche sont centrées sur l'évolution et la maintenance d'applications et la conception de nouveaux langages pour la définition de systèmes éternels. Le domaine de la thèse est 100% le thème central de l'équipe.

Une réunion bi-hebdomadaire/mensuelle sera organisée par le doctorant pour faire état de l'avancement de ses travaux à ses encadrants industriels et académique. Il préparera à cet effet :

- l'ordre du jour (avec revue d'articles de bibliographie, présentation de résultats, etc.)
- le compte-rendu (avec les points d'actions et directions décidées avec ses encadrants).

Des points plus rapprochés pourront par ailleurs être prévus lorsque le besoin s'en fait ressentir, par exemple lors de la finalisation de travaux ou lors de la rédaction d'articles. Le doctorant sera également amené à présenter ses travaux :

- au sein de séminaire de recherche du laboratoire et du groupe logiciel ;
- au sein de la communauté scientifique, via des conférences ou au sein de l'école doctorale.

Planning du déroulement de la thèse

Ce planning reprend les tâches pressenties identifiées à la section 4 et organisées dans des cycles de 4 mois :

Cycle 1:

- Apprentissage de Powerbuilder, intégration dans l'equipe de développement de CIM, prise de contact avec l'application cible, IziProtect ;
- Apprentissage de la plateforme Moose, compréhension de la méta-modélisation, revue de la littérature ;

- Définition d'indicateurs de progrès fins adaptés au contexte de l'entreprise.

Cycle 2:

- Identification d'information contextuelle (discussion avec les équipes de développement CIM, compréhension des contraintes, processus métier ou de développement des équipes). Identification d'anti-patterns ;
- Définition des expériences à mener. Identification des contraintes .

Cycle 3 et 4:

- Réalisation des expériences ;
- Evaluation des résultats, validation.
- Rédaction d'un premier article scientifique pour une conférence internationale de bon niveau (ex: ICSME–*International Conference on Software Maintenance and Evolution*).

Cycle 5 et 6:

- Définition de métriques et de requêtes *actionable*. Par exemple, des cartes logicielles identifiant le code mort, dupliqué, des métriques spécifiques pourront être développées ;
- Réalisation des expériences ;
- Evaluation des résultats, validation.
- Rédaction d'un article scientifique pour une revue internationale de bon niveau (ex: *IEEE Transaction on Software Engineering*).

Cycle 7 et 8:

- Identification de points de goulot d'étranglement, dépendances entre flots de données ;
- Définition d'outils pour le profilage et flots de données
- Rédaction d'articles scientifiques.

Cycle 9:

- Rédaction du manuscrit de thèse et préparation de la soutenanceRédaction et défense de la thèse

References

[1] H. Abdeen, S. Ducasse, D. Pollet, I. Alloui, and J.-R. Falleri. The package blueprint: Visually analyzing and quantifying packages dependencies. *Science of Computer Programming*, Feb. 2014.

[2] R. Bertuli, S. Ducasse, and M. Lanza. Run-time information visualization for understanding object-oriented systems. In *Proceedings of 4th International Workshop on Object-Oriented Reengineering (WOOR'03)*, pages 10–19. University of Antwerp, 2003.

[3] L. Briand, D. Bianculli, S. Nejati, F. Pastore, and M. Sabetzadeh. The case for context-driven software engineering research. *IEEE Software*, Sept. 2017.

[4] E. Chikofsky and J. Cross II. Reverse engineering and design recovery: A taxonomy. *IEEE Software*, 7(1):13–17, Jan. 1990.

[5] C. Couto, C. Silva, M. T. Valente, R. Bigonha, and N. Anquetil. Uncovering causal relationships between bugs and software metrics. In *Proceedings of the 16th European Conference on Software Maintenance and Reengineering (CSMR'12)*, 2012.

[6] W. De Pauw, E. Jensen, N. Mitchell, G. Sevitsky, J. M. Vlissides, and J. Yang. Visualizing the execution of java programs. In *Revised Lectures on Software Visualization, International Seminar*, pages 151–162, London, UK, 2002. Springer-Verlag.

[7] W. De Pauw, S. Krasikov, and J. Morar. Execution patterns for visualizing web services. In *Proceedings ACM International Conference on Software Visualization (SoftVis'06)*, New York NY, Sept. 2006. ACM Press.

[8] S. Demeyer, S. Ducasse, and O. Nierstrasz. *Object-Oriented Reengineering Patterns*. Morgan Kaufmann, 2002.

[9] M. Denker, O. Greevy, and M. Lanza. Higher abstractions for dynamic analysis. In *2nd International Workshop on Program Comprehension through Dynamic Analysis (PCODA 2006)*, pages 32–38, 2006.

[10] S. Ducasse, T. Gîrba, and A. Kuhn. Distribution map. In *Proceedings of 22nd IEEE International Conference on Software Maintenance*, ICSM'06, pages 203–212, Los Alamitos CA, 2006. IEEE Computer Society.

[11] S. Ducasse and M. Lanza. Towards a methodology for the understanding of object-oriented systems. *Technique et science informatiques*, 20(4):539–566, 2001.

[12] S. Ducasse and M. Lanza. The Class Blueprint: Visually supporting the understanding of classes. *Transactions on Software Engineering (TSE)*, 31(1):75–90, Jan. 2005.

[13] S. Ducasse, M. Lanza, and R. Bertuli. High-level polymetric views of condensed run-time information. In *Proceedings of 8th European Conference on Software Maintenance and Reengineering (CSMR'04)*, pages 309–318, Los Alamitos CA, 2004. IEEE Computer Society Press.

[14] S. Ducasse, M. Lanza, and S. Tichelaar. The moose reengineering environment. *Smalltalk Chronicles*, Aug. 2001.

[15] M. Lanza and S. Ducasse. A Categorization of Classes based on the Visualization of their Internal Structure: the Class Blueprint. In *Proceedings of 16th International Conference on Object-Oriented Programming Systems, Languages and Applications (OOPSLA '01)*, pages 300–311. ACM Press, 2001.

[16] M. Lanza and S. Ducasse. Understanding software evolution using a combination of software visualization and software metrics. In *Proceedings of Langages et Modèles à Objets (LMO'02)*, pages 135–149, Paris, 2002. Lavoisier.

[17] M. Lanza and S. Ducasse. Polymetric views—a lightweight visual approach to reverse engineering. *Transactions on Software Engineering (TSE)*, 29(9):782–795, Sept. 2003.

[18] T. Menzies and T. Zimmermann. Software analytics: So what? *IEEE Software*, July 2013.

[19] K. Mordal-Manet, F. Balmas, S. Denier, S. Ducasse, H. Wertz, J. Laval, F. Bellingard, and P. Vaillergues. The squale model – a practice-based industrial quality model. In *Proceedings of the 25th IEEE International Conference on Software Maintenance (ICSM'09)*, pages 94–103, Edmonton, Canada, 2009.

[20] C. Nagy, L. Vidacs, R. Ferenc, T. Gyimothy, F. Kocsis, and I. Kovacs. Solutions for reverse engineering 4gl applications, recovering the design of a logistical wholesale system. In *2011 15th European Conference on Software Maintenance and Reengineering*, pages 343–346, 2011.

[21] O. Nierstrasz and F. Achermann. Separating concerns with first-class namespaces. In R. E. Filman, T. Elrad, S. Clarke, and M. Aksit, editors, *Aspect-Oriented Software Development*, pages 243–259. Addison-Wesley, 2005.