Rapport de stage de fin d'études

Analyse et gestion de la co-évolution des systèmes et de leur sécurité

Pascal Douville de Franssu, étudiant

Salvador Martinez, tuteur entreprise

Antoine Beugnard, tuteur école

Dates du stage : 1 mai 2024 - 27 septembre 2024
Version : 1

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Lab-STICC
P4S

1. Remerciements

Je tiens tout d'abord à remercier mon tuteur en entreprise, M. Salvador Martinez, pour m'avoir offert cette opportunité de stage et pour m'avoir accompagné tout au long de celui-ci. Son attention et sa bienveillance m'ont permis de découvrir le domaine de la recherche en génie logiciel.

Je souhaite également remercier mon tuteur académique, M. Antoine Beugnard, pour m'avoir orienté et conseillé tout au long de ma formation ainsi que pour la suite de mon parcours.

Enfin, je remercie chaleureusement l'ensemble du département informatique pour m'avoir si bien accueilli. Un remerciement spécial à Yacine Mokhtari, qui m'a accompagné et grandement aidé tout au long de ce stage.

2. Résumé

Suite à mon parcours de formation d'ingénieur au sein d'IMT Atlantique, ce stage représente pour moi une découverte du milieu de la recherche et constitue une potentielle nouvelle orientation professionnelle.

Durant ce stage, un travail de formation basé sur la lecture d'articles m'a conduit à la production d'un état de l'art, notamment sur les pratiques DevSecOps, la modélisation de la sécurité via les Security Data Flow Diagrams (secDFD), ainsi que sur la cohérence entre modèle et système et l'automatisation pour son maintien. En parallèle, j'ai conçu une architecture de solution répondant aux nouvelles orientations de mon stage. Pour cela, divers outils ont été sélectionnés et testés afin de débuter l'implémentation d'une solution automatisée. Celle-ci a pour objectif d'établir une connexion entre un modèle de sécurité et le système qu'il représente, tout en maintenant la cohérence entre ces deux éléments.

3. Contenu

Ce rapport est organisé de la manière suivante :

La première partie présente l'IMT Atlantique et l'équipe de recherche P4S, au sein de laquelle j'ai effectué mon stage.

La deuxième partie est consacrée à la présentation de ma mission. Elle commence par une phase de mise en contexte et d'état de l'art, suivie par une phase de conception d'une solution, et se termine par mes avancées sur l'implémentation de cette solution.

Enfin, la dernière partie propose une analyse rétrospective du stage, en commençant par la mise en pratique des compétences acquises durant ma formation, suivie d'une analyse des contextes sociaux et environnementaux, pour conclure avec mes perspectives d'évolution futures.

Sommaire

Liste des figures

4. Présentation de l'organisme d'accueil

4.1. IMT Atlantique

IMT Atlantique est une grande école d'ingénieurs française fondée en 2017, issue de la fusion entre Télécom Bretagne et l'École des Mines de Nantes. Elle s'inscrit dans le groupe Institut Mines Télécom (IMT) et se distingue par une approche multidisciplinaire des sciences et technologies de l'information, de l'énergie et de l'environnement.

L'école abrite plusieurs laboratoires de recherche [1], notamment en raison de ses partenariats avec le CNRS, l'Inserm et l'INRIA. Elle accueille en son sein de nombreux projets de recherches nationaux et internationaux qui sont souvent en collaboration avec l'industrie. L'accent est porté sur l'interdisciplinarité pour créer des synergies entre les différents domaines de recherche dans l'objectif d'apporter des solutions novatrices et adaptées aux besoins complexes des entreprises et des sociétés modernes.

Parmi ces laboratoires, le Lab-STICC (Laboratoire des Sciences et Techniques de l'Information, de la Communication et de la Connaissance), une unité mixte de recherche (UMR 6285) affiliée au CNRS compte de nombreux pôles de recherches dont le pôle Software and Hardware, ARchitectures and Processes (SHARP).

4.2. L'équipe P4S au sein de Lab-STICC

Pour ma part, j'effectue mon stage dans l'équipe Processes for Safe and Secure Software and Systems (P4S), une des équipes du pôle de recherche SHARP du Laboratoire de recherche Lab-STICC (UMR 6285). Le Lab-STICC est un laboratoire multi-disciplinaire regroupant des chercheurs en électronique, informatique et télécommunications, avec un accent sur les applications civiles et militaires.

Les financements des laboratoires au sein de l'école et P4S se font par des projets de recherche nationaux et européens, des collaborations industrielles, ainsi que par des subventions des agences de recherche publiques (ANR, SPIN, . . .) . Ces financements soutiennent le développement d'outils et de nouvelles méthodologies pour améliorer la sécurité et la sûreté des systèmes logiciels et matériels.

Les recherches de l'équipe P4S s'articulent autour de plusieurs axes [10] :

- Modélisation des systèmes, processus et logiciels : La modélisation des systèmes, processus et logiciels est une discipline clé dans le domaine de l'ingénierie logicielle et des systèmes complexes. Elle consiste en le développement de modèles formels, c'est-à-dire des représentations mathématiques précises, qui permettent de capturer les comportements, les interactions et les structures internes des systèmes complexes. Ces modèles servent de fondement pour l'analyse et la compréhension des systèmes, facilitant ainsi la conception, la simulation, et l'optimisation de ceux-ci. La modélisation formelle joue un rôle crucial dans l'amélioration de la qualité et de la robustesse des systèmes, en permettant une analyse rigoureuse avant la mise en œuvre.

- Vérifications et analyses statiques, dynamiques et formelles : Les méthodes de vérification et d'analyse sont essentielles pour garantir la validité et la fiabilité des systèmes complexes. Les analyses statiques consistent en l'examen du code ou des modèles sans exécution de ce même code, permettant de détecter des erreurs potentielles, des failles de sécurité, ou des violations de contraintes logiques. Les analyses dynamiques, quant à elles, impliquent l'exécution contrôlée

du système pour observer son comportement réel dans différentes conditions. Les vérifications formelles, utilisant des techniques mathématiques rigoureuses comme la logique formelle et la théorie des automates, permettent de prouver de manière exhaustive la conformité du système à ses spécifications. L'application conjointe de ces techniques est cruciale pour détecter les erreurs et les corriger avant le déploiement, minimisant ainsi les risques d'échec en production.

- Sécurité et sûreté des mécanismes : Le développement de mécanismes robustes pour la sécurité et la sûreté des systèmes est une priorité dans la conception des systèmes critiques. Cela implique la mise en œuvre de principes de sécurité dès les premières étapes de la conception, souvent désignés sous le terme de "Security by Design". Les mécanismes de sécurité doivent non seulement protéger le système contre les menaces externes (comme les cyberattaques), mais aussi assurer sa résilience face aux pannes et aux comportements inattendus. Les approches formelles pour la sécurité permettent de modéliser et de vérifier les propriétés de sécurité, garantissant ainsi que les systèmes respectent les standards de sécurité tout au long de leur cycle de vie.

- Méthodes et processus pour le DevSecOps : Le DevSecOps est une méthodologie qui intègre les pratiques de sécurité dans l'ensemble du cycle de vie du développement logiciel, depuis la conception initiale jusqu'à l'exploitation. Cette approche met l'accent sur la collaboration entre les équipes de développement, de sécurité et d'exploitation, assurant que les aspects de sécurité sont considérés comme partie intégrante du processus de développement, plutôt que comme une réflexion après coup. Les méthodes DevSecOps incluent l'automatisation des tests de sécurité, l'analyse continue des vulnérabilités, et l'intégration des retours de sécurité dans les itérations de développement, ce qui permet de réagir rapidement aux nouvelles menaces et de maintenir un haut niveau de sécurité opérationnelle.

- Hétérogénéité et systèmes de systèmes : Il s'agit de l'étude des interactions entre différents systèmes dans des environnements complexes et interconnectés cela permet de comprendre et gérer la complexité croissante des systèmes modernes. Les systèmes de systèmes (SoS) sont des ensembles de systèmes indépendants qui interagissent pour accomplir des objectifs communs. L'hétérogénéité des composants, tant au niveau des technologies employées que des protocoles de communication et des modèles de données, pose des défis importants en matière de compatibilité, de coordination et de gestion de la complexité. La modélisation et l'analyse des SoS permettent de comprendre les dynamiques d'interaction, d'identifier les goulots d'étranglement et les points de défaillance potentiels, et de développer des solutions pour améliorer l'intégration et la performance globale du système.

On peut donc résumer les objectifs de recherche en un développement de méthodes et d'outils pour spécifier et décrire les systèmes et logiciels pour permettre une évaluation et une analyse pour assurer la sûreté et la sécurité. Cela s'inscrit dans une application à la cybersécurité, les jumeau numériques et plus généralement l'industrie du futur, le spatial et les transports. L'objectif est de garantir la sûreté et la sécurité des systèmes dans ces secteurs critiques.

5. Présentation des missions et des travaux effectués

5.1. Introduction du sujet

Le sujet de stage est "Maintien d'un niveau de sécurité au cours de l'évolution d'un système", il s'inscrit dans les thématiques de recherche de l'équipe P4S, en particulier dans le DevSecOps. Le défi consiste à maintenir un niveau de sécurité tout au long de l'évolution d'un système, en s'assurant que chaque modification ou mise à jour n'introduit pas de vulnérabilités. Pour cela, la modélisation joue un rôle clé, elle permet une simplification et une standardisation d'un système. Celui-ci est alors plus compréhensible et il est possible de vérifier des règles de sécurité standardisées sur les différents modèles qui le représentent. Mais après avoir conçu les différents modèles pour un système, encore faut-il maintenir une cohérence entre les modèles et le système au cours de son évolution. Or cette mise à jour au cours du développement du système est une tâche souvent évitée faute de temps. L'absence de cohérence engendrée entraîne alors dans de nombreux cas des confusions et une perte de temps lors de la résolution de problèmes opérationnels. C'est pourquoi le développement de solutions automatisées ou semi-automatisées pourraient amener de la cohérence et faciliter le maintien de la cohérence entre un système et ses modélisations.

C'est dans ce contexte que M. Salvador Martinez a obtenu les financements pour un sujet de thèse : "Model-Based Security Consistency Management for Complex Systems" dans laquelle les objectifs sont d'étudier des systèmes complexes pour évaluer et décrire leur niveau de sécurité et de proposer des mécanismes de détection d'incohérences et de réparation semi-automatiques. C'est alors comme exploration d'une première piste de réponse à ce sujet de thèse que vient s'inscrire le motif du stage. Il s'agit donc d'abord d'évaluer le niveau de sécurité d'un système au travers d'un modèle puis d'instaurer des règles de cohérence pour permettre une co-évolution entre le système et le modèle.

5.2. État de l'art

Principalement au début du stage, mais aussi tout au long de celui-ci, il était essentiel de se familiariser avec les concepts de base, notamment en matière de sécurité des logiciels, de modélisation des systèmes, et des pratiques DevSecOps. Mais pour permettre une appréciation et une assimilation rapide des notions clés, il fallait un cadre plus précis et des bases sur lesquelles s'appuyer. C'est pourquoi dès le début du stage, des orientations ont été données :

- Les systèmes étudiés seront des projets développés en java.
- La modélisation des projets pour l'évaluation de du niveau de sécurité du projet sera les Security DataFlow Diagram (secDFD).

En effet, une partie conséquente des contributions de P4S se font autour de systèmes développés en java, de plus la très large utilisation de ce langage dans l'industrie constitue un argument de poids en faveur du développement d'outils d'aide au processus de développement dans ce langage. De même pour les secDFD, l'utilisation très répandue des DFD (DataFlow Diagram) dans l'analyse sécurité de systèmes dans le milieu de l'entreprise permet de justifier le choix d'étudier ce modèle.

5.2.1. La sécurité dans les cycles de développement

L'ingénierie dirigée par les modèles (Model-Driven Software Engineering, MDSE) est une approche qui vise à améliorer la productivité et la qualité des logiciels en utilisant des modèles comme principaux artefacts tout au long du cycle de vie du développement. Selon M.Brambilla et al. [2], cette approche permet de réduire la complexité en séparant les préoccupations, en automatisant certaines tâches et en facilitant la réutilisation des composants logiciels. Cependant, pour garantir que les systèmes restent sûrs tout au long de leur développement, il est nécessaire d'intégrer l'analyse de la sécurité dès les premières phases de conception.

L'introduction de pratiques de sécurité dès le début du cycle de développement présente plusieurs

avantages. Comme le décrivent P.Meland et J.Jensen [12], l'intégration précoce de l'analyse de sécurité et des pratiques de développement sécurisées peut entraîner des économies de coûts significatives, ainsi qu'une réduction du nombre de défauts de sécurité graves jusqu'à 50 %. Cette approche proactive permet de détecter et de corriger les vulnérabilités avant qu'elles ne deviennent critiques, évitant ainsi des coûts de correction élevés en phase de post-déploiement.

Toutefois, cette intégration de la sécurité n'est pas sans défis. Il existe une pénurie de spécialistes maîtrisant à la fois l'ingénierie logicielle et la sécurité [12]. La complexité croissante des systèmes informatiques et l'augmentation des incidents de sécurité exigent une collaboration plus étroite entre ces deux domaines pour garantir la sécurité des systèmes dès leur conception. Ainsi, l'évaluation continue du niveau de sécurité au cours du développement d'un système est non seulement nécessaire, mais essentielle pour prévenir les failles de sécurité et assurer la robustesse du produit final.

Face aux défis posés par la complexité croissante des systèmes informatiques et la rareté des experts maîtrisant à la fois l'ingénierie logicielle et la sécurité, l'approche DevSecOps apparaît comme une réponse stratégique. DevSecOps intègre la sécurité à chaque étape du cycle de développement, en automatisant les contrôles de sécurité dès la conception. Cette approche vise à combler les lacunes du modèle DevOps traditionnel, en garantissant que la sécurité n'est plus un ajout tardif, mais un élément central du processus de développement. Cela répond à la nécessité d'une collaboration renforcée entre les domaines de la sécurité et du développement logiciel pour assurer la robustesse des systèmes dès leur conception.

L'intégration continue de la sécurité, un concept clé de DevSecOps, permet d'identifier et de corriger les vulnérabilités dès les premières étapes du développement. En incorporant des pratiques de sécurité dès la phase de planification et en les maintenant tout au long du cycle de vie du logiciel, le DevSecOps réduit les risques de failles de sécurité et minimise les coûts liés aux corrections ultérieures. Les travaux de T.Chen et H.Suo [3] montrent que cette approche non seulement améliore l'efficacité des équipes, mais réduit aussi les cycles de publication, tout en diminuant le nombre de bugs en production, ce qui contribue à une meilleure qualité et sécurité du produit final.

En définitive, le DevSecOps joue un rôle crucial dans la préservation de la cohérence entre les modèles de systèmes et leur sécurité, en garantissant une réponse proactive aux menaces. En intégrant la sécurité dès le début et en favorisant la collaboration entre les équipes de développement, de sécurité et d'exploitation, le DevSecOps assure que les systèmes développés sont non seulement fonctionnels, mais aussi résilients face aux défis sécuritaires actuels. Cette méthode est donc essentielle pour prévenir les failles de sécurité et assurer la robustesse et la fiabilité des produits logiciels dans un environnement en constante évolution.

5.2.2. Modélisation pour l'évaluation d'un niveau de sécurité

Dans le cadre des démarches de sécurisation des systèmes d'information, en particulier dans les pratiques DevSecOps, on retrouve parmi les outils couramment utilisés les Data Flow Diagrams (DFD). Ce sont des outils visuels qui modélisent les flux d'information au sein d'un système, en mettant en évidence la manière dont les données circulent entre les processus, les stockages de données, les entités externes et les flux de données eux-mêmes. Introduits et popularisés par E.Yourdon [20] et T.DeMarco [4] en 1979, les DFD se composent de quatre éléments principaux : les processus qui représentent les transformations ou manipulations des données, les flux de données qui indiquent les mouvements d'informations entre les éléments du système, les dépôts de données où les informations sont stockées et les entités externes qui interagissent avec le système mais sont extérieures à celui-ci.

Ces diagrammes servent à clarifier la structure fonctionnelle d'un système d'information, facilitant ainsi la détection de failles potentielles dans les flux de données. Comme l'explique A.Shostack [14], les DFD sont particulièrement utiles dans le contexte de l'analyse des menaces, car ils permettent d'identifier

les points où les données pourraient être compromises. En représentant visuellement les interactions entre les composants du système, les DFD aident les équipes DevSecOps à anticiper les vulnérabilités, à renforcer les contrôles de sécurité, et à s'assurer que les mesures de protection sont intégrées là où elles sont les plus nécessaires.

L'utilisation des DFD dans les pratiques DevSecOps, en particulier pour la sécurisation des flux de données, s'avère donc essentielle pour garantir la robustesse des systèmes face aux menaces. En fournissant une vue d'ensemble claire et structurée des mouvements de données, les DFD permettent non seulement de mieux comprendre les systèmes complexes, mais aussi de guider le développement sécurisé et d'assurer une gestion efficace des risques tout au long du cycle de vie du logiciel.

Cependant, les DFD traditionnels présentent des limitations en matière de sécurité. Pour pallier cette insuffisance, les secDFD ont été développés. Ils intègrent des annotations de sécurité directement dans les diagrammes, ce qui permet une évaluation plus fine dès la conception d'un système des risques associés aux transferts de flux de données. En annotant les DFD avec des informations telles que les exigences de confidentialité, d'intégrité et d'authentification, il devient possible de détecter plus précocement les vulnérabilités potentielles au sein d'un système en évolution.

En effet, les approches plus classiques basées sur l'automatisation de la détection de menaces sur des DFD doivent prendre en compte un plus grand nombre de possibilités en raison du manque d'informations spécifiques à la sécurité sur les diagrammes. Cela peut induire un temps de recherche des menaces qui croît de manière exponentielle en fonction de la taille du diagramme en entrée, on parle alors d'explosions combinatoires. Des premières propositions ont conduit à l'ajout d'extensions pour limiter le nombre de menaces identifiées et l'effort nécessaire à leur traitement [15]. Cependant, ces extensions peuvent, si elles sont mal utilisées, conduire à l'omission de menaces de sécurité ou de confidentialité non réellement atténuées.

Les secDFD, en revanche, permettent de réduire cet espace d'élucidation des menaces en s'appuyant sur la connaissance des solutions de sécurité existantes dans le système en cours de conception. Cette approche améliore la qualité sémantique et la traçabilité des effets de sécurité, en veillant à ce que les décisions d'appliquer ces solutions restent bien documentées. L'utilisation des secDFD formalise le niveau de sécurité d'un système, facilitant ainsi l'évaluation rigoureuse des flux d'information pour identifier et corriger les défauts de conception ou d'implémentation. Cette formalisation est essentielle pour une analyse d'impact précise, permettant de détecter les failles de sécurité potentielles et d'assurer une prise de décision éclairée au niveau architectural [18]. Bien que cette approche introduise une complexité supplémentaire, celle-ci reste gérable, d'autant plus si des solutions visant à l'automatisation de la correspondance du modèle avec le système sont apportées.

5.2.3. La cohérence dans les cycles de développement

La divergence entre les modèles de systèmes et leur implémentation pose un défi important dans le développement logiciel, particulièrement dans un cadre DevSecOps où la sécurité est intégrée tout au long du cycle de vie du développement. Lorsque les modèles et le code ne sont pas alignés, les risques de vulnérabilités de sécurité augmentent, et la maintenance devient plus complexe. La conséquence directe est une perte de confiance dans les modèles utilisés pour analyser et garantir la sécurité du système, ce qui peut conduire à des failles non détectées et donc à des coûts de correction accrus en phase de post-déploiement.

Pour prévenir ces divergences, des méthodes de co-évolution du code et des métamodèles ont été développées. Ces approches permettent de maintenir une cohérence continue entre le modèle et le code en détectant automatiquement les changements dans le métamodèle et en les propageant dans le code. Cette stratégie [9] garantit que toute modification dans la structure conceptuelle d'un système est immédiatement reflétée dans son implémentation. Ainsi, le système reste cohérent avec ses modèles au fil des évolutions, évitant les incohérences qui pourraient introduire des vulnérabilités.

De plus, pour améliorer la traçabilité et faciliter la détection des incohérences, il est crucial de mettre en place des liens explicites et sérialisés entre les modèles et le code. Ces liens [8] permettent une vérification automatique de la cohérence, en identifiant rapidement les divergences potentielles. On obtient ainsi une traçabilité qui garantit que chaque changement dans le modèle est correctement implémenté dans le code, réduisant le risque d'erreurs et améliorant la sécurité globale du système.

Pour assurer la robustesse et la sécurité des systèmes développés dans un environnement DevSecOps, il donc est indispensable de maintenir une cohérence stricte entre les modèles et le code. Les méthodes de co-évolution et les mécanismes de traçabilité offrent des solutions efficaces pour répondre à ces défis, permettant de prévenir les failles de sécurité et de garantir une architecture de système cohérente et sécurisée tout au long de son cycle de vie.

5.2.4. Automatisation de la connexion entre deux modèles

L'automatisation de l'intégration de la sécurité dans le cycle DevOps représente une avancée significative pour répondre aux exigences d'efficacité et de rapidité dans le développement logiciel moderne. Dans un contexte où les itérations sont courtes et fréquentes, comme c'est le cas avec les méthodologies Agile et DevOps, l'intégration manuelle des pratiques de sécurité peut devenir un obstacle. Les processus de vérification de la sécurité, traditionnellement lourds et nécessitant une expertise approfondie, s'accordent difficilement avec ces cycles de développement rapides. Tuma et al. [19] soulignent que l'agilité et la rapidité sont essentielles dans le développement logiciel contemporain, mais cela ne doit pas se faire au détriment de la qualité ou de la sécurité. C'est ici qu'intervient l'automatisation, en permettant d'intégrer des vérifications de sécurité sans ralentir le rythme du développement.

L'automatisation des processus de sécurité présente plusieurs avantages majeurs. Elle permet de rendre les vérifications de sécurité moins consommatrices de temps et d'énergie, s'intégrant harmonieusement dans les cycles courts et itératifs du DevOps. En automatisant la détection précoce des failles de sécurité, les équipes de développement peuvent identifier et corriger les vulnérabilités dès les premières étapes du cycle de vie du logiciel, réduisant ainsi les coûts et les efforts liés à la correction de failles découvertes en post-déploiement. De plus, l'automatisation facilite l'adoption des outils de sécurité au sein des pratiques des entreprises. En abaissant les barrières à l'entrée, elle rend ces outils plus accessibles, ce qui permet aux entreprises de réaliser des économies tout en améliorant leur posture de sécurité.

Cependant, l'automatisation de la sécurité dans le cycle DevOps ne va pas sans défis. Tuma et al. [19] identifient plusieurs besoins essentiels pour que l'automatisation soit efficace. D'abord, il est nécessaire de formaliser les modèles utilisés pour la détection des failles. La précision et la constance des modèles sont cruciales, car des notations informelles ou des ambiguïtés dans les modèles peuvent entraîner des erreurs de détection. Par exemple, si un modèle ne représente pas correctement les actifs sensibles, la détection automatique pourrait ne pas identifier les expositions de données non sécurisées. De plus, l'automatisation dépend fortement de la capacité des modèles à refléter fidèlement les intentions du modélisateur. Des malentendus ou des erreurs dans la modélisation peuvent conduire à des analyses incorrectes, compromettant ainsi la détection des failles de sécurité.

Malgré ces défis, les résultats obtenus par l'automatisation dans la détection des failles de sécurité sont encourageants, bien que limités. Comme le mentionne Tuma et al. [19], si l'automatisation ne peut pas encore remplacer complètement l'expertise humaine en matière de sécurité, elle offre une valeur ajoutée en fournissant une liste de problèmes potentiels à examiner par un expert. Cette approche de semi-automatisation, où l'automatisation est utilisée pour assister les experts plutôt que pour les remplacer, apparaît comme un compromis prometteur. Par exemple, dans le travail de Tuma et al. [17] sur la vérification de la conformité de la sécurité entre un modèle secDFD et le code Java, la semi-automatisation a démontré des résultats très encourageants. L'approche développée, qui associe des suggestions automatisées avec l'intervention humaine pour affiner les correspondances, a montré une grande précision et a permis de

réduire les fausses alertes.

5.3. Précisions et évolution de la mission

Après avoir introduit les notions et concepts permettant de contextualiser le sujet du stage, nous pouvons explorer plus en détail le dernier article cité car il présente un élément essentiel du stage.

L'article "Checking Security Compliance between Models and Code" de Tuma et al. [17] s'inscrit dans la continuité des travaux de recherche menés par Katja Tuma et Sven Peldzus. Ce travail est le fruit de deux thèses, chacune contribuant à l'amélioration des méthodes de sécurisation des systèmes logiciels en phase de développement. L'article se divise en deux parties principales : la première, sur laquelle nous nous concentrerons ici, explore les méthodes permettant de connecter semi-automatiquement un modèle secDFD avec le code Java qu'il représente. La seconde partie propose une analyse approfondie de la sécurité des systèmes basée sur ces secDFD, justifiant ainsi leur pertinence et leur efficacité.

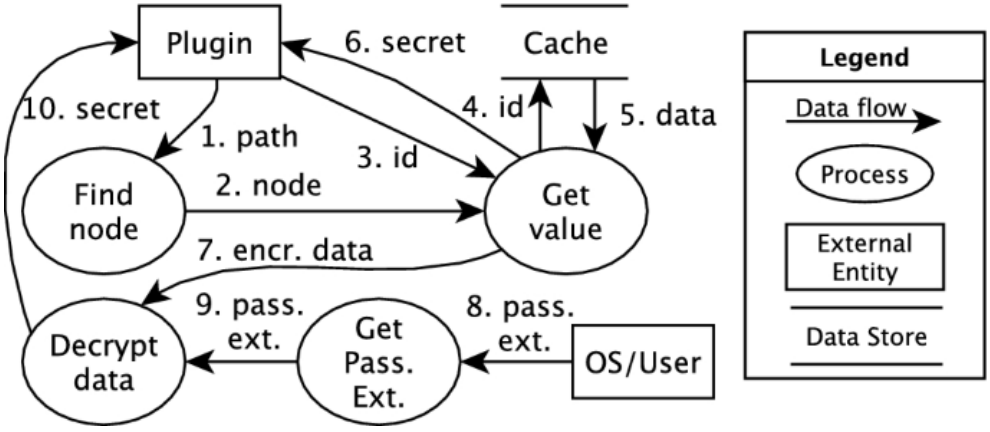

FIGURE 1 – Extrait de DFD correspondant à "Eclipse Secure Storage" (Source[17])

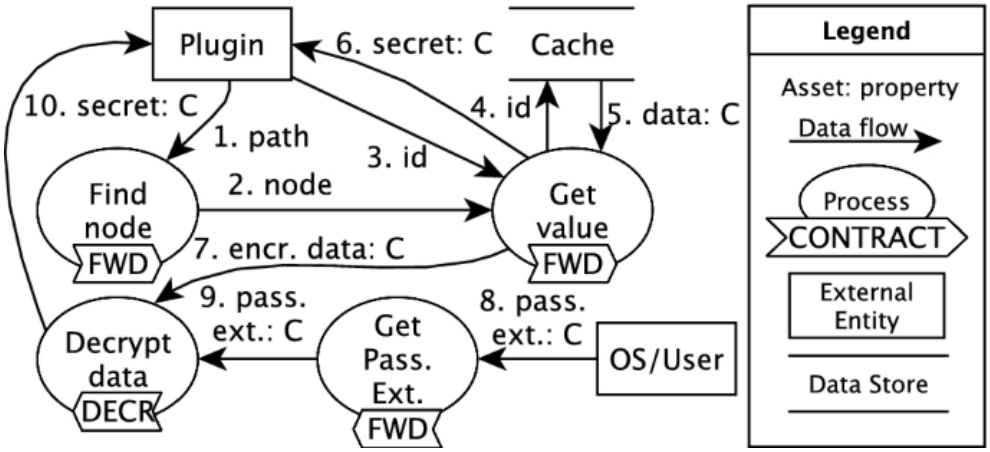

FIGURE 2 – Extrait de secDFD correspondant à "Eclipse Secure Storage" (Source[17])

Les secDFD (Figure 2) représentent une évolution des traditionnels DFD (Figure 1), enrichis par des annotations de sécurité. Ces annotations comprennent des étiquettes de sécurité sur les flux de données, telles que des marquages pour indiquer un niveau élevé de confidentialité, des contrats de sécurité sur les

processus pour s'assurer de leur conformité avec les données manipulées, et l'identification de zones de confiance. Ces ajouts permettent d'évaluer les risques dès la phase de conception, en identifiant les zones potentiellement vulnérables aux attaques externes .

La première partie de l'article, qui nous intéresse particulièrement, se concentre sur la connexion entre le modèle secDFD et le code source, en l'occurrence du code Java. Cette connexion est facilitée par l'extraction d'un "Program Model" (PM) à partir du code, une version modifiée d'un "Abstract Syntax Tree" (AST). Ce Program Model permet de visualiser les appels de méthodes ainsi que leurs paramètres, offrant une meilleure compréhension de l'exécution du code. Pour renforcer l'efficacité de cette connexion, une méthode de recherche itérative est employée, explorant le Program Model pour établir des correspondances entre le modèle et le code.

La méthodologie adoptée repose sur des heuristiques qui proposent à l'utilisateur des correspondances possibles entre le modèle et le code, d'où le caractère semi-automatique de la procédure. Ces heuristiques s'appuient sur plusieurs critères, tels que la similitude des noms, mesurée par la distance de Levenshtein, ainsi que sur les paramètres et les appels de méthodes, identifiés grâce au Program Model. Des règles d'association spécifiques sont également définies, comme l'association entre les processus du modèle et les méthodes du code, ou encore entre les donées des flux du modèle et les types du code.

Une fois les suggestions proposées, l'utilisateur sélectionne les correspondances les plus pertinentes, et le processus continue de manière itérative jusqu'à convergence. Les résultats obtenus montrent une grande précision et un rappel élevé, bien que ces performances varient selon les projets analysés. Ce processus de semi-automatisation démontre l'efficacité de la méthode, permettant une intégration plus fluide de la sécurité dans le cycle de développement tout en minimisant les risques de divergence entre le modèle et le code.

Aussi, suite à la lecture de cet article en début de stage, une nouvelle orientation a été décidée quant aux objectifs du stage : Il s'agit de la reproduction de la solution de connexion automatique entre secDFD et un code Java présentée dans l'article de K. Tuma [17] tout en y intégrant des outils supplémentaires pour la détection automatique des changements et de notification de ceux-ci. Cette approche vise à assurer la cohérence de la sécurité lors des évolutions du système.

5.4. Choix des outils et apprentissages

Dans cette partie, nous allons examiner les différents outils étudiés et utilisés dans le stage en commençant par une description générale de ceux-ci puis nous discuterons de leurs avantages respectifs permettant de justifier leur choix.

5.4.1. Spoon

Spoon m'a été conseillé dès le début du stage comme outil d'analyse et de manipulation du code source d'un projet java. Au-delà de permettre l'exploration d'un nouvel outil dans le cadre des recherches au sein de l'équipe P4S, celui-ci présente de nombreux avantages pour la réalisation de la mission.

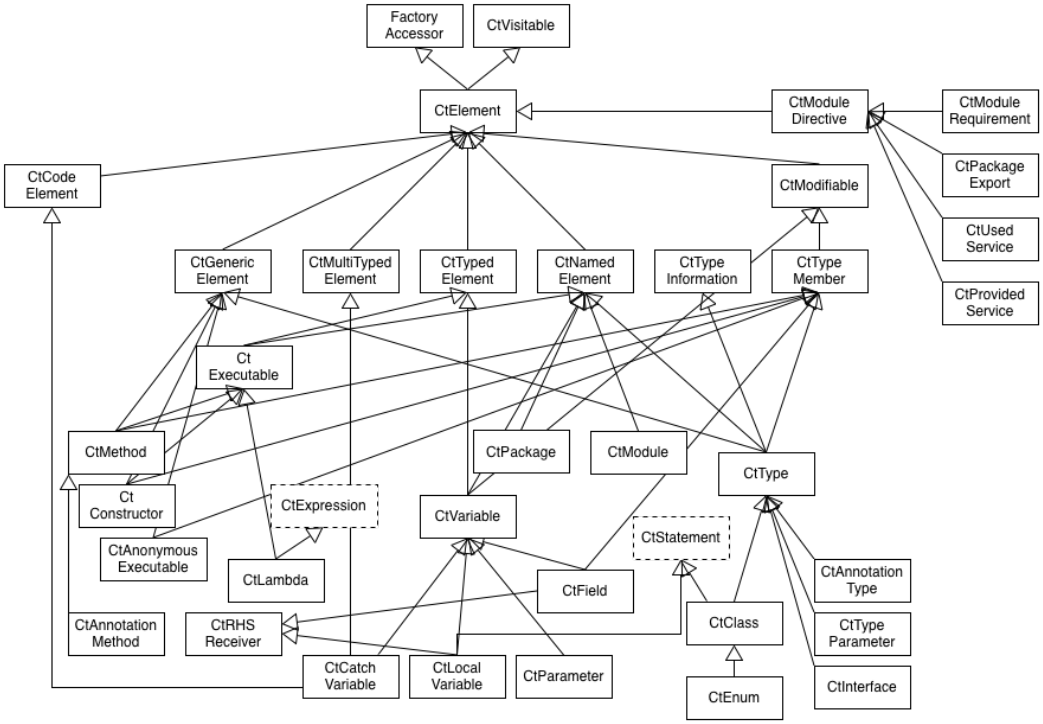

FIGURE 3 – métamodèle éléments structurels Spoon (Source[16])

Spoon [13] est un outil open-source permettant d'extraire et de manipuler les arbres de syntaxe abstraite (AST) à partir de projets Java. Conçu pour faciliter l'analyse et la transformation du code source Java, Spoon offre une plateforme robuste pour extraire les représentations des programmes et les manipuler de manière fine, sans pour autant nécessiter la modification directe du code source.

Pour pouvoir utiliser Spoon, il faut d'abord configurer un environnement Java dans lequel Spoon est intégré. Ensuite, à partir du code source Java, Spoon extrait l'AST, une représentation hiérarchique du code source sous forme d'objets (Figure 3). Ces objets peuvent être parcourus, modifiés ou même entièrement réécrits selon les besoins. Spoon fournit également des outils pour analyser les dépendances, détecter les patterns ou les anomalies dans le code, et transformer le code source à grande échelle. L'outil se distingue par sa capacité à maintenir la fidélité entre le code source original et l'AST généré, ce qui permet des modifications précises.

L'une des fonctionnalités intéressantes de Spoon est la possibilité de sérialiser l'AST, c'est-à-dire de le sauvegarder pour une utilisation ultérieure. Cette sérialisation est particulièrement utile pour garder une trace des modifications au fil du temps et pour comparer différentes versions du code. De plus, Spoon offre des performances élevées même lorsqu'il est appliqué à de grands projets Java, grâce à une gestion efficace des ressources et à une optimisation de l'extraction des AST. En effet, l'extraction de l'AST, qui est de loin l'opération la plus coûteuse en temps dans l'utilisation de Spoon sur l'ensemble d'un projet, est dans une majorité de cas largement en dessous de son temps de compilation.

L'utilisation de Spoon dans le cadre du stage est donc avantageux pour plusieurs raisons. Sa simplicité d'utilisation permet une intégration rapide dans les processus de développement, tout en offrant des performances adéquates pour traiter des projets de grande envergure. La capacité à sérialiser les modèles AST permet de conserver les représentations du code source au fil du temps, facilitant ainsi la détection de modifications.

5.4.2. Xtext/EMF

Du côté des modèles secDFDs, l'outil utilisé pour les décrire et les manipuler dans l'article de référence est Xtext, un environnement de création de langages basé sur EMF. Celui-ci est utilisé pour créer une abstraction générale de ce qu'est formellement un secDFD, on parle alors de méta-modèle à partir duquel, il est ensuite possible de créer des instances. La conservation de cet outil dans le projet n'est pas uniquement dû à la praticité de pouvoir utiliser les modèles secDFD qui ont servi d'exemples dans le papier de référence mais aussi pour son efficacité et la possibilité de détecter les modifications du modèle.

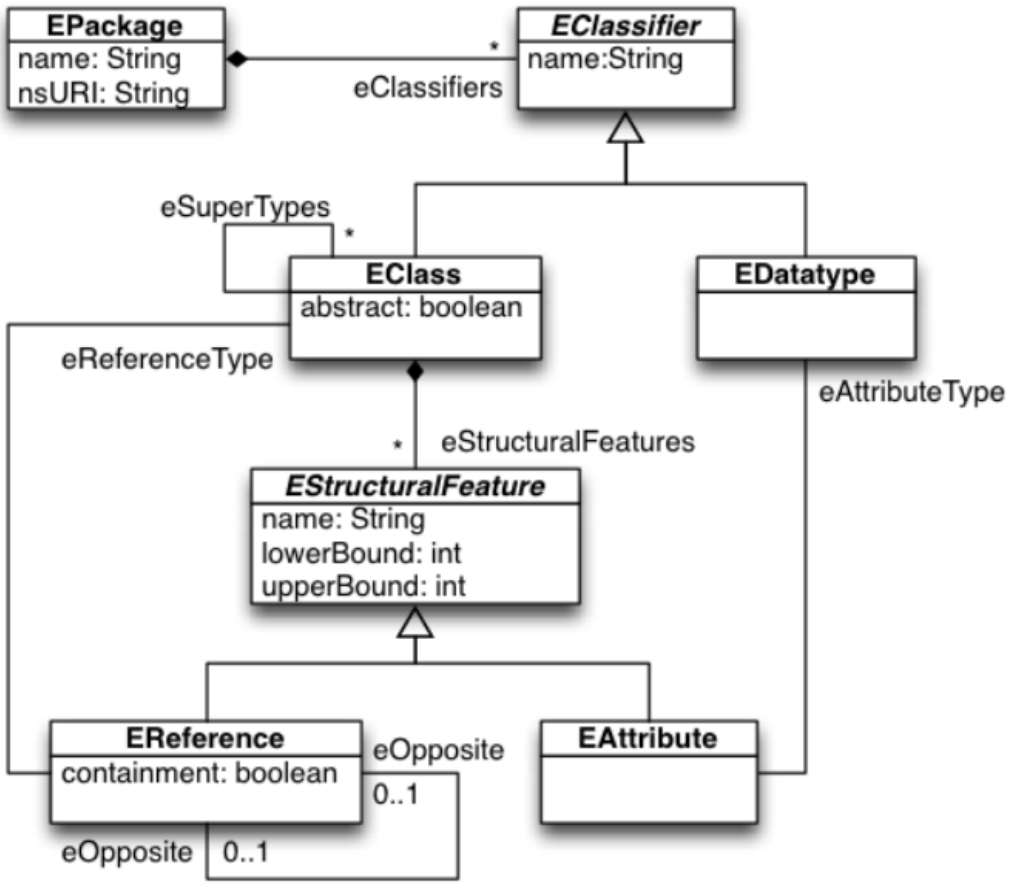

FIGURE 4 – Principaux éléments du méta-modèle d'EMF (Source [6])

Plus précisément, Eclipse Modeling Framework (EMF) est un environnement utilisé pour modéliser des données structurées dans des applications Java et de générer du code Java à partir de ces modèles. Conçu pour simplifier la création et la maintenance de modèles complexes, EMF permet de créer des modèles à partir de diverses sources, telles que des schémas XML, des diagrammes UML, ou même des annotations Java. Ces modèles (Figure 4) représentent des objets, leurs propriétés, leurs relations, ainsi que les contraintes qui les régissent. Une fois définis, EMF peut générer automatiquement du code Java correspondant à ces modèles, incluant des classes Java, des mécanismes de sérialisation pour la persistance des données, et des éditeurs pour la manipulation des instances de modèles. Un des aspects clés d'EMF est qu'il offre un support natif pour la gestion des événements, permettant de détecter et de propager automatiquement les modifications à travers le système en plus de permettre la sérialisation pour la persistance des données.

C'est dans ce contexte que Xtext intervient, il apporte une dimension supplémentaire en permettant

de formaliser et de représenter un modèle de manière textuelle. Construit au-dessus d'EMF, Xtext utilise EMF pour gérer la représentation interne des modèles manipulés, il se distingue par sa capacité à créer des langages spécifiques à un domaine (DSL). Ces DSL permettent de décrire des modèles sous forme de texte, qui sont ensuite transformés en objets EMF manipulables en Java.

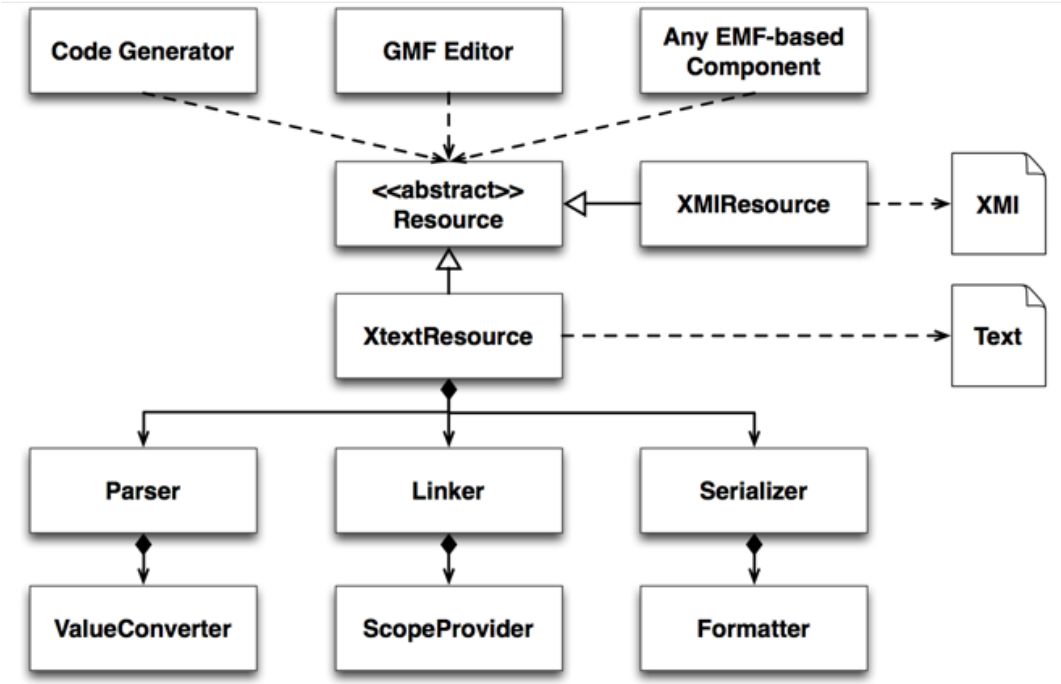

FIGURE 5 – Implémentation de XtextRessourceSet avec EMF (Source[6])

En effet, en définissant une grammaire spécifique pour un domaine donné, Xtext génère un parseur qui convertit le texte en objets EMF selon cette même grammaire. Ces objets peuvent ensuite être utilisés pour les mêmes opérations que les modèles EMF classiques ce qui permet d'utiliser tout l'écosystème EMF pour manipuler, valider, et sauvegarder ces objets, mais avec l'avantage supplémentaire de pouvoir représenter et formaliser le modèle sous une forme textuelle.

En résumé, EMF et Xtext permettent de modéliser un système de manière cohérente et extensible, de générer du code Java à partir de ces modèles. Ces outils sont donc pratiques pour traduire un modèle secDFD en objets Java manipulables tout en assurant la gestion des événements et la propagation des modifications, facilitant ainsi le maintien de la cohérence entre les modèles et le système au cours de son évolution.

5.4.3. Epsilon

En ce qui concerne l'association de modèles, je me suis tourné vers Epsilon suite à des conseils, discussions et recherches. Celui-ci présente en effet de nombreux avantages pour ses performances, sa capacité de formalisation les règles de connexion entre les modèles et la possibilité de garder en mémoire un modèle fusionné représentant les connexions pour la détection de modifications.

Epsilon est une suite de langages de script et d'outils dédiée à l'automatisation des tâches courantes en ingénierie logicielle orientée modèle (Model-Based Software Engineering). Elle permet la génération de code, la transformation de modèle à modèle, la validation de modèles et la visualisation de modèles, tout en étant compatible avec diverses technologies telles qu'EMF, UML, Simulink, XML, et autres. Epsilon se distingue par une approche unifiée où tous les langages de la suite partagent une syntaxe commune

basée sur EOL (Epsilon Object Language), facilitant la réutilisation du code à travers différentes tâches et garantissant une cohérence syntaxique.

Un des avantages majeurs d'Epsilon réside dans son support solide pour EMF. Cela est rendu possible grâce à la couche de connectivité des modèles d'Epsilon, connue sous le nom d'EMC (Epsilon Model Connectivity) (Figure 6). Les drivers EMC, tels que ceux pour EMF, XML, ou Simulink, permettent à Epsilon de se connecter et de manipuler divers types de modèles de manière transparente. De plus, Epsilon n'est pas limité à l'environnement Eclipse, ce qui le rend flexible pour une utilisation dans des applications Java ou Android indépendantes via des fichiers JAR.

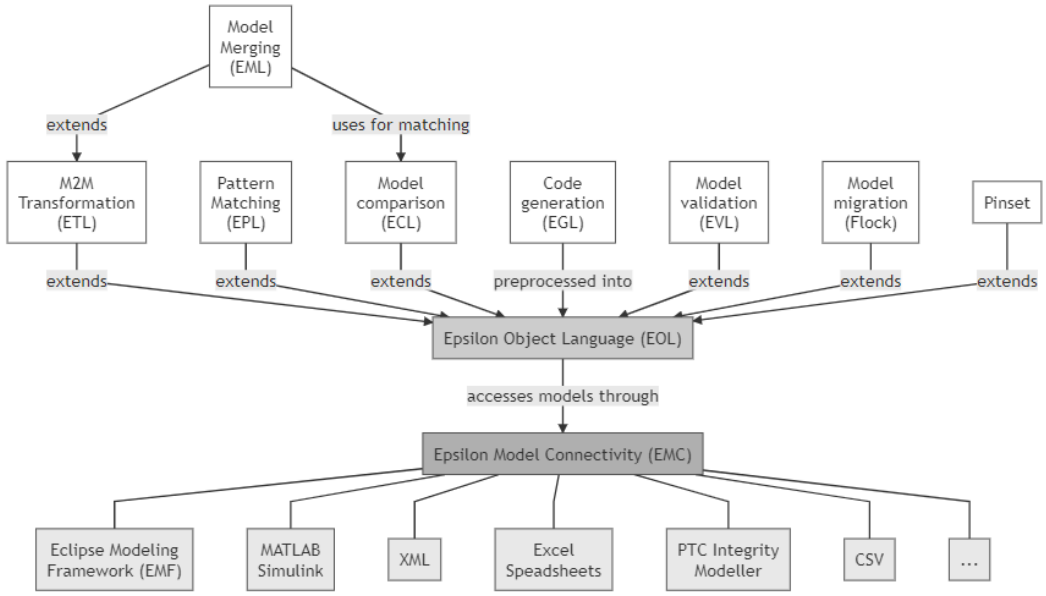

FIGURE 6 – Structure générale d'Epsilon (Source[5])

EOL (Epsilon Object Language) est au cœur de la suite Epsilon. C'est le langage d'expression fondamental sur lequel reposent les autres langages spécifiques aux tâches d'Epsilon, comme la validation de modèles, la transformation de modèles, et la migration de modèles. EOL peut également être utilisé comme un langage de gestion de modèles à usage général, permettant l'automatisation de tâches qui ne sont pas couvertes par les langages spécifiques. Un programme EOL est structuré en modules, chaque module définissant un ensemble d'opérations qui sont exécutées dans un bloc de code principal (Figure 7). Ces opérations sont contextuelles, c'est-à-dire qu'elles s'appliquent à un type spécifique d'objet, et peuvent être partagées entre modules via des instructions d'importation, ce qui favorise la modularité et la réutilisation du code.

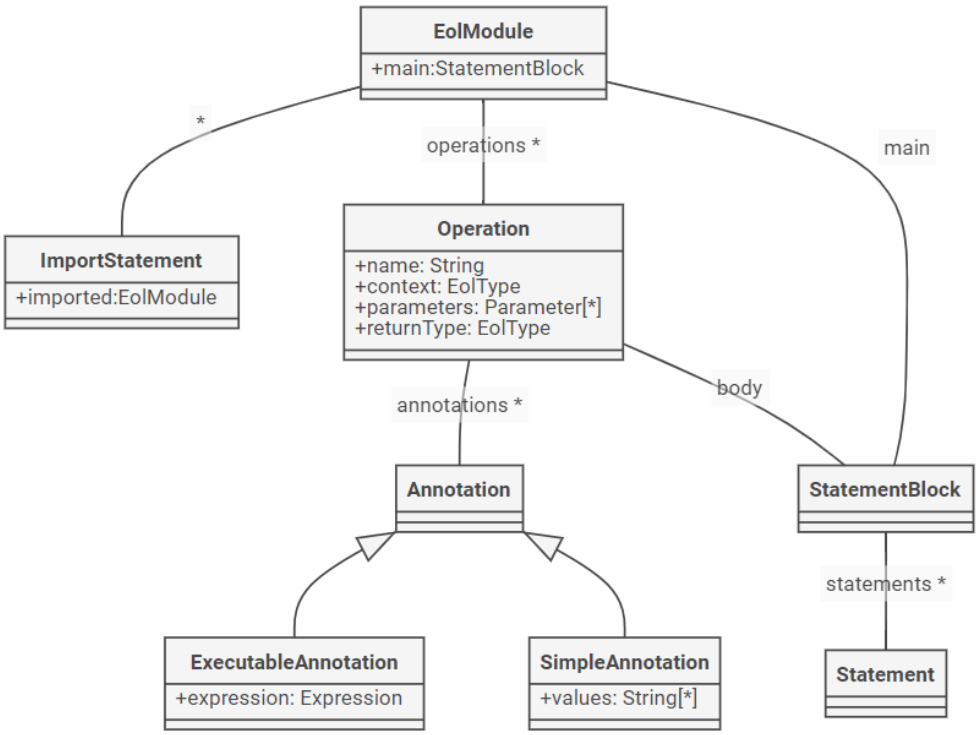

FIGURE 7 – Syntaxe d'EOL (Source[5])

L'EML (Epsilon Merging Language) (Figure 9) est un langage qui étend les capacités d'EOL en combinant les fonctionnalités d'ECL (Epsilon Comparison Language) (Figure 8) et d'ETL (Epsilon Transformation Language). EML est conçu pour faciliter la fusion de modèles en définissant formellement des règles d'association entre les éléments de différents modèles et en générant un modèle fusionné qui conserve les connexions établies. L'utilisation d'EML présente plusieurs avantages notables dans le contexte de la fusion de modèles. Tout d'abord, EML permet de formaliser de manière rigoureuse les règles d'association entre les deux modèles, ce qui garantit une clarté et une précision accrues dans le processus de fusion. Le modèle fusionné peut ensuite être sérialisé, ce qui facilite la traçabilité et la gestion des connexions établies. De plus, grâce aux fonctionnalités avancées d'ECL, EML permet de créer des connexions complexes avec des conditions multiples, telles que l'utilisation de la distance de Levenshtein pour mesurer la similarité entre éléments. Enfin, la capacité de traitement multi-threadé d'Epsilon assure une fusion des modèles à la fois efficace et performante, en exploitant au mieux les ressources disponibles pour optimiser les connexions entre les modèles.

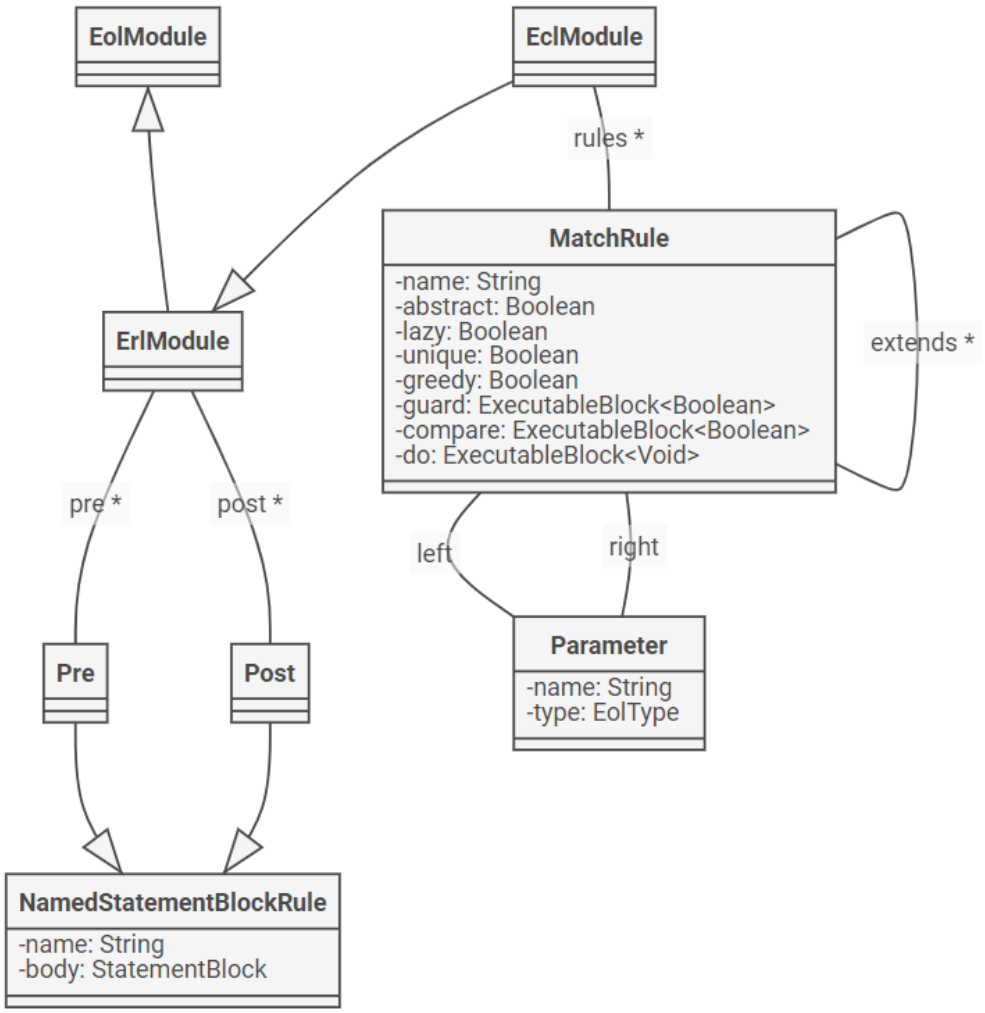

FIGURE 8 – Modèle d'abstraction d'ECL (Source[5])

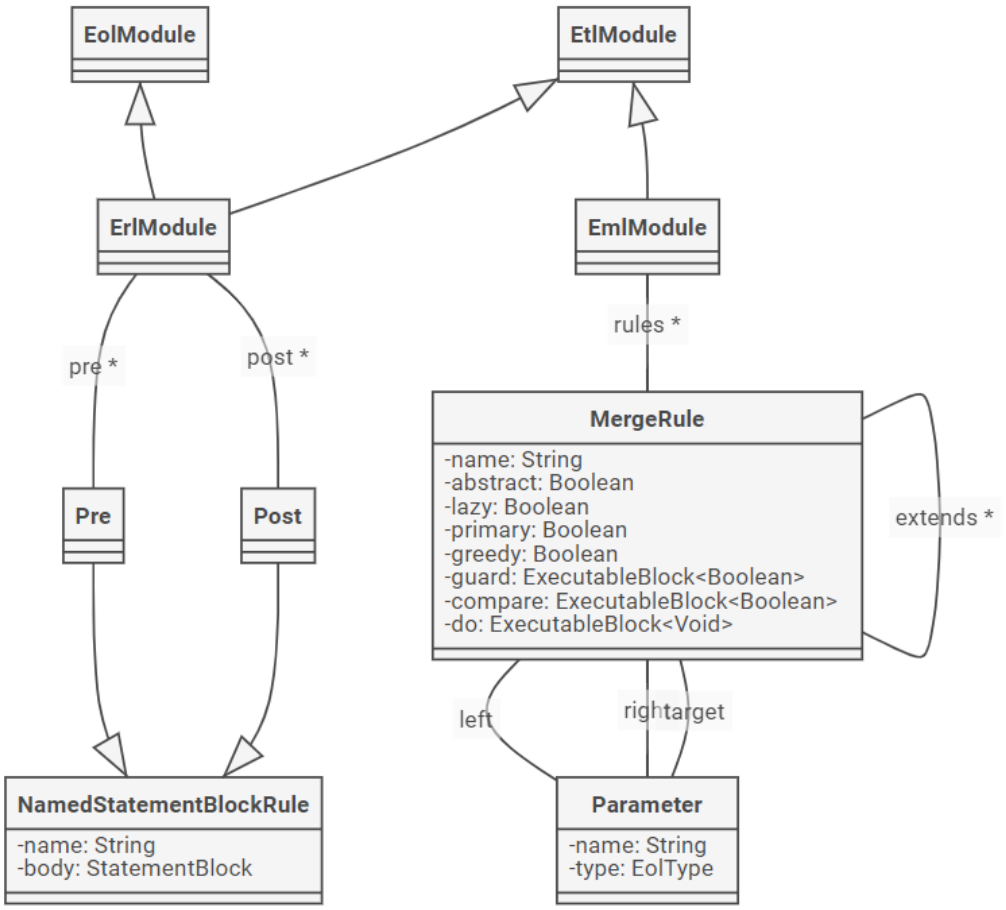

FIGURE 9 – Modèle d'abstraction d'EML (Source[5])

5.5. Conception, implémentation et perspectives

5.5.1. Conception

Maintenant que les principales pièces du puzzle ont été présentées, il ne reste plus qu'à les assembler. Ainsi, suite aux différents apprentissages et tests, je suis parvenu à la conception d'une architecture pour répondre à la problématique principale du stage (Figure 10).Reprenons donc la problématique par étapes afin de construire une architecture adaptée.

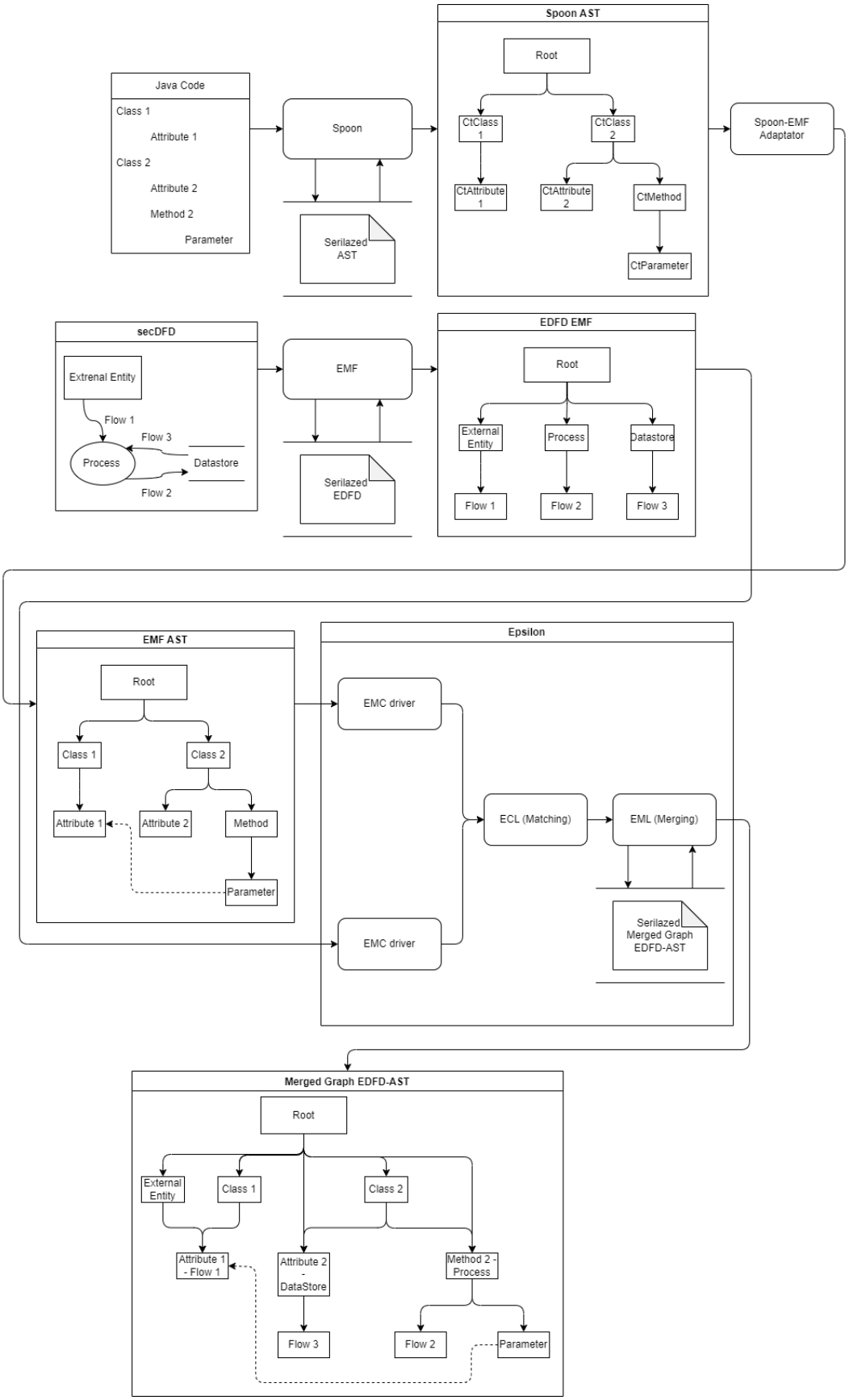

FIGURE 10 – Architecture de solution

Le premier temps consiste à reproduire en partie l'implémentation proposée, partant d'un modèle secDFD dans un fichier Xtext d'extension ".secdfd" et d'un projet ou d'une partie de projet écrit en java qu'il représente. Cependant, cette reproduction doit être pensée dans l'optique d'y ajouter des fonctionnalités de détection de modification et de correction de modification. Cela commence par extraire une représentation du code java manipulable. Pour effectuer cette tache, Spoon comme nous l'avons vu précédemment est tout à fait adapté. En effet, l'extraction rapide d'un AST très complet. Un de ses nombreux avantages réside dans le fait d'être proche du code offrant ainsi la possibilité de choisir les informations sur le code que nous souhaitons garder pour améliorer la précision, je fais là référence à la possibilité de reproduire le Program Model décris dans l'article de K. Tuma [17]. De plus concernant la cohérence, la possibilité de garder l'AST en mémoire va permettre par la suite de mettre en place des solutions de détection de modifications par comparaison. Cependant puisque cette représentation du code va être amenée à être manipulée au travers d'Epsilon, il va falloir l'adapter. Ainsi l'introduction d'un adaptateur va permettre de passer de l'AST fournit par Spoon vers une instance d'un modèle EMF représentant le code Java.

Ensuite, du côté secDFD il s'agit de récupérer une version manipulable à partir des fichiers d'extension ".secdfd" existants. L'environnement Xtext, défini dans la solution proposé par le papier de référence, offre des fonctionnalités pour obtenir une instance d'un modèle EDFD de ces fichiers et cela même hors de l'instance d'Eclipse que Xtext génère (Figure 5). Il s'agit ici des fonctionnalités "Standalone" permettant de manipuler les fonctionnalités de Xtext hors du contexte de celui-ci dans une application java indépendante. Le modèle EDFD fournit est un modèle EMF permettant de représenter un secDFD dans un objet java manipulable. Or autour de cet instance du modèle EDFD, il existe des événements afin de notifier une modification permettant de déclencher une propagation des mises à jour pour maintenir la cohérence.

Après l'extraction des deux modèles en tant qu'instances d'un modèle EMF, il faut établir les connexions entre ceux-ci, pour cela c'est la suite Epsilon qui a fait l'objet de mon choix pour les raisons précédemment expliquées. Ainsi grâce au driver EMF d'EMC (Epsilon Model Connectivity), il est possible d'interagir directement avec les instances de modèle EMF. Les deux modèles ainsi récupérés vont ensuite être comparés au travers de règles définies dans le langage ECL reprenant les critères heuristiques présentés dans le papier de K.Tuma [17]. Cela inclut une sélection sur le type, par exemple les processus associés avec des méthodes, une sélection sur la proximité des noms avec une contrainte sur la distance de Levenshtein ainsi que la prise en compte des types des paramètres pour les méthodes. Pour des améliorations futures, ECL offre la possibilité de gérer les matchs multiples après une première phase de comparaison et cela permettrai de raffiner la comparaison. De plus, étant donné que l'on peut introduire du code java dans le code ECL au travers d'outils, on peut envisager l'introduction d'une interface utilisateur pour reproduire la solution de l'article de référence. Une fois les modèles comparés, on obtient alors des connexions qui vont être traduites par la fusion des deux modèles. Cela se fait au travers du langage EML (Epsilon Merging Language) qui conserve la structure des deux modèles pour les parties qui n'ont de connexions et formalise la sortie après fusion, pour les parties connectées.

On se retrouve alors avec un modèle fusionné représentant les connexions établies entre les deux modèles en entrée. Celui-ci étant conservé pour permettre par la suite d'intégrer des fonctionnalités en vue de notifier à quels endroits, un changement du côté du secDFD ou du code, l'autre côté est impacté.

5.5.2. Implémentation

Étant encore en cours d'implémentation, je n'ai pas d'implémentation fonctionnelle de la conception précédemment expliquée à l'heure de l'écriture de ce rapport. J'ai néanmoins implémenté des petits projets modulaires permettant de tester et de faire fonctionner les outils afin de les intégrer dans l'implémentation finale.

Mais pour permettre l'implémentation au cours du stage, il me fallait d'abord définir des environnements de test et de développement. Sachant que les outils et projets sont implémentés en java et

sont manipulables dans ce même langage, il fallait que ceux-ci soient adaptés à ce langage. Cela commence d'abord par un outil de gestion des paquets et de la construction de projets java pour faciliter l'importation de dépendances, automatiser la compilation et l'empaquetement dans des fichier jar et donc faciliter les tests. Et parmi les deux principaux qui sont Maven [11] et Gradle [7], c'est Maven qui a fait l'objet de mon choix. En effet, pour la majorité des projets que j'ai étudiés au début de mon stage, Maven en était l'automate de production me forçant ainsi à comprendre plus en détails son fonctionnement. Puis la facilité de gestion des dépendances et des métadonnées avec Maven et la standardisation que celui-ci produit m'ont conforté dans le choix de le conserver malgré la moindre performance de Maven face à Gradle en temps dans la phase de build.

Ensuite, j'ai effectué l'implémentation de manière modulaire et incrémentale. Ainsi, cela me permet de tester les différentes fonctionnalités des parties que je développe indépendamment les unes des autres et pouvoir ensuite les connecter. Pour cela, j'ai créé des petits projets Maven afin de pouvoir gérer facilement les dépendances nécessaires pour chacun des outils.

J'ai commencé par essayer les fonctionnalités de Spoon, celui-ci étant facile à prendre en main, j'ai rapidement pu obtenir l'AST d'un projet java et le manipuler. Ensuite, j'ai pu implémenter les fonctions de sauvegarde et chargement pour sérialiser un AST et le récupérer ensuite. Il va maintenant falloir créer un adaptateur pour convertir cet AST ou plutôt une partie de celui-ci en une instance d'un modèle EMF qu'il faudra aussi créer.

Concernant la partie secDFD, celle-ci fut plus complexe pour moi. En effet, Xtext et EMF présentent une barrière à l'entrée plutôt importante et nécessitent une compréhension approfondie de leur fonctionnement pour pouvoir être utilisés dans plusieurs contextes. Et bien que Eclipse, en tant qu'IDE intégrant des plugins adaptés à ces environnements, puisse simplifier l'apprentissage, en cachant un grand nombre d'opérations, les bugs et incompréhensions peuvent rapidement subvenir dès que l'on essaye de s'éloigner de l'utilisation standard de ces environnements. Ainsi n'ayant pas ou très peu de notions dans ce type d'environnements, j'ai dû commencer par créer des projets simples pour comprendre les mécanismes essentiels. Ensuite, j'ai essayé d'implémenter l'extraction du model EMF à partir d'un fichier Xtext d'extension ".secdfd" dans une application java standalone pour pouvoir récupérer un objet manipulable à partir de ces modèles. Cependant je me suis heurté à un bug que je n'ai pas réussi à résoudre. En effet, pour réaliser l'extraction, il faut d'abord initialiser l'environnement "standalone" ce qui permet d'obtenir une "Factory" dont la fonction est de pouvoir lire un fichier de terminaison ".secdfd" pour ensuite créer une ressource (Figure 5) à partir de celui-ci dans laquelle le modèle manipulable y est disponible . Cependant, au moment de la création de la ressource, une erreur se produit sur l'absence d'un fichier XMI n'existant pas en dehors du fonctionnement interne de Xtext. Malgré de nombreuse tentatives de résolution incluant des changements d'environnement, de contexte, d'IDE, des modifications et ajout de dépendances le problème n'a pas été résolu. Mais pour pouvoir continuer l'implémentation, il fallait trouver une alternative et pour le moment, il s'agit d'une instanciation manuelle d'un fichier secDFD effectué avec le code généré par EMF à partir du méta-modèle EDFD représentant un secDFD.

Enfin pour découvrir les fonctionnalités d'Epsilon, je me suis servi d'un environnement de test en ligne afin d'appréhender les premières notions. Et pour l'instant, bien que je n'ai pas encore réalisé l'implémentation des fonctionnalités d'Epsilon dans le cadre du projet, il est possible d'en décrire les étapes. Cela commence par charger les instances de modèle EMF dans un module Epsilon. Pour cela, il faut enregistrer les instances de modèles EMF au format XMI, puis charger les modèles Ecore dans le module Epsilon pour pouvoir ensuite lire les instances sous format XMI. Ensuite il suffit de lancer sur ce même module les scripts ECL et EML afin de les exécuter et récupérer le modèle fusionné.

5.5.3. Perspectives

Concernant la suite du stage, il s'agit d'abord de finir l'implémentation de l'architecture présentée précédemment. Une fois que cette implémentation permettra une connexion entre le modèle et le code satisfaisante, on pourra alors y ajouter la gestion des événements de modifications de l'instance EMF du secDFD. Ces événements déclencheraient une comparaison des associassions faites entre le secDFD et le code avant et après modifications pour pouvoir notifier l'emplacement dans le code du projet à vérifier ou modifier pour correspondre aux modifications faites dans le but de maintenir le niveau de sécurité établit et évalué sur les secDFD. Par la suite, on pourra continuer en implémentant la détection de modification du côté du code du projet toujours dans un objectif de cohérence. Et pour aller plus loin, plusieurs améliorations pourront être apportées au niveau de l'établissement des connexions, comme l'introduction de nouvelles règles de connexions pour améliorer la précision notamment au niveau des "Assets" ou l'introduction de solution multi-threads comme le permettent Epsilon et Spoon pour améliorer la rapidité d'exécution.

Dans le cadre du sujet de thèse, ce stage constitue une première piste d'exploration, ici l'évaluation du niveau de sécurité se fait au travers de la modélisation secDFD mais ce n'est pas la seule modélisation qui intègre des critères pour l'évaluation d'un niveau de sécurité, je pense par exemple à UMLsec, BPMNsec qui sont aussi des représentations d'un système intégrant des annotations et des contrats de sécurité. On peut donc naturellement penser à un élargissement de cette méthode de connexion et de cohérence semi-automatique entre système et modèle à ces modélisations.

6. Analyse critique

6.1. Mise en application des compétences de la formation et apprentissages

Ce stage représente pour moi une première expérience dans le milieu de la recherche. Au-delà de l'adaptation à un nouvel environnement professionnel, il a également constitué une opportunité pour appliquer et développer les compétences acquises tout au long de ma formation à l'IMT Atlantique, ainsi que celles cultivées lors de mes expériences passées.

L'une des premières compétences que j'ai mobilisées lors de ce stage est la lecture d'articles scientifiques, un pilier fondamental dans le domaine de la recherche, mais également dans la pratique quotidienne d'un ingénieur. En effet, c'est par la lecture d'articles que l'on peut se tenir au courant des dernières avancées technologiques et théoriques. Au cours de ma formation, notamment dans le cadre du parcours ILSD, nous avons abordé cette méthodologie à travers des exercices basés sur la méthode de la triple lecture, une approche structurée qui permet d'appréhender un article scientifique de manière progressive. La première lecture sert à identifier les grandes lignes et les points essentiels, tandis que les lectures suivantes plongent plus en détail dans les aspects techniques et méthodologiques. Cette méthode m'a été extrêmement utile durant mon stage pour traiter efficacement une bonne quantité de documents académiques.

De plus, en travaillant aux côtés des doctorants du département informatique, j'ai enrichi cette compétence par l'apprentissage de nouvelles techniques de gestion des connaissances. Il s'agit notamment des méthodes de prise de notes structurées et de l'organisation des articles par thématiques et par tags pour pouvoir se constituer une bibliographie organisée de manière rapide et efficace.

Sur le plan du développement logiciel, ce stage a été l'occasion d'appliquer les bonnes pratiques que j'ai apprises dans les modules de formation DCL (Développement Collaboratif et Logiciel) et ILSD (Ingénierie Logicielle et Systèmes Distribués). Ces compétences couvrent l'ensemble du cycle de vie d'un

projet logiciel, depuis la phase de conception jusqu'à l'implémentation. Durant ce stage, j'ai mis en œuvre une démarche structurée qui a commencé par la modélisation de l'architecture du projet, une étape clé pour assurer la cohérence et la maintenabilité de l'application que je devais développer.

Parmi ces bonnes pratiques, j'ai adopté une approche modulaire et incrémentale dans l'implémentation des fonctionnalités, ce qui a facilité les phases de test, d'intégration, et d'évolution du projet. Cette méthode a non seulement permis de réduire les erreurs en phase de développement, mais elle a également augmenté la réutilisabilité du code, un atout essentiel dans un contexte de recherche où les outils évoluent rapidement et où des adaptations sont souvent nécessaires.

Un autre aspect crucial que j'ai eu l'occasion de développer pendant mon stage est la capacité à synthétiser et à restituer les informations et les avancées obtenues. Dans le milieu de la recherche, savoir présenter ses résultats de manière claire et concise est tout aussi important que la qualité des travaux réalisés. Cela implique non seulement la rédaction de rapports ou d'articles, mais aussi la présentation orale des concepts et des découvertes à un public varié, incluant parfois des non-spécialistes. Cette capacité de communication scientifique est fondamentale, car elle permet de faire progresser la recherche en partageant les connaissances acquises. De plus, elle est directement applicable dans d'autres contextes professionnels, où la capacité à transmettre des informations techniques de manière accessible est primordiale.

6.2. Contexte social et environnemental

La diversité est un élément clé dans le milieu de la recherche. Elle se traduit non seulement par la diversité des profils et des origines des chercheurs, mais aussi par la diversité des idées, des perspectives, et des méthodologies qui en résultent et les deux aspects sont indissociables. Au cours de ma formation et pendant mon stage, j'ai eu la chance de travailler au sein d'un environnement stimulant à IMT Atlantique, une institution où la diversité est non seulement encouragée, mais également intégrée au cœur de sa stratégie. IMT Atlantique, grâce à ses étudiants et chercheurs venus du monde entier, favorise des échanges culturels et académiques enrichissants, qui permettent de faire avancer la recherche de manière plus innovante et inclusive.

Concernant le sujet de mon stage maintenant, si le génie logiciel et la numérisation en général ont permis de nombreuses avancées technologiques et sociales, il est impossible de négliger leur impact environnemental croissant. Le secteur numérique est en effet responsable d'une part de plus en plus importante des émissions mondiales de CO2. En démontre la masse croissante de matériels connectés pour faire fonctionner les infrastructures numériques qui sont de gros consommateurs d'énergie. À ce titre, le développement de logiciels doit impérativement s'accompagner d'une réflexion sur leur efficience énergétique et sur les moyens de réduire leur empreinte carbone. Alors que de plus en plus de services sont dématérialisés et que la demande en puissance de calcul ne cesse d'augmenter, il est essentiel de se questionner sur la pertinence de certaines solutions technologiques. On peut alors penser à des recherches contribuant à développer des technologies moins consommatrices en se tournant vers des solutions plutôt low-tech accompagnées de recherches liées à l'optimisation de solutions existantes, voire même repenser certaines pratiques numériques pour en limiter l'expansion non contrôlée.

6.3. Evolution et plan de carrière future

Cette expérience dans le milieu de la recherche est pour moi une chance de découvrir un environnement qui m'a toujours intrigué. La possibilité d'explorer plus en profondeur des sujets complexes afin de mieux comprendre leurs subtiles mécanismes fut pour moi une des motivations à l'origine de la

volonté de découvrir ce milieu. Aussi cette expérience a confirmé mon souhait grandissant d'effectuer une thèse suite à mon parcours ingénieur. Cela est dû à de nombreuses raisons dont notamment ma curiosité par rapport à des sujets complexes, la liberté et l'autonomie associées aussi à plus de responsabilités, la qualité des échanges humains dans ce milieu et enfin du défi personnel. Concernant le choix du sujet de ma thèse, je ne pense pas continuer uniquement dans le domaine du génie logiciel bien que celui-ci m'intéresse beaucoup. En effet, je souhaiterais dans l'idéal pouvoir y intégrer des composantes qui me passionnes telles que les questions environnementales ou biologiques comme par exemple la bio-informatique. Pour conclure, ce stage aura confirmé ma volonté d'effectuer une thèse et en aura précisé la direction.

Références

[1] IMT Atlantique. Recherche et innovation : croiser approche fondamentale et applications. https://www.imt-atlantique.fr/fr/recherche-innovation, 2024. Last accessed 06 September 2024.

[2] Marco Brambilla, Jordi Cabot, and Manuel Wimmer. *MDSE Use Cases*, pages 25–42. Springer International Publishing, Cham, 2017.

[3] Tao Chen and Haiyan Suo. Design and practice of security architecture via devsecops technology. In *2022 IEEE 13th International Conference on Software Engineering and Service Science (ICSESS)*, pages 310–313, Oct 2022.

[4] Tom DeMarco. *Structured analysis and system specification*. Englewood Cliffs, N.J. : Prentice Hall, 1979.

[5] Eclipse. Epsilon documentation. https://eclipse.dev/epsilon/doc/, 2024. Last accessed 06 September 2024.

[6] Eclipse. Xtext documentation. https://eclipse.dev/Xtext/documentation/index.html, 2024. Last accessed 06 September 2024.

[7] Gradle. User manual. https://docs.gradle.org/current/userguide/userguide.html, 2024. Last accessed 06 September 2024.

[8] Robbert Jongeling, Johan Fredriksson, Federico Ciccozzi, Antonio Cicchetti, and Jan Carlson. *Towards Consistency Checking Between a System Model and Its Implementation*, pages 30–39. 10 2020.

[9] Djamel Eddine Khelladi, Benoit Combemale, Mathieu Acher, Olivier Barais, and Jean-Marc Jézéquel. Co-evolving code with evolving metamodels. In *2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE)*, pages 1496–1508, 2020.

[10] Lab-STICC. P4s. https://labsticc.fr/fr/equipes/p4s, 2024. Last accessed 06 September 2024.

[11] Maven. Wlecome page. https://maven.apache.org/index.html, 2024. Last accessed 06 September 2024.

[12] Per Håkon Meland and Jostein Jensen. Secure software design in practice. In *2008 Third International Conference on Availability, Reliability and Security*, pages 1164–1171, 2008.

[13] Renaud Pawlak, Martin Monperrus, Nicolas Petitprez, Carlos Noguera, and Lionel Seinturier. Spoon : A Library for Implementing Analyses and Transformations of Java Source Code. *Software : Practice and Experience*, 46 :1155–1179, 2015.

[14] Adam Shostack. *Threat Modeling : Designing for Security*. Indianapolis, IN : J. Wiley & Sons, 2014.

[15] Laurens Sion, Koen Yskout, Dimitri Van Landuyt, and Wouter Joosen. Solution-aware data flow diagrams for security threat modeling. In *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*, SAC '18, page 1425–1432, New York, NY, USA, 2018. Association for Computing Machinery.

[16] Spoon. Source code analysis and transformation for java. https://spoon.gforge.inria.fr/index.html, 2024. Last accessed 06 September 2024.

[17] Katja Tuma, Sven Peldszus, Daniel Strüber, Riccardo Scandariato, and Jan Jürjens. Checking security compliance between models and code. *Software and Systems Modeling*, 22(1) :273–296, Feb 2023.

[18] Katja Tuma, Riccardo Scandariato, and Musard Balliu. Flaws in flows : Unveiling design flaws via information flow analysis. In *Proceedings - 2019 IEEE International Conference on Software Architecture, ICSA 2019*, Proceedings - 2019 IEEE International Conference on Software Architecture, ICSA 2019, pages 191–200, United States, April 2019. Institute of Electrical and Electronics Engineers Inc. Funding Information : This research was partially supported by the Swedish VIN-NOVA FFI project "HoliSec : Holistic Approach to Improve Data Security". Musard Balliu was partially supported

by the JointForce project financed by the Swedish Research Council and the TrustFull project financed by the Swedish Foundation for Strategic Research. Publisher Copyright : © 2019 IEEE. Copyright : Copyright 2019 Elsevier B.V., All rights reserved. ; 2019 IEEE International Conference on Software Architecture, ICSA 2019 ; Conference date : 25-03-2019 Through 29-03-2019.

[19] Katja Tuma, Laurens Sion, Riccardo Scandariato, and Koen Yskout. Automating the early detection of security design flaws. In *Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems*, MODELS '20, page 332–342, New York, NY, USA, 2020. Association for Computing Machinery.

[20] Edward Yourdon and Larry L. Constantine. *Structured design : fundamentals of a discipline of computer program and systems design*. Englewood Cliffs, N.J. : Prentice Hall, 1979.

OUR WORLDWIDE PARTNERS UNIVERSITIES - DOUBLE DEGREE AGREEMENTS

3 CAMPUS

IMT Atlantique Bretagne–Pays de la Loire – **http://www.imt-atlantique.fr/**

Campus de Brest
Technopôle Brest-Iroise
CS 83818
29238 Brest Cedex 3
France
T +33 (0)2 29 00 11 11
F +33 (0)2 29 00 10 00

Campus de Nantes
4, rue Alfred Kastler
CS 20722
44307 Nantes Cedex 3
France
T +33 (0)2 51 85 81 00
F +33 (0)2 99 12 70 08

Campus de Rennes
2, rue de la Châtaigneraie
CS 17607
35576 Cesson Sévigné Cedex
France
T +33 (0)2 99 12 70 00
F +33 (0)2 51 85 81 99

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom