25-27/03/2021

Projet de Thèse

Par MASSAGA NGONDA Aristide Marie

Thème : Une approche d'apprentissage automatique pour la vérification des architectures micro-services.

CONTEXTE :

Les styles architecturaux tels que microservice, SOA, MVC, modèle en couches, etc., constituent des approches standardisées et réutilisables pour des problèmes architecturaux récurrents. Ils comprennent un vocabulaire d'éléments conceptuels (composants et connecteurs), impose des règles de configuration entre les éléments du vocabulaire (ensemble de contraintes) et véhicule une sémantique qui donne un sens (non ambiguë) [1]. Pour tout projet logiciel, ils constituent un moyen important de standardisation et de réutilisation des connaissances. Toutefois, une fois le système conçu, aussi bien au moment de son implémentation que lorsqu'il évolue, son architecture tend à s'écarter de la conception prévue, ce qui entraîne une perte de réutilisabilité et de standardisation, d'où le besoin d'outils d'aide à la vérification de la conformité du système avec les règles du style architectural.

De nombreux chercheurs se sont attaqués à ce problème et ont proposé des outils notamment *Weinreich et al* [2], qui l'ont abordé pour les architectures SOA. Dans un contexte marqué par l'adoption massive du style microservice par l'industrie du logiciel, il devient vital de disposer d'outils d'aide à la vérification de la conformité du système, qui tient compte des spécificités de ce style architectural.

PROBLÉMATIQUE :

Avec le développement du cloud computing, on assiste à une conception et migration des systèmes informatiques vers le style architectural microservice. Parmi les raisons de ce changement de direction de l'industrie, nous avons qu'il est parfaitement adapté aux besoins liés à l'adoption des méthodes de développement agile pour la gestion des projets logiciels comme SCRUM ou XP, où la gestion des changements rapide est primordiale. Toutefois, ce style apporte une complexité supplémentaire non-négligeable liés notamment au caractère très distribué et à la petite taille des services, qui sont deux de ses plus importantes spécificités.

Les systèmes logiciels ont une tendance naturelle à devenir plus complexes et donc moins faciles à maintenir correctement structuré au fil du temps : on parle d'érosion de l'architecture [4]. En outre, les caractéristiques de petite taille et de distribution des services, ont pour conséquences un accroissement de la complexité à maintenir la conformité architectural du système et à conserver les facteurs de réutilisabilité et de standardisation liés au style.

Le projet de recherche auquel nous nous intéressons dans le cadre de cette thèse, s'inscrit dans cette problématique, en posant cette fois la question : Comment vérifier la conformité architecturale des systèmes à base de microservice aux spécificités de réutilisabilité et de standardisation du style architectural par un processus basé sur l'apprentissage automatique ?

MÉTHODE DE RECHERCHE :

Les styles architectures sont définies sur la base de règles constituées de rôles (composants et connecteurs) et de contraintes sur les rôles et les relations entre rôles [2]. Pour répondre à la question de recherche énoncé précédemment, nous avons adopté un protocole en 4 étapes :

- Définition des règles et des rôles spécifiques au style architectural : A. MASSAGA [1], recense 27 patrons architecturaux applicables lors de la réalisation d'un système à base de microservices. Dans le cadre de ce travail nous, nous limiterons à l'extraction des règles et rôles de 3 de ces patrons qui interviennent systématiquement dans tout système à base de microservice : (1) Service discovery : service registry pattern, (2) Data consistency : SAGA pattern, (3) Query implementation : CQRS pattern.
- Extraction du modèle architectural d'un système informatique existant : Nous utiliserons une approche permettant d'extraire et d'analyser l'architecture d'un système logiciel basé sur des microservices en combinant des informations statiques sur les services et des informations agrégées sur l'infrastructure [5].
- Correspondance entre les rôles de l'architecture de référence et ceux du modèle extrait : Ici nous utiliserons une approche basée sur l'intelligence artificielle (celle-ci restant à clairement définir) permettant d'assigner aux composants issus de l'extraction de l'architecture un rôle précis pris d'ensemble des rôles précédemment identifiés pour le style architectural ;
- Évaluation de la conformité des règles.

Nous avons recensé, 5 applications open source construites dans le style microservice et compatibles avec nos travaux. Nous nous appuierons sur elles pour la réalisation de chaque étape de la recherche: TeaStore, ACME Air, Spring Cloud Demo, Sock Shop, MusicStore [7].

CALENDRIER ET ÉTAPE :

Pour ce projet de thèse, nous avons choisi une méthode de gestion agile, c'est-à-dire de façon incrémentale et itérative. Pour cela, nous avons diviser le projet en 5 phases comme le montre la figure 1.

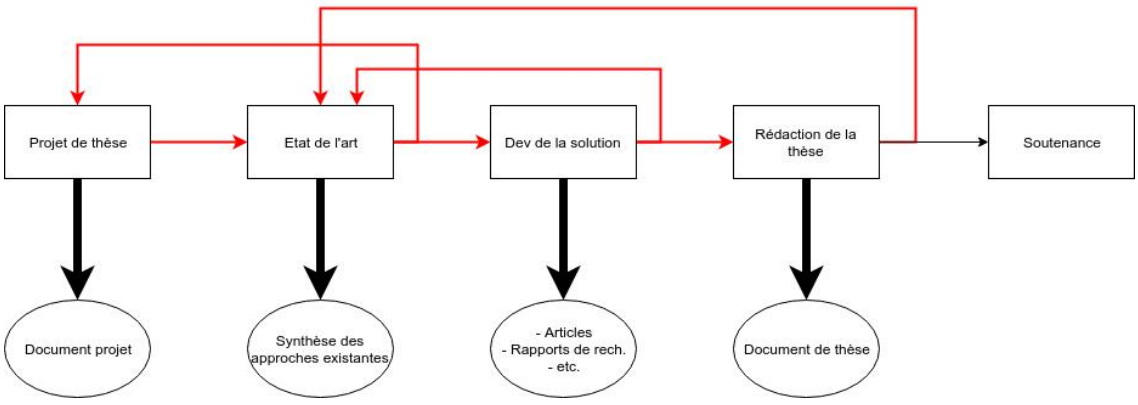

Figure 1 : Cycle du projet de thèse

De chaque étape, certains livrables sont attendus :

- ➢ Projet de thèse : document projet;
- ➢ L'état de l'art : synthèses des approches existantes et positionnement par rapport à l'existant ;
- ➢ Développement de la solution : De cette phase il ressort des articles, rapports de thèse, etc. Il s'agit de la réalisation des 4 étapes citées dans la méthodologie.

- ➢ Rédaction de la thèse : il en ressort le document de thèse
- ➢ La soutenance : grade de docteur

La durée totale du projet est de 38 mois et s'organise comme suit :

Phase 1 : Définition des règles et des rôles spécifiques au style architectural (Durée 4*2 mois : Mars 21 - Octobre 21)

- État de l'art
- Définition des règles et des rôles spécifiques au style architectural
- Publication d'un article (si cela en donne lieu)
- Réalisation d'un rapport d'avancement
- Rédaction du document de thèse

Phase 2 : Extraction du modèle architectural d'un système informatique existant (Durée 8 mois : Novembre 21 - Juin 22)

- État de l'art
- Extraction du modèle architectural d'un système informatique existant
- Publication d'un article (si cela en donne lieu)
- Réalisation d'un rapport d'avancement
- Rédaction du document de thèse

Phase 3 : Correspondance entre les rôles de l'architecture de référence et ceux du modèle extrait (Durée 8 mois : Juillet 22 - Février 23)

- État de l'art
- Correspondance entre les rôles de l'architecture de référence et ceux du modèle extrait
- Publication d'un article (si cela en donne lieu)
- Réalisation d'un rapport d'avancement
- Rédaction du document de thèse

Phase 4 : Évaluation de la conformité des règles (Durée 8 mois : Avril 23 - Novembre 23)

- État de l'art
- Évaluation de la conformité des règles
- Publication d'un article
- Réalisation d'un rapport d'avancement
- Rédaction du document de thèse

Phase 5 : Soutenance(Durée 6 mois : Décembre 23 - Mai 24)

- Consolidation et finalisation du document de thèse
- Procédure administrative nécessaire à la soutenance
- Soutenance

RÉFÉRENCES BIBLIOGRAPHIQUES :

[1] A. MASSAGA « Évaluation de la mise en oeuvre des architectures micro-services : Application aux plateformes Spring Boot et Java EE ». Mémoire de master 2. Université de Yaoundé 1. Faculté des Sciences, Août 2020.

[2] Weinreich, Rainer, et Georg Buchgeher. « Automatic Reference Architecture Conformance Checking for SOA-Based Software Systems ». In 2014 IEEE/IFIP Conference on Software Architecture, 95-104. Sydney, Australia: IEEE, 2014. https://doi.org/10.1109/WICSA.2014.22.

[3] A. Jansen and J. Bosch. Software architecture as a set of architectural design decisions. In Software Architecture, 2005. WICSA 2005. 5th Working IEEE/IFIP Conference on, pages 109–120, Nov. 2005. doi: 10.1109/WICSA.2005.61

[4] D. E. Perry and A. L. Wolf. Foundations for the study of software architecture. ACM SIGSOFT Software Engineering Notes, 17(4):40–52, Oct. 1992. URL http://www.bell-labs.com/user/dep/work/papers/swa-sen.ps.

[5] Mayer, Benjamin, et Rainer Weinreich. « An Approach to Extract the Architecture of Microservice-Based Software Systems ». In 2018 IEEE Symposium on Service-Oriented System Engineering (SOSE), 21-30. Bamberg: IEEE, 2018. https://doi.org/10.1109/SOSE.2018.00012.

[6] Chihada, Abdullah & Jalili, Saeed & Hasheminejad, Seyed Mohammad Hossein & Zangooei, Mohammad Hossein. (2014). Source code and design conformance, design pattern detection from source code by classification approach. Applied Soft Computing. 10.1016/j.asoc.2014.10.027.

[7] Kistowski, Jóakim v., Simon Eismann, Norbert Schmitt, André Bauer, Johannes Grohmann, et Samuel Kounev. « TeaStore: A Micro-Service Reference Application for Benchmarking, Modeling and Resource Management Research ». University of Würzburg, s. d., 14.

[8] « Qu'est-ce que l'architecture Microservices ? | SUPINFO, École Supérieure d'Informatique ».Consulté le 13 février 2019. https://www.supinfo.com/articles/single/5676-qu-est-ce-que-architecture-microservices.

[9] Richardson, Chris. Microservices Patterns. Manning Publications Co. S HELTER ISLAND, 2019. Livre.

[10]microservices.io. « Microservices Pattern: Microservice Architecture pattern ». http://microservices.io/patterns/microservices.html.