Faciliter la migration vers une architecture microservices

Approche basée sur la rétroingénierie dirigée par les modèles et l'apprentissage par renforcement.

MohammadHadi Dehghani[1], Shekoufeh Kolahdouz-Rahimi[1], Massimo Tisi[2], Dalila Tamzalit[3]

1. Isfahan University of Technology, Iran
mohammadhadi.dehghani@jku.at, sh.rahimi@eng.ui.ac.ir,
2. IMT-Atlantique, CNRS, LS2N, F-44000 Nantes, France
massimo.tisi@imt-atlantique.fr,
3. Nantes Université, CNRS, LS2N, F-44000 Nantes, France
dalila.tamzalit@univ-nantes.fr

RESUME. Le présent document synthétise l'article accepté dans Software and Systems Modeling, Dehghani, M., Kolahdouz-Rahimi, S., Tisi, M., & Tamzalit, D. (2022). Facilitating the migration to the microservice architecture via model-driven reverse engineering and reinforcement learning, et présenté à Models 2022 dans First-Journal Track.

Mots-clés : Architecture de microservices – Apprentissage par renforcement – Rétroingénierie dirigée par les modèles – Migration.

1. Introduction et problématique

Les microservices (MS) sont un style architectural de développement d'applications (Fowler, 2016). De par leurs caractères distribués et faiblement couplés, ils permettent une grande souplesse dans le développement logiciel, les évolutions et le déploiement. Les microservices permettent donc aux entreprises une grande réactivité face aux besoins changeants, augmentant leur compétitivité. Pour cette raison, plusieurs organisations entreprennent le coûteux processus de migration de leurs architectures logicielles traditionnelles vers des microservices (MS). Cependant, la migration est souvent faite de manière manuelle, notamment de par la complexité à déterminer la bonne granularité des MS et la difficulté à identifier l'affectation des fonctions métier au sein des MS (Gouigoux et Tamzalit. 2017).

2. Proposition

L'approche proposée offre une assistance semi-automatisée aux architectes logiciels pour faire face à la migration vers des MS. Elle suit trois étapes clés :

1. Extraire du code source existant son modèle orienté-objet grâce à MoDisco, un outil de rétroingénierie dirigé par les modèles (Bruneliere *et al*. 2014).

2. Extraire le modèle à base de MS en analysant les modèles entité-association et de cas d'utilisation du système existant avec Service Cutter (Gysel. *et al.* 2016).

3. **Contribution** : automatiser l'affectation des méthodes existantes de l'étape (1) aux MS identifiés à l'étape (2). Cette affectation est un problème complexe et multifactoriel. L'approche est basée sur le Deep Q-Learning par renforcement (Reinforcement Learning), une technique d'apprentissage d'IA. Elle génère des épisodes d'entraînement et d'apprentissage cherchant la meilleure correspondance possible entre les méthodes et les MS, basée sur des réseaux de neurones. L'applicabilité et la correction de l'approche développée ont été évaluées sur des cas d'utilisation et en exécutant des tests de performance.

3. Conclusion et perspectives

La solution développée est basée sur une technique d'IA pour approcher la meilleure correspondance possible entre les méthodes et les MS, afin de remodulariser les systèmes logiciels en vue d'une migration vers une architecture MS. Dans le futur, nous souhaiterions améliorer l'identification de la granularité des microservices en prenant mieux en compte la dimension métier, notamment avec la notion de contextes bornés (*bounded context*) du Domain-Driven Design (Evans 2003). Un contexte borné spécifie un périmètre métier cohérent et unifié pour chaque service. Cette notion est fondamentale pour identifier la bonne granularité d'un microservice car elle représente un sous-système aligné sur une partie du domaine métier.

Principales références

Bruneliere H, Cabot, J., Dupé, G., Madiot (2014), F.: Modisco: A model driven reverse engineering framework. *Information and Software Technology 56(8), 1012–1032.*

Evans (2003) Domain-Driven Design: Tackling Complexity in the Heart of Software. *Addison-Wesley*

Fowler (2016), Production-ready microservices: Building standardized systems across an engineering organization. *O'Reilly Media, Inc."*

Gouigoux et Tamzalit (2017) From monolith to microservices: Lessons learned on an industrial migration to a web oriented architecture. *IEEE International Conference on Software Architecture Workshops*

Gysel, Kölbener, Giersche, Zimmermann (2016): Service cutter: A systematic approach to service decomposition. *European Conference on Service-Oriented and Cloud Computing*.

Sutton et Barto (2018) A.: Reinforcement learning: an introduction. *The MIT Press.*