IFT4055 - Projet Honor

MQL à Cypher – Un Framework pour Traduire un langage de requêtes spécifique au domaine pour interroger un projet versionnable en des expressions Cypher

Philippe Gabriel - 20120600

18 décembre 2022

Problème

Les systèmes de versions de contrôle (*Version Control System* – VCS) présentent une grande importance dans les besoins de vouloir gérer le versionnement de collections de données. Dans le domaine de l'informatique et de l'ingénierie logicielle, des systèmes comme Git représentent une très grande importance dans la pratique menant à l'intérêt industrielle du développement de grands logiciels. L'ingénierie dirigée par les modèles (*Model-Driven Engineering* – MDE) est une approche présentant plusieurs avantages dans la conception de logiciels. Cette approche permet au concepteur d'abstraire des détails des langages de programmation générals, de sorte à concentrer ses efforts dans la résolution du problème. L'usage de modèles ou de langages spécifiques au domaine (*Domain-Specific Languages* – DSL) aide l'expert à s'exprimer dans une terminologie propre à son domaine d'étude et lui permettrait de contribuer à la conception d'un logiciel sans avoir de connaissances de programmation. Un intérêt industrielle commence à se manifester pour la MDE poussant à l'adopter pour concevoir de larges projets. Un VCS orienté-ligne comme Git utilisé pour gérer le versionnement des changements apporter à un projet MDE n'est pas la meilleure approche à adopter gérer de gros projets. L'interprétation des changements sur un modèle conçu par l'expert du domaine sera au niveau de la sérialisation du modèle. Cette interprétation n'est donc pas spécifique au domaine du modèle. La sémantique riche du domaine est complètement négligé. Un VCS spécifique au domaine (*Domain-Specific VCS* – DSVCS) serait plus approprié pour des projets MDE. DSMCompare [1] permet de comprendre les différences entre modèles en terme de la sémantique d'un langage de modélisation. Il s'agit d'une approche considérant la syntaxe abstraite et concrète d'une DSL pour exprimer les différences entre modèles. Ce niveau de détection de changements est une composante cruciale dans un DSVCS. Un DSVCS devrait adhérer à différents critères tel que la définition d'une unité versionnable pour permettre la comparaison sémantique, les commandes offertes pour manipuler le projet, la possibilité d'interroger le VCS selon le projet, la méthode de stockage, etc... [2]. Pour ce projet, je présente une approche MDE basé sur la traduction d'expressions MQL – un langage de requêtes spécifique au domain (*Domain-Specific Query Language* – DSQL) pour interroger un projet versionnable – en des expressions Cypher – un langage de requêtes supporté par des bases de données de graphes Neo4j. Ce projet vient essentiellement proposer une solution quant à la composante d'interroger le VCS.

Méthodes utilisées

La solution proposée est basée sur la MDE. Sa presque totalité a donc été rédigée dans le Eclipse Modeling Framework (EMF). Je présente, dans les sous-sections ci-dessus, les différentes étapes de la solution.

Projet Versionnable

Afin d'interroger le VCS sur un projet versionnable, il est nécessaire de déterminer à priori les unités versionnables d'un projet. Cela est décrit dans le métamodèle d'un projet versionnable présenté à la Figure 1. Un projet versionnable a une structure assez similaire à celle des projets utilisant un VCS linéaire. Un projet consiste en des branches. Les branches contiennent des commits définissant un point où l'on désire enregistrer des changements. Les modèles constituent ici l'unité dont l'expert du domaine modifiera dans le projet MDE. Chaque modèle possède une durée de vie exprimée sous forme d'une série de snapshots.

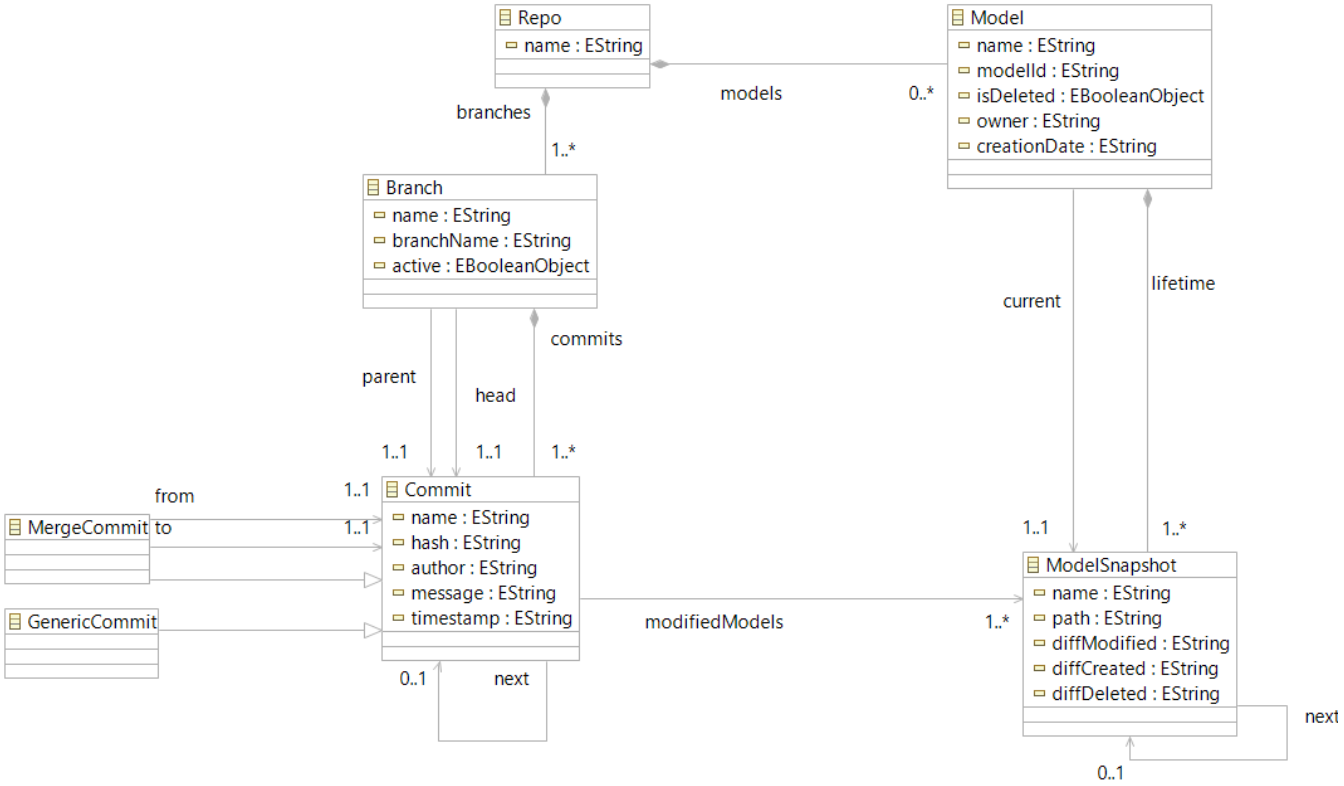

Figure 1: Métamodèle d'un Projet Versionnable

Model Query Language - MQL

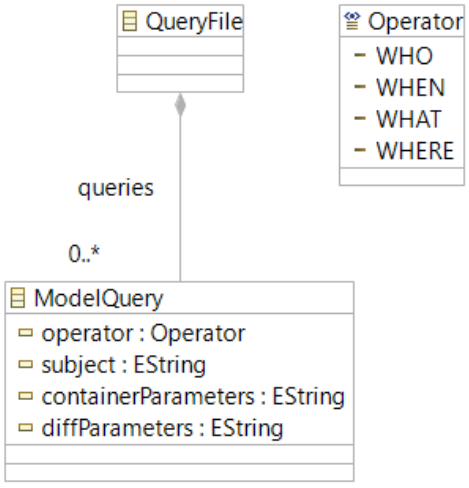

Figure 2: Métamodèle de MQL

La syntaxe abstraite du langage MQL est exprimé dans le métamodèle de MQL à la Figure 2. Une expression MQL est constituée de plusieurs requêtes. Chaque requête est composée d'un opérateur, du sujet de requête, et de paramètres pour mieux filtrer les résultats attendues. La syntaxe concrète de ce langage a été définie à l'aide de Xtext. Le Listing 1 démontre un exemple

d'expression MQL. Dans cette expression, on remarque que chaque requête est séparée par une virgule et que la dernière requête se termine par un point d'interrogation. Chaque requête commence par un opérateur. La seconde requête utilise l'opérateur `DESCRIPTION` qui s'agit ici tout simplement de sucre syntaxique pour l'opérateur `WHERE` qui a été jugé ne pas être très intuitif dans ce contexte.

```
1   WHO head {
2     branchName = "main",
3     active = "true"
4   },
5   DESCRIPTION parent {
6     branchName = "b1"
7   },
8   WHEN created {
9     timestamp < "2022-09-04"
10  } [
11    "MyDomainSpecificObject.x = 19"
12  ]?
```

Listing 1: Expression MQL

Un parseur Xtend a été implémenté pour produire un modèle MQL à partir d'une expression MQL. Ce dernier génère un modèle MQL conforme au métamodèle MQL lorsque l'expert du domaine sauvegarde le fichier où il rédigeait ses requêtes MQL.

Transformations Modèle-à-texte

À partir d'un modèle d'un projet versionnable conforme à son métamodèle.

Automatisation

Résultats obtenus

Améliorations futures

Références

[1] Manouchehr Zadahmad Jafarlou, Eugene Syriani, Omar Alam, Esther Guerra, and Juan de Lara. Dsmcompare: domain-specific model differencing for graphical domain-specific languages. *Software and Systems Modeling*, 21(5):2067–2096, October 2022.

[2] Eugene Syriani and Manuel Wimmer. A roadmap towards domain-specific version control systems. *White Paper*, September 2018.