

Do Developers Refactor Data Access Code? An Empirical Study

Biruk Asmare Muse, Foutse Khomh,
Giuliano Antoniol

2022 IEEE International Conference on Software Analysis, Evolution and
Reengineering (SANER)

発表者：松尾 和輝

背景

- デジタル機器や人間が生み出す膨大なデータを処理するために、データ集約型システム（データベースに頻繁にアクセスするようなシステム）が必要となっている
- データ集約型システムの開発には、データの保存、処理、学習のための多数の専門フレームワークの統合が必要となるため、特有の技術的負債やコードスメルが混入しやすい
- 技術的負債やコードスメルはリファクタリングによって返済できるが、データ集約型システムにおけるリファクタリングは調査が行われていない

目的とアプローチ

- 目的

- DA(データアクセス)クラスにおけるリファクタリング活動を調査すること
 - DAクラス：データ集約型システムで、ドライバ関数やAPIを呼び出してデータベースや他のシステムと直接やり取りするクラス

- アプローチ

- オープンソースのデータ集約型システムに関するケーススタディを実施

リサーチクエスチョン

- RQ1：DAクラスにおけるリファクタリングはどの程度普及しているのか？
- RQ2：対象システムのライフタイムにおいて、リファクタリング活動はどの程度変化するのか？
- RQ3：DAリファクタリングの対象となるコード要素は、何を実装しているのか？
- RQ4：DAリファクタリングは、SQLクエリやSQLコードスメルに影響を与えるか？

対象リポジトリ

- SQLコードスメルの普及と影響を研究したMuseら[4]の研究で特定された**150**の**GitHub**リポジトリから、SQLクエリ数の多い上位**12**リポジトリを選択

Project Name	Number of commits	Number of Queries	Number of data access files	Number of refactoring
Eclipse-ee4j/eclipselink	10403	1371	43	80, 311
Adempiere/adempiere	15754	941	365	2,104,700
Appirio-tech/direct-app	3073	876	95	1492
DotCMS/core	17957	740	40	60, 211
Wso2/carbon-apimgt	33174	656	12	32,101
Oltpbenchmark/oltpbench	1110	303	131	2396
Mtotschnig/MyExpenses	9065	287	3	9949
Querydsl/querydsl	7874	249	23	38,065
Wordpress-mobile/WordPress-Android	59048	221	17	31, 647
AppLozic/Applozic-Android-SDK	2298	202	6	2136
Xipki/xipki	6328	193	21	88, 623
Deegree/deegree3	8692	190	45	21, 459

[4] B. A. Muse, M. M. Rahman, C. Nagy, A. Cleve, F. Khomh, and G. Antoniol, "On the prevalence, impact, and evolution of sql code smells in data-intensive systems," in Proceedings of the 17th International Conference on Mining Software Repositories, 2020, pp. 327–338.

データセット

- 対象リポジトリからRefactoringMiner[19]を用いて、リファクタリングインスタンスを収集
 - 174,776コミットから2,473,090件を収集
- DAリファクタリングインスタンスの特定
 - SQLに関連するインポート文を含む
 - DAクラスが少なくとも一つのSQLクエリに関連する
 - 以上の基準で2,473,090件のリファクタリングインスタンスから18,892件のDAリファクタリングインスタンスを収集
- 通常リファクタリング2,454,198件とそれが適用された通常クラス81,857件、DAリファクタリング18,892件とそれが適用されたDAクラス1,318件をデータセットとする

RQ1：アプローチ

RQ1：DAクラスにおけるリファクタリングはどの程度普及しているか？

- 通常クラスとDAクラスのリファクタリングを比較
 - 1クラスあたりのリファクタリングインスタンスの数
 - リファクタリングの密度を表したバイオリンプロット
 - 以下の式で定義

$$\text{Refactoring density} = \frac{\text{no of refactorings}}{\sqrt{\text{average code size}}}$$

- no of refactorings：適用されたリファクタリングの数
- average code size：ターゲットクラス全体のリビジョンのソフトウェア行数の平均値
- 特定のタイプのリファクタリング

RQ1：結果

- クラス1件あたりのリファクタリング数の比較
 - 通常クラスは29.98件，DAクラスは14.33件
 - DAリファクタリングはあまり浸透していない
- リファクタリングの密度を表したバイオリンプロット



- DAクラスのリファクタリング密度は通常クラスのそれよりわずかに低い
 - DAリファクタリングは，少数のコアクラスに集中している

RQ1：結果

- DAクラスのリファクタリングはRename Variableが多いが、通常クラスはMove Attributeが多かった

Data access class			Regular class		
Refactoring Type	count	percentage	Refactoring Type	count	percentage
Rename Variable	1880	9.951	Move Attribute	1470365	59.912
Change Variable Type	1539	8.146	Change Attribute Access Modifier	600345	24.462
Add Parameter	1119	5.923	Add Method Annotation	26913	1.097
Add Parameter Modifier	998	5.283	Rename Method	22963	0.936
Change Parameter Type	910	4.817	Change Parameter Type	21618	0.881
Rename Parameter	888	4.7	Change Variable Type	21372	0.871
Extract Method	887	4.695	Change Return Type	19459	0.793
Add Method Annotation	816	4.319	Add Parameter	18528	0.755
Rename Method	705	3.732	Rename Variable	18345	0.747
Change Method Access Modifier	620	3.282	Rename Parameter	16107	0.656

RQ2：アプローチ

RQ2：対象システムのライフタイムにおいて、リファクタリング活動はどのように変化するか？

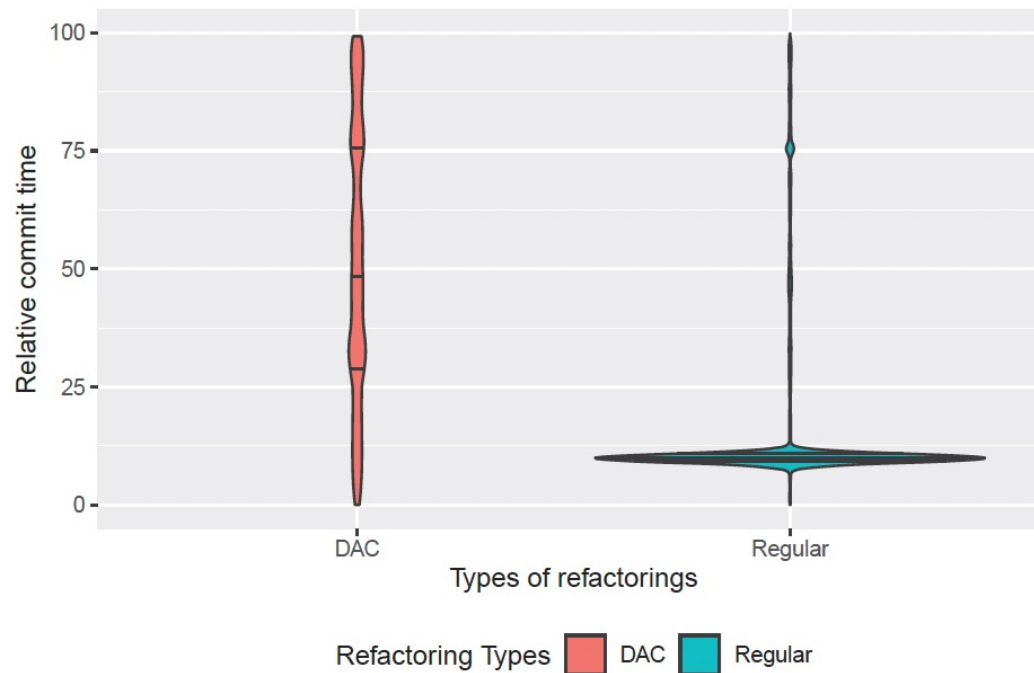
- 通常クラスのリファクタリングと、データアクセスクラスのリファクタリングで相対コミット時間を計測し比較
 - 相対コミット時間：以下の式で定義

$$RelativeCommitTime(\%) = \frac{distance * 100}{TotalCommits}$$

- distance：リファクタリング時に対象システムが持つコミット数
- TotalCommits：実験時に対象システムが持つコミット数

RQ2：結果

- 通常リファクタリングは開発初期に行われるが、**DA**リファクタリングは時期にかかわらず行われる



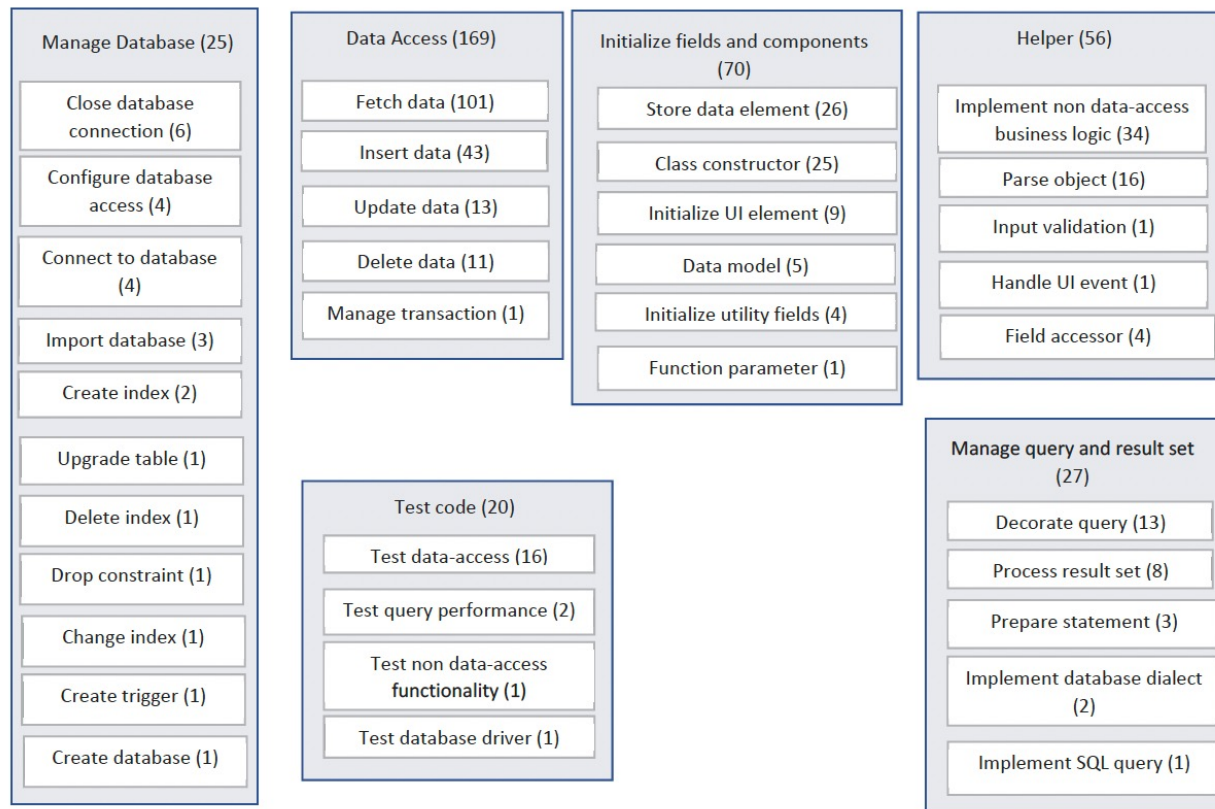
RQ3：アプローチ

RQ3：DAリファクタリングの対象となるコード要素は、何を実装しているのか？

- DAリファクタリングインスタンスをランダムに抽出
 - 378件のサンプルを収集
- 収集したサンプルをリファクタリングタイプで分類
 - 偽陽性である10件のサンプルと、分類できなかった2件のサンプルを除いた366件を分類

RQ3：結果

- データアクセス機能に関するリファクタリングが多い



RQ4：アプローチ

- 対象システムに対して SQLInspect [20] を実行し，SQLクエリとSQLコードスメルを抽出
- DAリファクタリングインスタンスと，SQLクエリ/SQLコードスメルを行レベルおよびメソッドレベルで紐付ける
 - メソッドレベル
 - リファクタリングが行われたメソッドに，SQLクエリ/SQLコードスメルが含まれているか
 - 行レベル
 - SQLクエリやSQLコードスメルの行番号がリファクタリングに関与したコードの行番号と一致しているか

RQ4：結果

- DAリファクタリングインスタンスのうち、SQLクエリやSQLコードスメルを扱ったものはごく一部である
- 行レベルでは、SQLクエリに関与しているリファクタリングインスタンスは0.45%のみであった。
- リファクタリングに関連するSQLコードスメルのインスタンスはなかった
- メソッドレベルでは、SQLクエリに関連するリファクタリングが29.68%，SQLコードスメルに関連するリファクタリングが1.35%であった

まとめ

- 目的
 - DAクラスにおけるリファクタリング活動を調査すること
- アプローチ
 - オープンソースのデータ集約型システムに関するケーススタディを実施
- 結果
 - DAクラスと通常のクラスで、異なるタイプのリファクタリングインスタンスが広く普及している
 - 通常クラスでは、初期にリファクタリングが行われるが、DAクラスでは、時期に関わらずリファクタリングが行われる
 - DAリファクタリングは一般的にSQLクエリとSQLコードスメルを扱わない

所感

- DAリファクタリングが時期にかかわらず行われるのはなぜ？
 - 大量のデータを扱うから、バグが起こらないようにするため？
- データアクセス機能に関するリファクタリングが多いのはなぜ？
 - 同上
- DAリファクタリングの数が少ないのはなぜ？
 - 実装時にリファクタリングが必要ないくらいきっちりコードを書いている？