概念モデル再利用のための欠陥検知手法†

飯山 隆章*

Defect Detection Method for Conceptual Model Reuse

Takaaki Iiyama

1 はじめに

概念モデルとはシステムの設計を表現したものである．本研究では，システムの静的構造を表現するクラス図を扱う．クラス図は図 1 が示すようにクラス，フィールド，メソッドで構造，関連で構造間の相互関係を表現する．

今日のソフトウェアは大規模化，複雑化している．これは，クライアントの要望が複雑かつ肥大化していることが要因である．このような大規模かつ複雑化したシステムを 1 から設計すると膨大な時間が必要である．そのため，ソフトウェアパターンを再利用して解決する．ソフトウェアパターンとは，開発過程において頻出する特定の問題に対する解答を表現したものである．しかしながら，ソフトウェアパターンの 1 種のデザインパターン[2][3]は 23 個の問題しか対応ができない．また，ソフトウェアパターンは適応する場面に応じて構造や動作を変更が必要である．ソフトウェアパターンで解決できない問題が発生した場合，クラス図を作成または変更が必要である．ソフトウェアパターンの構造や相互関係を理解しないで設計すると図 2 が示すように欠陥が混入したクラス図を作成してしまう．図 2 は無駄なメソッドと関連がある欠陥を示している．

本研究ではクラス図を作成や変更を加える際に発生する欠陥の混入を防ぐため，自身または第 3 者が設計した優れたクラス図を保管し，それを再利用する．このクラス図のことを本研究では Case と呼ぶ．Case の再利用例を図 3 に示す．

Case には 2 つの問題がある．この問題を以下に示す．

1. クラス図だけでは Case の意図がわからない
2. 欠陥があるモデルが混入する可能性がある

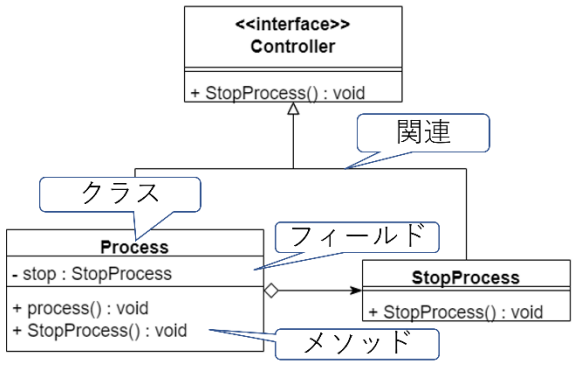

図 1 クラス図の例

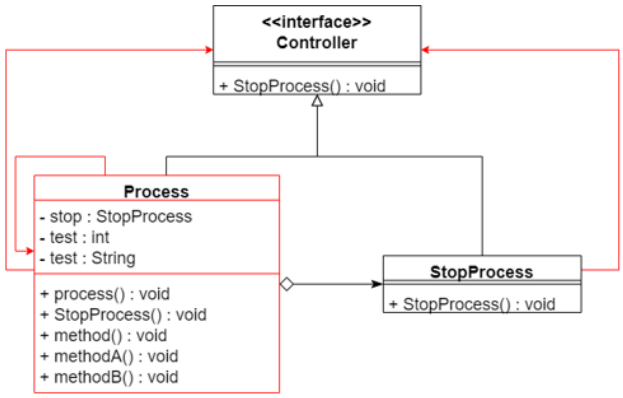

図 2 欠陥が混入したクラス図

図 1 が示すようにクラス図だけでは，意図が第 3 者に伝わらない．この状態でモデルの意図を理解するには，モデルの構造またはクラス名やメソッドで動作を理解してモデルの意図を判断する．しかしながら，人によって意図を読み違える可能性があり，適切に Case を再利用ができない．

ソフトウェアパターンは優れた設計者が作成したモデルを再利用するが Case は必ずにも優れた設計者が作成しないため欠陥が混入する可能性がある．欠陥が混入した Case を再利用するとシステム全体に悪影響を与える．そのため

† 本研究の一部は以下において発表した
・日本ソフトウェア科学会第 37 回大会公演論文集 50-L
* 電子情報メディア工学専攻 2218002 橋浦研究室

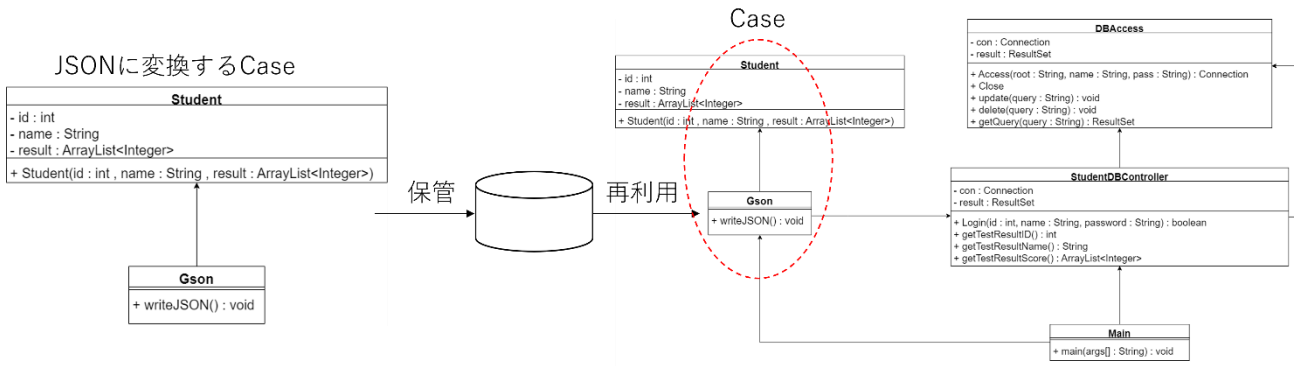

図 2 Case の活用例

欠陥を検知して省く必要がある．保管するクラス図は図 1 のクラス図を保管し，図 2 のクラス図は省く必要がある．

本研究は，2 つの問題をドキュメントと欠陥検知手法[4]を用いて解決する．

ドキュメントはデザインパターンに用いるドキュメントを参考に 5 つの要素で Case の意図を伝える．

1. 前提：Case に用いる際に必要となる情報
2. 目的：Case の意図
3. フォース：Case を再利用する際に変更してはいけない箇所
4. 実装：実装のガイドライン
5. 責務：クラスの役割

図 1 のクラス図のドキュメントを図 3 に示す．

Case の欠陥は欠陥検知手法[4]を用いる．欠陥検知手法は Anti-Pattern と Code Smell を検知する．Case で欠陥を検知した場合，その Case を保管しない．Anti-Pattern はシステム全体に悪影響を与え，Code Smell はシステムの局所的に悪影響を与える．

Case を再利用する事で，ソフトウェアパターンが対応できない問題に対応かつモデルの設計時に発生する欠陥を防ぎ，その評価を行なった結果について述べる．

2 目的

本研究の目的は，Case の保管と Case の有効性を明らかにすることである．有効性は，Case のドキュメントで Case の意図を適切に伝えられることができることと設計の手助けに貢献できることを確認する．

3 実現手法

JavaScript とグラフデータベースの neo4j[6]を用いて Case の保管および欠陥検知手法[4]を実装した．Web サイト上で Case のクラス図とドキュメントを保管する．

前提
　割り込み処理を行う
　インターフェースで処理と割り込み処理を同一化する

目的
　処理の際にエラーを検知した場合処理を割り込みをし、処理を停止する

フォース
　インターフェースに変更するとサブクラスに変更が及ぶためインターフェースに変更を加えない

実装
　Main等の呼び出しクラスからProcessクラスに関連をつける
　　Processクラスが処理の起点なため

責務
　Controllerクラス
　　ProcessクラスとStopProcessクラスを同一視する

　Processクラス
　　主に処理を行うクラスである
　　Process()は処理を行っているときエラーを検知した場合、StopProcess()を呼び出す
　　エラーの検知はtry/catchを用いる
　　StopProcess()はStopProcessクラスのインスタンスを生成し、そのインスタンスで委譲(StopProcessクラスに処理を渡す)を行う

　StopProcessクラス
　　Processクラスから呼び出しをもらったときに動作するクラス
　　現在行っている処理を終了させる
　　処理の停止はexit()を用いる

図 3 Case のドキュメント例

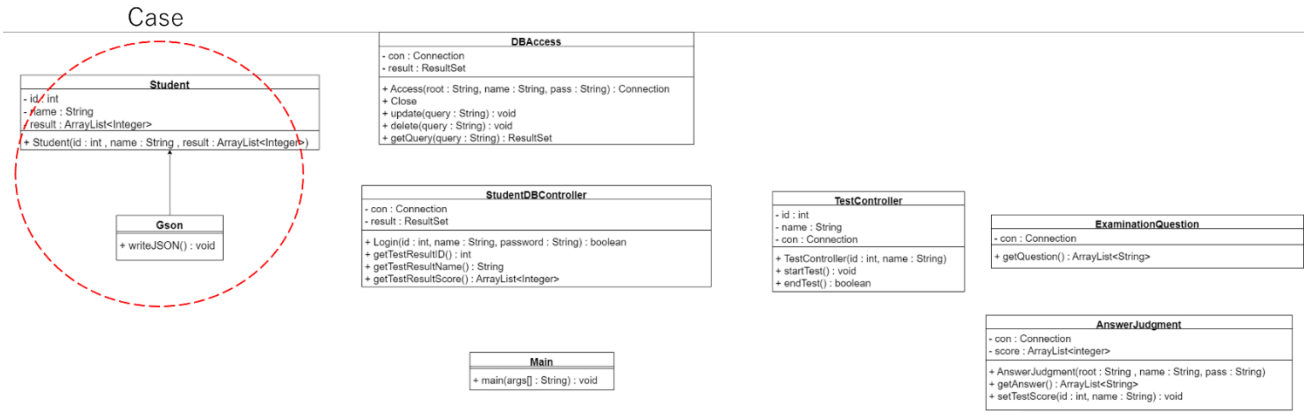

図 4 設計時に被験者に提示した Case ありのモデル

4 評価と考察

Case の有効性を明らかにするために以下の RQ を立てた.

RQ1. 適切な Case を選択できるのか

RQ2. Case を用いることで品質は向上するのか

被験者は日本工業大学の橋浦研究室の生徒 10 人で実施した. この内訳は M2 が 4 人, B4 が 2 人, B3 が 4 人である.

4.1 RQ1:適切な Case を選択できるのか

Caseのドキュメントから意図を伝えることができることの確認が必要である. 要件定義をもとに複数の Case の中から要件を満たしている Case を選択する実験を行った. 被験者に提示した情報は, 図 1 と図 3 のようなクラス図とドキュメントである. 実験に用いた要件定義は以下に示し, Case の詳細情報を表 2 に示す.

1. エラーを検知した際, 処理に割り込みしシステムを停止するシステムを求める
2. 今回のエラーは致命的なエラーを対象

表 1 Case の選択時に被験者に提示した Case の詳細情報

Case	*クラス数*	*フィールド数*	*メソッド数*	*ドキュメント量*	*要求の充足度*
1	3	1	4	331	2
2	1	0	2	201	0
3	4	0	4	298	1

表 1 の要求の充足度は, 要求を満たしているほど値が高くなり, 最大値は 2 である. この充足度から被験者の充足度の合計 /充足度の合計の値で適切な Case を選択したことを確認する. 結果は, 12/20=0.6 となり, 半分以上の被験者が要求を満たす Case を選択したことが明らかになった.

被験者が選択した各 Case の人数は, 充足度 2 が 2 人, 残りが充足度 1 の Case を選択した. この結果から Case の意図から離れた Case をドキュメントから伝えることができることが明らかになった.

4.2 Case を用いることで品質は向上するのか

Case を設計の手助けに貢献していることを確認する. Case が設計の手助けができていれば, Case を適切に再利用ができたことになる. それにより完成した設計モデルの品質は良くなる. そのため, 品質を見ることで Case が設計の手助けに貢献していることが確認できる. 品質は Anti-Pattern と Code Smell の欠陥の数で測定する. 欠陥は, 品質が低いほど数が多く検出される. そのため, Case が設計の手助けに貢献しているほど欠陥の数は少なくなる.

実験は, Case の有無で設計をしてもらい欠陥の数の差を見る. 設計は要件定義をもとに関連を引く設計を被験者に行ってもらう. 要件定義は以下に示し Case ありのモデルを図 4 に示す. 図 4 は本来はドキュメントが付いた状態だが今回は省く. また設計時に被験者に提示したモデルの詳細情報を表 2 に示す.

- Case なし : 二段階認証
- Case あり : web テストができ, 結果を DB から取り出し JSON に変換

表 2 Case の設計時に被験者に提示したモデルの詳細情報

Case 有無	*クラス数*	*フィールド数*	*メソッド数*	*正解の関連数*
Case 無	5	4	15	5
Case 有	8	13	19	10

Case 有無の品質の結果を表 3 に示す. 結果は被験者が欠陥の数を増減した人数を表記する. 欠陥を増やした人数を Case 有無で見ると Case なしのときに Anti-Pattern が増えていることが確認できる.

欠陥の数が少なくなる理由は，関連の数が正解の関連の数より少ないからだ．欠陥検知手法[4]は，平均を各クラスが超えた際に検知する．そのため数が少なくなると平均を超えずらくなり検知がしづらくなる．

Case なしに欠陥が増えた理由は，1 つのクラスに関連を多く引いたからだ．図 5 が示すように，機能を分類できないと無駄な関連を引いて 1 クラスあたりの関連数が多くなる．Case が 1 つの機能と分類ができるため，システム全体の機能の分類がしやすくなったと考える．

結果から Case を用いた際に設計の手助けに貢献していることが明らかになった．

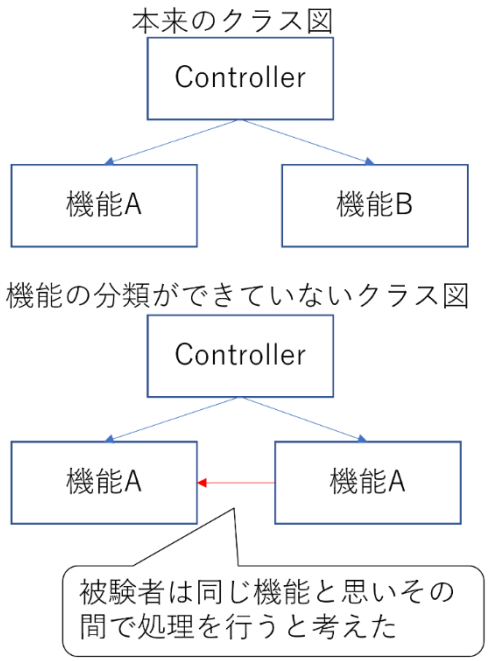

図 5 設計のモデルの機能の分類ができない例

表 3 欠陥の数を増減した被験者の数

	Case なし		*Case あり*	
	Code Smell	Anti-Pattern	Code Smell	Anti-Pattern
増加	0	4	0	0
減少	8	0	10	3

5 関連研究

Kienzle[6]はアジャイル開発のような繰り返し機能の拡張や変更を加える開発においてデザインパターンを再利用して開発の手助けを行う研究である．デザインパターンの再利用方法は，デザインパターンの機能をモデルに継承して再利用する．

Braga と Chan[7]は Web アプリケーションの設計の問題をソフトウェアパターンを再利用して解決する研究である．再利用方法は，設計問題を解決するソフトウェアパターンを検索機能を用いてモデルを出し再利用する．その際に検索に出たモデルは，説明を表示する．この説明から問題を解決できるモデルを再利用者が判断を下す．

6 結論

本研究では自身または第 3 者が作成した優れたモデルを再利用する Case とクラス図の欠陥を検知する手法を開発し，Case の有効性を明らかにする評価を行った，その結果，要求を満たす Case の選択は，半分以上の被験者が要求を満たす Case を選択できた，また，Case を用いた設計の手助けの貢献では，Case を用いた方が欠陥を増やすことがなく手助けに貢献していることが明らかになった．

参 考 文 献

1) 鷲崎弘宜,丸山勝久,山本里枝子,久保淳人,” ソフトウェアパターン：パターン指向の実践ソフトウェア開発,” 近代科学社,Dec.2007.
2) Gamma, E., Helm, R., Johnson, R. and Vissides, J., 本位田真一(訳), “オブジェクト指向における再利用のためのデザインパターン 改訂版,”ソフトバンク パブリッシング株式会社, 1999.
3) 結城浩: 増補改訂版 Java 言語で学ぶデザインパター ン入門, SB クリエイティブ株式会社, 2019.
4) 飯山 隆章, 橋浦 弘明, "ソフトウェアの概念モデルに対する欠陥検知手法の検討," 日本ソフトウェア科学会第 37 回大会公演論文集, 50-L, Sep. 2022.
5) Neo4j, https://neo4j.com, accessed:2022-08-10.
6) Jörg Kienzle, “Reusing software design models with TouchRAM, ” In Proceedings of the 12th annual international conference companion on Aspect-oriented software development (AOSD '13 Companion), pp23 - 26,Mar.2013.
7) Rosana T. Vaccare Braga and Alessandra Chan, "Peony: A Web Environment to Support Pattern-Based Development," 2008 Eighth International Conference on Web Engineering, pp. 358-361, Jul.2008.

指導教授	審査委員（主査）	准教授	橋浦 弘明
	審査委員（副査）	教授	粂野 文洋
	審査委員（副査）	教授	高瀬 浩史