開発者はどのようにしてコンフリクトを解消しているのか: コンフリクト解消の自動化に向けて

湯 月 亮 平[†1] 畑 秀 明[†1] 松 本 健 一[†1]

コンフリクトに関して多くの研究があるが，実用的かつファイルレベルでのコンフリクトの解消については多くの課題がある．本研究では，Kim らによるパターンベースのバグ修正手法がコンフリクト解消の自動化に応用できると考え，その先行研究としてソフトウェアプロジェクトごとにコンフリクトが発生したファイルをどのように修正したかを調べる．現時点で，いくつかのソフトウェアプロジェクトにおけるマージ・コンフリクトの回数が得られている．

How Do Developers Resolve Conflicts: Towards the Automation of Conflict Resolution

Ryohei Yuzuki,[†1] Hideaki Hata[†1] and Kenichi Matsumoto[†1]

Although there are various studies about conflicts, there are still various tasks in practical conflict resolution at file level. We assume that pattern-based aoumatic program repair method that Kim propsed is useful for automation of conflict resolution, and are studying how do developers resolve conflicts for each software project. Now we obtain the amount of merges and conflicts in some software projects.

1. はじめに

ソフトウェア開発において，バージョン管理はとても重要である．バージョン管理とは，ある時点でのソースコードなどを含むファイルのスナップショットを保存し（これをバージョンと呼ぶ），各バージョンを系列として管理する概念である．これにより複数人で並行してソースコードを変更することが可能となる．系列を分岐した状態にさせることができ（これをブランチと呼ぶ），各系列ごとに独立してファイルを変更することできる．また，ブランチを 1 つに統合することができ（これをマージと呼ぶ），各系列での変更を 1 つに統合することができる．しかしながら，マージを実行する上で，コンフリクトが発生してしまうという問題がある．コンフリクトとはマージを実行する 2 つ以上のファイル間での競合であり，各系列において同一ファイルの同一箇所が変更されていると発生する．コンフリクトが発生してしまうとマージが実行できず，ブランチを 1 つに統合することができない．また，そのコンフリクトを人手で解決しなければならず，それは非常に時間とコストがかかる．実際，Phillips らの研究によると，ブランチとマージに関してプロジェクト管理者にアンケート調査を行い, 回答者の内 54 %がマージにおける最重要課題はコンフリクトであるという回答が得られている 1)．

コンフリクトの軽減については，様々な手法が提案されている．例えば，Apel らはコードを半構造化することにより，文法エラーおよび構文エラーを軽減するマージ手法を提案した 2)．これによりほとんどのソフトウェアプロジェクトにおいて，コンフリクトの発生回数を減らすことができるとしている．しかし，いずれの手法によってもコンフリクトの発生を完全に無くすことはできないため，コンフリクト解消をサポートする手段，特にコンフリクトを自動的に解消する手段が必要となる．

2. 本研究のねらい

プログラムのバグ修正に関しては，いくつかの自動化手法が提案されている．Weimer らは，遺伝的プログラミングを用いてプログラムのバグの自動修正を行った 3)．具体的には，全てのテストケースが通るようになるまでランダムなコードの変更を繰り返すというものである．しかし，彼らの手法はプログラムの機能性を損なうような不合理な修正を生成する可能性が

†1 奈良先端科学技術大学院大学
NAra Institute of Science and Technology

ある．この問題に対し，Kim らはパターンベースを用いたプログラムのバグの自動修正を行った 4)．彼らはランダムにコードを変更するのではなく，60000 人以上の人が書いた修正パッチから，関数の引数を変更する，条件式を変更する等の共通の修正パターンを抽出し，それらを基にコードの変更を行った．人間の知識を活用することで，Kim らは Weimer らの手法に比べてより合理的な修正を行うことができた．

コンフリクトの解消についても，実際の人間の知識を活用すればより精度の高い自動化が可能であると考えられる．例えば，Kim らの手法と全く同様にしてコンフリクトが解消されるまで，実際のコンフリクト解消の手段を基にコード変更を繰り返すといった手法が考えられる．そこで本研究では，実際のコンフリクト解消の手段についてなんらかのパターンが存在するかどうかを明らかにするために，ソフトウェアプロジェクトごとにコンフリクトが発生したファイルをどのように修正したかを調べることを目的とする．

3. 質 問 調 査

マージに関する調査として，Phillips らの研究がある．彼らは実際にブランチとマージがどのような使い方をされているか，成功したマージ戦略とは何かを明らかにするために，およそ 300 人のプロジェクト管理者に対して 21 項目の質問調査を行った 1)．この調査により，「成功するマージ戦略は，コンフリクトの頻度・複雑さを軽減するだけでなく，品質低下を防ぐことにも焦点を当てている」と結論づけられていたが，これはあくまで性質に過ぎず，そのようなマージを実現するための具体的な手法については結局のところ明らかにされていない．それ故，本研究ではリポジトリマイニングによってより実証的なコンフリクトの実態を調べる．

4. 実証的な調査に向けて

今回は Git 内のソフトウェアプロジェクトを対象とする．Git では，リポジトリ内のファイルの内容を遡って再現することができるため，実際に 2 つのコミットに対してマージを実行しコンフリクトの発生を再現することが可能である．同様に，コンフリクトを解消した後のファイルの内容も見ることができるため，具体的にどのようにしてコンフリクトを解消したのかを調べることが可能であり，今後コンフリクト解消における何らかの自動化手法を提案した場合に，その手法によって得られた修正ファイルと実際の修正ファイルの比較により手法を評価することも可能となる．今回はその前段階として，Python 用のライブラリである GitPython を用いてマージの実行回数・コンフリクトの発生回数を調査した．

表 1 プロジェクトにおけるコミット
Table 1 Commits in Project.

	コミット総数	マージコミット数	コンフリクト
AUIL	723	43	4
libgdx	7780	1125	53
cloudstack	25754	756	197

表 1 にいくつかのソフトウェアプロジェクトごとの調査結果を示す．例えば，libgdx では 1125 回マージが実行されており，その内 53 回においてコンフリクトが発生している．コミット総数が多いプロジェクトほどコンフリクトが多く発生していることが分かる．特に cloudstack では，実行されたマージのおよそ 26 %でコンフリクトが発生している．

5. ま と め

現時点で，マージ及びコンフリクトの回数が明らかになっている．今後，対象となるソフトウェアプロジェクトを増やしながらコンフリクトを起こしたファイルを詳しく調べ，コンフリクトをどのようにして解決したのかを明らかにする．また，それらの適切な分類の仕方や他にどのような項目を調べるべきかについても検証を行う必要がある．また，本研究によって得られた知見により，どう自動化すべきか，どのような自動化手法が適切であるかについても議論する必要がある．

参 考 文 献

1) Shaun Phillips, Jonathan Sillito, and Rob Walker. : *Branching and Merging: An Investigation into Current Version Control Practices*,ICSE ’ 11.
2) Sven Apel, Jorg Liebig, and Christian Kastner. : *Semistructured Merge: Rethinking Merge in Revision Control Systems*,ESEC/FSE ’ 11.
3) Westley Weimer, ThanhVu Nguyen, Claire Le Goues, and Stephanie Forrest. :*Automatically Finding Patches Using Genetic Programming*,ICSE ’ 09.
4) Dongsun Kim, Jaechang Nam, Jaewoo Song, and Sunghun Kim. : *Automatic Patch Generation Learned from Human-Written Patches*,ICSE ’ 13.