

コンフリクトに焦点を当てた新しいバージョン管理システムの開発

堀脇 尚人[†] 長 慎也[†]明星大学大学院情報学研究科[†]

1 はじめに

今日、バージョン管理システムは、多人数が参加するプロジェクトにおいて幅広く利用されており、その種類は多様である。

本研究では、ほとんどのバージョン管理システムで発生する事象の一つであるコンフリクトに焦点を当て、既存のシステムの問題点を解決するシステムを提唱し、開発に及んだ。

2 既存のシステムの問題点

バージョン管理システムとは、主に、ソースファイルの作成、編集の履歴を管理するシステムで、CVS[1]や、Git[2]などが有名である。これらは、ネットワークを利用し、プログラマ各々がソースファイルを更新した場合にも、履歴を管理できることから、多人数による開発に適している。

バージョン管理システムの利用方法について順を追って説明すると、以下のようになる。

- ① ファイルを作成したプログラマが、ファイルをリポジトリに登録する。
- ② 各プログラマが、ファイルをリポジトリからローカル環境へ取り出す。(チェックアウト)
- ③ 各プログラマが、ローカル環境で、ファイルを編集する。
- ④ 各プログラマが、編集したファイルをリポジトリに書き戻す。(コミット)

この手順を繰り返すことで、リポジトリが更新され、プロジェクトが進行する。そして、別々のプログラマが②～④の手順を近い時間で行った場合に、一定の条件下で、ソースファイル同士の衝突(コンフリクト)が発生することがある(図1)。

コンフリクトが発生した場合には、後からコミットしようとしたプログラマが、事態の解決を行わなくてはならない。解決方法の例としては、それぞれの記述を確認しながら修正する、どちらかのソースファイルを優先するなどがあるが、全ての処理は、後からコミットしたプログラマのローカル環境内で行われており、先にコミットしたプログラマは、解決に関して、介入・支援することができない。そこで、全てのプログラマがコンフリクト状況の認識と解決を行えるシステムを開発し、よりコーディングしやすくなるような環境の構築を試みた。

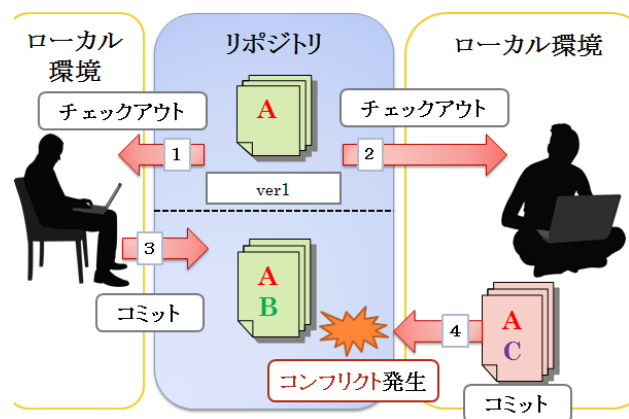


図1 コンフリクトの発生

Version control system focused on conflict

Naoto Horiwaki

Shinya Cho

[†]Graduate School of Information Science, Meisei University

3 本システムについて

本システムは、一般的な集中型バージョン管理システムの基本仕様に加えて、サブリポジトリの概念を導入している。

コンフリクト発生時に、後からコミットされたソースファイルは、リモート環境にある、リポジトリとは別の「サブリポジトリ」と呼ばれる領域に書き込まれる。サブリポジトリは、リポジトリと同じく、各プログラマが共有している空間のため、全員が、コンフリクト状況の認識が可能であり(図2)、リポジトリとサブリポジトリのソースファイルの記述を比較¹することで、コンフリクトを発生させたプログラマ以外のプログラマも、コンフリクトを解決できるようになる。図3は、本システムによるコンフリクト状況の報告と、コンフリクトしたソースファイルを修正するためのコマンドの一例である。このコマンドを入力したプログラマの作業用ディレクトリには、図4の例が示すような、リポジトリとサブリポジトリのソースファイルの記述を比較した、差分ファイルが出力される。

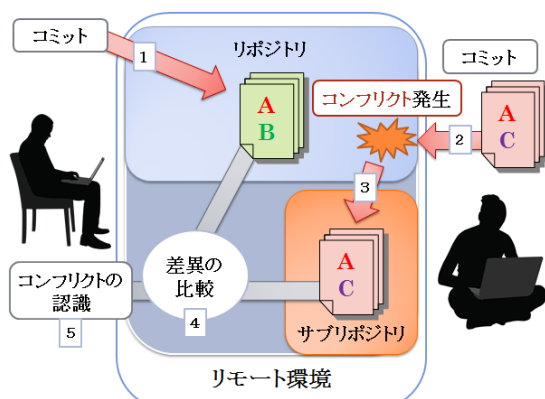


図2 サブリポジトリを利用した状況認識

```
-----message-----
コンフリクトしているファイル(Main.java)がサブリポジトリに存在します。
-----Command-----
コマンドを入力してください。
[King]>startfix
fixを開始します。
リポジトリとサブリポジトリのファイルを比較したものを、ユーザーディレクトリに出力します。
コンフリクトしているファイル(Main.java)を出力します。
```

図3 本システムによる報告

¹ diff プログラムを用いて、行単位でテキストファイル間の差異を比較する。

```
public class Main {

    Mover m;

    <<<<-----repository code-----
        float t = 0;
    -----repository code----->>>>
    <<<<-----subrepository code-----
        float time = 0;
        int x = 10;
    -----subrepository code----->>>>

    public void setup() {
```

図4 出力される差分ファイルの例

4 実験と評価

実験と評価は、本システムを実際のプロジェクトに利用することで行う。対象のプロジェクトには、本学の研究室における、卒業研究の共同開発プロジェクトを利用する予定である。その結果を元に、システムの利便性を評価する。

5 まとめと課題

コンフリクトは、プログラマ同士が常にコーディングの事前に連絡を取り合える状況にあれば、回避可能な事象である。しかしながら、必ずしもそうであるとは限らないため、発生しうることを想定しなくてはならない。本研究では、バージョン管理システムのサポートの一つに、コンフリクトの全体像を可視化させ、解決手順を示す機能を実装することで、プログラマがコーディングを行いやすくなるような環境の提供を試みた。

現在直面している課題は、三人以上のプログラマによるコミットによって、コンフリクトが発生した場合の処理である。サブリポジトリという共有空間に、コンフリクトしたソースファイルの出力を行っているため、ローカル環境で解決する既存のシステムよりも、複雑化している。今後は、本システムによる、コンフリクトの検出と処理を、明確化していく方針である。

参考文献

- [1] CVS - Concurrent Versions System
<http://www.nongnu.org/cvs/>
- [2] Git
<http://git-scm.com/>