# コード生成のための管理機能を含めたバージョン管理システム

松本 大貴† 高橋 由樹† 堀江 夏子† 後藤 滋文† 中村 章人‡ 塚本 享治†
東京工科大学† 産業技術総合研究所‡

## 1 はじめに

コンパイラによるコード生成は古くから行われてきたが、これまではコード生成前とコード生成後が別のフェーズとして切り離されており、一連の流れとしてコード生成を管理する、ということはなされてこなかった。XML（XSLT）の登場によりコード生成の敷居は低くなり、小規模な開発でもコード生成が行われるようになった。また、スパイラル型開発の増加により、生成物のファイルへの変更を生成元のファイルへと反映させる機会も増えている[1]。

一般にソフトウェア構成管理ではバージョン管理システムとビルド・ツール（Make、Ant など）を組み合わせて使用する。コード生成はビルド・ファイルに記述されているが、これらは互いに独立に管理されており、二者間の情報交換は人に任されてきた。ソフトウェア構成管理には「コード生成管理」機能が求められている。

## 2 コード生成管理機能の要件

まず、コード生成管理に求められる要件を整理し、実現方法を検討する。

### 2.1 ファイル編集とコード生成

（複数の）コードから（複数の）コードが生成されることをコード生成というが、コード生成により生成元自身が変更される例は稀である。生成元のファイル集合と生成物のファイル集合が互いに疎である場合に限定してコード生成を扱うこととした。

バージョン管理システムはファイルごとに履歴管理を行っており、コード生成を全く考慮していない。編集しているファイルが生成元であるかどうか、生成物はどのくらいあるか、その生成物が他のファイルの生成元になっているかどうか、といったことは利用者が把握しなければならない。また、生成物への変更を生成元へと反映させようとした場合、編集している生成元を頭の中で展開（コード生成）し、生成物との比較をする必要がある。

生成元と生成物の関係をバージョン管理システムに付加して、以下の 2 つの機能を実現する。これを「コード生成管理」と呼ぶことにする。

①あるファイルへの変更が他のファイルに対してどの程度の影響を与えるか利用者に提示する
②生成元へ変更を加えたときに一時的にコード生成し、生成物との差分比較を利用者に提示する

### 2.2 ファイルへの影響

図 1 はリポジトリ上でファイルがどのように変移していくのかを表している。矢印はファイルの変更を示すものであり、要因としては、ファイルの編集やコード生成、あるいはそれらの組み合わせが考えられる。T は時間の変化を示したもので、時間の経過とともに数字が増える。ファイルは変更が加えられると数字が増える（$c_0 \rightarrow c_1$）。時刻 $T_0$から時刻 $T_3$において、a と b から c と d が生成される関係（a, b ⇒ c, d）があり、時刻 $T_4$おいては a と b から c と e が生成される関係（a, b ⇒ c, e）があったとする。

$T_2$にてファイル a を変更することは、ファイル c とファイル d が変更されることにもなり、ファイル a への変更は注意が必要である。依存関係（a, b ⇒ c, d）が解れば、予め利用者に情報を提示することができる。

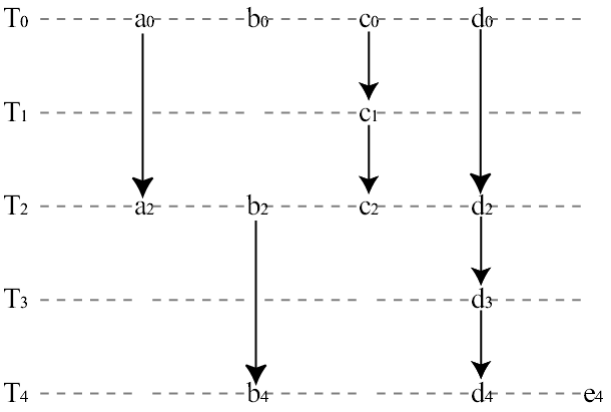


図 1：ファイル変移図

時刻 $T_3$と時刻 $T_4$では、依存関係が異なるため、時刻 $T_3$でファイル c はファイル a とファイル b から生成される関係にあったが（a, b ⇒ c, d）、時刻 T4 では独立している（a, b ⇒ d, e）。c は時刻 $T_4$において削除される、依存関係のない独立した状態で存在する、2 つの状態が考えられる。また、

“A version control system with the management function for code generation”, by †Taiki MATSUMOTO, †Yuki TAKAHASHI, †Natsuko HORIE, †Shigehumi GOTO, ‡Akihito NAKAMURA, †Michiharu TSUKAMOTO, †Tokyo University of Technology and ‡The National Institute of Advanced Industrial Science and Technology

新たにファイル e が作成される。コード生成によってファイルが作成・変更されたり、手動で作成・変更されたりしても、バージョン管理システムは区別することができない。依存情報を管理することで、ファイル c が時刻 $T_4$ にて不要である可能性を検出でき、利用者に提示することが可能となる。

### 2.3 コードの一時的な展開

コピー&マージモデルを採用したバージョン管理システム（Subversion[2]、CVS[3]など）は複数ユーザによる同時編集時に生じた競合を自動的に解決、または提示する機能を備えている。新しい生成元を一時的に展開したコードと、以前の生成物のコードとの比較を行い、差分を提示することで、利用者のコーディングの補助が可能となる。

### 2.4 コード生成の関係（依存関係）

上述のコード生成管理機能を実現するためには、バージョン管理システムが生成元と生成物の関係を把握しなければならない。依存情報は変移するため、依存情報自体がバージョン管理される必要がある。リビジョン番号と同じように属性情報としてファイルごとに記述する方法と、独立した関係表を用意し、それ自体をバージョン管理する方法とが考えられる。複数の生成元ファイルと複数の生成物ファイルの関係をファイル単位で記述することは大変であり、（独立した関係表を用意し）フォルダ指定やワイルドカード、正規表現を使ってグループを指定するほうが簡単である。

## 3 システム設計

前章の検討に基づいて、コード生成管理機能を含めたバージョン管理システムを設計した。本章ではこのシステムの概要を述べる。

### 3.1 システム概要

本システムの実装には、Subversion を採用した。オープンソースである、タグ（スナップショット）を自動で作成してくれる、API による機能の抽象化がしっかりしている、などが理由である。プログラミング言語は C 言語で書かれた Subversion の API を利用するため、C++を選んだ。

本システムの構成を図 2 に示す。作業コピー管理ライブラリは Subversion に用意されているクライアントライブラリのひとつである。依存関係ライブラリは Subversion ファイルシステムに用意した依存関係を記述したファイル（2.4 参照）を元に、依存検出を行う。展開ライブラリは前章の 2.3 で述べた差分提示を行うために、ファイルの一時展開を行う。

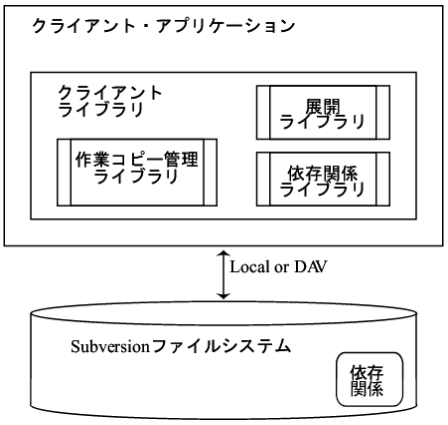


図 2：システム構成図

### 3.2 競合検出、競合解決の流れ

本システムが，コード生成における競合を検出し、解決する流れを以下に記す。

状態：作業コピーには、生成元のファイル $A_0$ と生成先のファイル $B_0$ があり、$B_0$ には変更が加えられ $B_1$ となっており、ファイル A を編集しようとしている。

①システムは依存関係ライブラリにより A と B との依存関係を検出し、利用者に依存情報を提示する。
②利用者が A を編集する。
③システムは展開ライブラリにより、$A_1$ から B’を一時的に生成する。
④システムは B’と $B_1$ を比較し、自動解決が行える場合 $B_2$ を作成し、行えない場合はその差分を利用者に通知する。
⑤利用者は差分情報を参考にしながら $A_0$ を編集し、$A_2$ を作成する。

## 4 おわりに

本稿では、バージョン管理システムにコード生成管理機能を付加することによる、人的処理の自動化や生産性の向上とその具体的な手法について述べた。

ソースから実行ファイルを生成する点において、ビルド・ツールもコード生成を行っているといえる。コード生成管理をビルド管理にまで拡張し、バージョン管理との関係を密接にすることを計画している。

## 参考文献

[1] Jack Herrington: Code Generation in Action (In Acti-on), Independent Pub Group, 2003.
[2] Subversion project. http://subversion.tigris.org/
[3] Concurrent Versions System. http://www.cvshome.org/