ソフトウェア工学　136－7
（２００２．３．７）

オープンソースソフトウェア開発プロセスに適した CVS リポジトリの階層的分散構成のモデル化と設計

嶋田 大輔† 藤枝 和宏‡ 落水 浩一郎†

†北陸先端科学技術大学院大学 情報科学研究科

‡北陸先端科学技術大学院大学 情報科学センター

〒 923-1292 石川県能美郡辰口町旭台 1-1

Phone: 0761-51-1699(内線 1363), Fax: 0761-51-1360

e-mail: {dshimada, fujieda, ochimizu}@jaist.ac.jp

概要

オープンソースソフトウェアの開発プロジェクトでは、規模が大きくなるにつれ、目的の違う複数のサブプロジェクトを持つ場合がある。これらの開発で広く用いられている CVS は、リポジトリが単一構成のため、サブプロジェクトごとに異なる運用方針や管理方針の設定が難しい。本稿では、この問題を解決する CVS リポジトリの階層型分散モデルを提案する。このモデルでは、リポジトリ間の関連を維持しながら、リポジトリごとに運用方針と管理方針が設定可能である。

和文キーワード

オープンソースソフトウェア開発、管理方針、運用方針、CVS リポジトリ、階層型分散構成

A Model of Hierarchical Distribution of CVS Repository for Open Source Software Development

Daisuke SHIMADA† Kazuhiro FUJIEDA‡ Koichiro OCHIMIZU†

†School of Information Science, JAIST

‡Center for Information Science, JAIST

Asahidai 1-1, Tatsunokuchi, Ishikawa 923-1292

Abstract

Open source software development often has several sub-projects with different goals along with its augmenting scale. CVS (Concurrent Versions System) used in such development can have only one repository. So it is difficult to support various policy of operation and management for these sub-projects. In this paper, we propose a model of hierarchical distribution of CVS repository to solve this problem. It allows us to set policy proper for each of repositories and to manage various relationships of them.

英文 key words

open source software development, policy of management, policy of operation, CVS repository, hierarchical distributed configuration

1 はじめに

近年、インターネットの普及と計算機資源の発達に伴い、広域分散環境においてソフトウェアの共同開発が行われている。この 1 つに、**オープンソースソフトウェア開発**がある。著名なプロジェクトとして、*Apache*[1]、*Mozilla*[2]、*Gnome*[3] が挙げられる。

オープンソースソフトウェア開発で広く利用されている版管理システムとして、*CVS (Concurrent Versions System)*[4] がある。CVS は、並行作業モデルや複数のクライアント認証方式など、インターネットを介した共同開発に適した特徴を持っている。しかし、CVS のリポジトリは、単一構成のため以下の問題が生じている。

- **開発拠点ごとに異なる管理方針と運用方針を適用できない**

  オープンソースソフトウェア開発では、複数の開発拠点を持ち、開発拠点ごとにソースコードの変更方針等の異なる成果物管理方針を持つ場合がある。また、各拠点は、成果物の管理だけではなく、リソース管理、アクセス権管理、コミット権限管理といったプロジェクト管理を独自に適用できる独立したリポジトリが必要である。

  CVS のリポジトリ構成では、複数の異なる管理方針や運用方針を共存させることが難しい。

- **リポジトリの負荷分散やアクセスの分離ができない**

  世界中から単一リポジトリに全てのアクセスが集中するため、計算機資源に高い負荷がかかり開発に支障をきたす恐れがある。また、リポジトリに対して、開発を行うためのアクセスと、開発以外の目的 (参照、版の取得など) のアクセスを分離できない。

これらの問題点から、オープンソースソフトウェア開発を支援する版管理システムは、**リポジトリの分散構成**を持つことが必要である。

オープンソースソフトウェアは、広域分散したユーザから、アイデア、ソースコード、テストやバグレポートなどの貢献を受け付け、それらを取り込むことで開発プロジェクトの推進力と発展が得られてきた。そこでリポジトリ分散モデルは、サブプロジェクトの貢献をメインプロジェクトのリポジトリ (マスタリポジトリ) に反映可能にするため、リポジトリ間において成果を反映させる、または取り込む関連が必要である。この関連は、リポジトリ間である程度の**版の矛盾**を許すが、最終的に成果を反映するために**一貫性**を保つべきである。

そこで本稿では、以下の特徴を持つリポジトリの階層型分散モデルを提案する。

feat.1 **リポジトリの分散は階層型 *(親-子関係)* である**

feat.2 **プロジェクト構造ごとにリポジトリの分散構成を柔軟に構成できる**

feat.3 **リポジトリごとに異なる成果物管理方針やプロジェクト運用方針を設定できる**

feat.4 **リポジトリ間の関連を維持できる**

本稿の構成は、2 節で既存のリポジトリ分散モデルの問題点を指摘する。3 節では、既存の問題点を解決する本モデルの概要を述べる。4 節では、リポジトリ間の一貫性と関連の維持機構を詳細に述べる。5 節では、本モデルに基づいたプロトタイプシステムの設計を示す。最後に、6 節でまとめと今後の課題について述べる。

2 既存のリポジトリ分散モデルの問題点

既存のソフトウェア構成管理システムや版管理システムには、リポジトリの分散構成をサポートするものがある。オープンソースソフトウェア開発の要求を満たすリポジトリの分散モデルは、はじめにでまとめた feat.1 から feat.4 の特徴を持つ必要がある。それらの特徴を踏まえ、既存の分散モデルの問題点について指摘する。

Rational ClearCase MultiSite

Rational ClearCase[5, 6] は、大規模な開発プロセスに適したソフトウェア構成管理システムである。版管理機能だけではなく、プロジェクト管理、ビルド管理、テスティング、デバッガなど多くの機能を備えている。

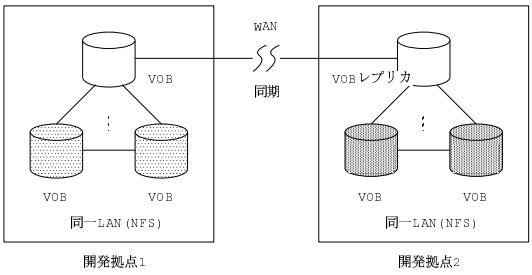

図 1: VOB のリポジトリ分散モデル

ClearCase のリポジトリ VOB(バージョン付きオブジェクトベース) は、同一の LAN(NFS) 上で分散を実現している。VOB の WAN レベルの広域分散を実現するためには、Rational ClearCase MultiSite [7] を利用する必要がある。

ClearCase MultiSite は、VOB のレプリカを作成して、常に VOB のレプリカ間で同期をとる必要があるために、各リポジトリ固有の構成や管理方針を持つことができない。これより、feat.1、2 の特徴が備わっていない。

BitKeeper

BitKeeper[8] は、オープンソースソフトウェア開発の支援を目的とした版管理システムである。リポジトリの分散やディスコネクテッドオペレーションなど分散環境を考慮した多くの機能を持つ。リポジトリの分散は、clone コマンドを用いて parent(複製元) から child(複製先) を作成することで実現している。各開発者は、child を持ち、作業成果を parent に反映 (push) させる作業スタイルをとっている。

しかし、基本的に child は、parent リポジトリの全複製であり、リポジトリごとの固有の版構成を取れない。また、リポジトリを作成する際、チェンジセットを指定できるが、特定のリビジョンまでの全複製しか指定できない。このため、ある特定の範囲のリビジョンや、特定の版のみといった木目細やかな複製ができない。これより、feat.3 の特徴が備わっていない。

CVSup

CVSup[9] は CVS リポジトリのミラーリングツールである。マスタリポジトリ (複製元) とローカルリポジトリ (複製先) の差分検出を行い、RCS 形式ファイルの高速な複製を行う。ローカルリポジトリでは、マスタリポジトリと衝突の発生しないブランチ番号を付けることによって、マスタリポジトリと同期をとりながら、そのブランチで独自の開発を行える。

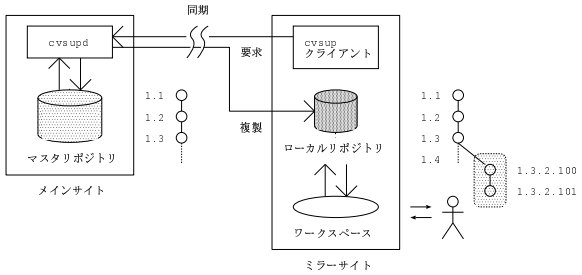

図 2: CVSup のリポジトリ分散モデル

マスタ–ローカルリポジトリ間の関連は単方向 (マスタからローカル) であり、ローカルリポジトリの成果をマスタリポジトリに反映させる機能を持たない。このため、各ローカルリポジトリからのフィードバックは、マスタリポジトリの版との差分を抽出し、パッチの作成を手動で行う必要がある。これより、feat.4 の特徴が備わっていない。

3 リポジトリの階層型分散モデル

本稿で提案するリポジトリの階層型分散モデルは、CVS のバージョンモデルに基づき、特に版の複製元である親リポジトリと、複製先である子リポジトリ間の関連を維持することを可能にする。2 階層モデルを図 3 に示す。

親/子リポジトリとは、版の複製元 (親リポジトリ) と複製先 (子リポジトリ) の関係である。子リポジトリの版は、基本的に親リポジトリの版のサブセットである。CVS リポジトリは、テキストベースの成果物 (ソースコード、設定ファイル、ドキュメントなど) を RCS ファイル形式 [10] で保存する。リビジョン付けはファイル単位に行われる。本モデル

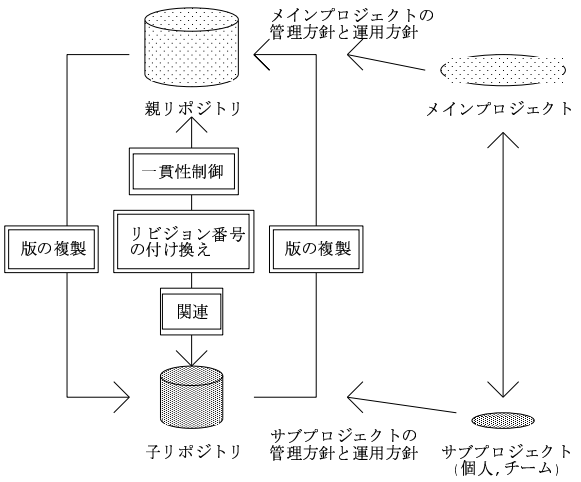

図 3: 2 階層モデル

では、リポジトリ固有の版構成を実現するため、必要な版のみを複製可能にする。版の複製を行う際、以下の 2 つを指定可能にする。

- **モジュール名**

  親リポジトリに対して、全部、特定のモジュール、特定のファイルと指定可能

- **リビジョン番号**

  モジュールに対して、全部、特定の範囲、特定のリビジョンと指定可能

複製した版は、基本的に Read 許可されている。親リポジトリの管理者は、子リポジトリの版に対して Write 許可を与えることによって、子リポジトリで独自の開発を許可する。

以下に、本モデルの**基本シナリオ**を述べる。

(1) 親リポジトリから子リポジトリに版を複製する

(2) 親リポジトリと子リポジトリでは、異なる管理方針と運用方針に基づき、独立に並行開発が進められる

(3) 子リポジトリは、自身の開発成果を親リポジトリに反映させる

  または、親リポジトリは、子リポジトリの開発成果を取り込む

親–子リポジトリ間では独立に開発が進行するため、版の内容やリビジョン番号に矛盾が生じる。最終的に、子リポジトリの成果を親リポジトリに反映するため、版の矛盾を防ぐ一貫性制御と、リビジョン番号の矛盾を防ぐリビジョン番号の操作を定義する。また、必要に応じてリポジトリ間の関連を設定可能にする。

これまでは、2 階層 (親/子リポジトリ) の分散構成を説明したが、実際の開発に利用される場合では、多階層が考えられる。また、複数の関連プロジェクトを持つ場合では、複数の親/子リポジトリを持つことが考えられる。子リポジトリがさらに子を持つ、複数の子リポジトリを持つ、複数の親リポジトリを持つ場合なども、同じモデルで行うことができる。

本モデルは、前述した 4 つの特徴を持つ。feat.1 は、**版の複製元**(親リポジトリ) と**複製先**(子リポジトリ) の関係から成り立つ。feat.2 は、2 階層モデルを組み合せることで、複数の親/子リポジトリや多階層の構成を実現する。feat.3 は、feat.4 を守った上で、独自に設定できるようにする。feat.4 は、版の矛盾を防ぐ**一貫性制御**と、リビジョン番号の矛盾を防ぐ**リビジョン番号の操作**を導入する。また、必要に応じて**リポジトリ間の関連**を設定可能にすることで実現する。

4 リポジトリ間の一貫性維持機構

本節では、基本モデルの構成要素である、一貫性制御、リビジョン番号の操作、リポジトリ間の関連の詳細について説明する。

4.1 一貫性制御

開発が進行するにつれ、子リポジトリに独自のファイルやモジュールが作成され、また親リポジトリから複製した版に対して独自の進化が行われ、親–子リポジトリ間の版に矛盾が生じる。親–子間にある程度の矛盾を許可し、子リポジトリの成果を親リポジトリに比較的容易にマージ可能にするため、2 つの一貫性制御方式を定義する。

Int.1 ブランチ制御方式

ブランチ制御方式では、子リポジトリの開発はブランチを作成して行う。子リポジトリが親リポジトリの全てを複製した場合、その後の親リポジトリの

版の進化は全て子リポジトリに反映される。子リポジトリで試験的なマージを行いコンフリクトを解消したのち、親リポジトリに子リポジトリの成果を反映させる。

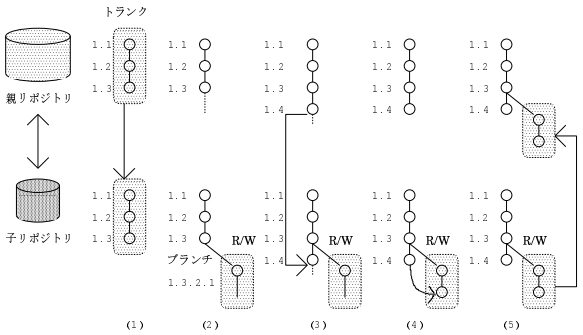

図 4: ブランチ制御方式

図 4 はブランチ制御方式を用いた場合の版の進化を示している。(1) まず、親リポジトリから子リポジトリに必要な版を複製する。(2) 子リポジトリの開発はブランチを作成して進められる。トランクは Read-Only である。(3) 親–子リポジトリ間の関連として版の同期を設定した場合、親リポジトリの版の進化が子リポジトリに自動的に反映される。(4) 子リポジトリでは、親リポジトリに成果を反映させる前にトランクと試験的なマージを行い、コンフリクトを解消する。(5) 子リポジトリは、親リポジトリに反映要求を発行し親リポジトリに承認されたなら、親リポジトリにブランチを作成し子リポジトリの成果を反映させる。または、親リポジトリは、子リポジトリの成果を主導的に取り込む。

子リポジトリが親リポジトリの一部を複製している場合が考えられる。また親–子リポジトリ間に版の同期を設定しない場合もある。もしマージを行う際、子リポジトリに複製していない版が必要な場合、不足しているファイルを親リポジトリから子リポジトリに複製する。

Int.2 ロック制御方式

ロック制御方式では、親リポジトリの該当ファイルにロックを掛け、子リポジトリの開発が行われる。親リポジトリの開発の一部を子リポジトリに委譲する場合に用いられる。子リポジトリの成果は、子リポジトリから親リポジトリに主導的に反映される。

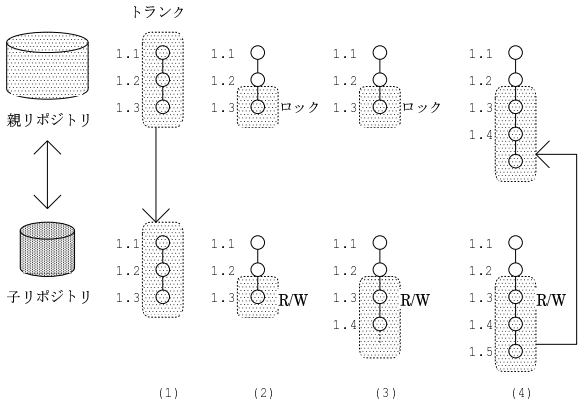

図 5: ロック制御方式

図 5 は、ロック制御方式を用いた場合の版の進化を示している。(1) まず、親リポジトリから子リポジトリに必要な版を複製する。(2) 親リポジトリの該当ファイルにロックを掛ける。(3) 子リポジトリの作業はトランクまたはブランチを作成して進められる。(4) 子リポジトリは、成果を親リポジトリに主導的に反映させる。

4.2 リビジョン番号の操作

親–子リポジトリ間の矛盾は、版の内容だけではなく、リビジョン番号にも発生する。リビジョン番号の矛盾を解消するため、リビジョン番号の操作を定義する。

子リポジトリにブランチを作成して作業を行う場合、そのリビジョン番号は親リポジトリのブランチ番号と衝突しない番号にする必要がある。開発が進行すると、親リポジトリに新規にブランチが作成され、子リポジトリの既存のブランチ番号と衝突する恐れがある。そこで、ブランチ番号の衝突を防ぐため、リビジョン番号の付け換えを定義する。

図 6 では、ブランチ制御方式を用いて子リポジトリで開発を行っている。(1) 子リポジトリは、親リポジトリから版を複製し、ブランチを作成して作業を進める。(2) 親リポジトリの同一のブランチポイントにブランチが作成され、親リポジトリと子リポジトリとブランチ番号が衝突する。それを自動検出し、該当する子リポジトリのブランチ番号を 1 つ進める。(3) 子リポジトリの成果を親リポジトリに

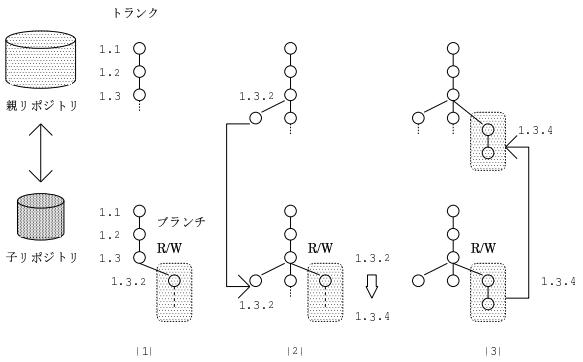

図 6: リビジョン番号の付け換え

反映させる際、ブランチ番号の衝突を防ぐことができる。

4.3 リポジトリ間の関連

本モデルでは、親–子リポジトリ間の版の内容とリビジョン番号の矛盾を解消する機能だけではなく、必要に応じて関連も設定可能にする。基本的な関連は、版の同期、フィードバックの期間、版の矛盾の通知、開発の委譲を設ける。

R.1 版の同期

版の同期は、親リポジトリの版の進化を一定時間ごとに子リポジトリに反映させることを可能にする。この関連では、親–子間のリビジョン番号が衝突する可能性があるため、自動的にリビジョン番号の付け換えが行われる。

例えば、マスタリポジトリのミラーサイトを構成する場合、リポジトリ間にこの関連を設定することにより、マスタリポジトリと同一の版構成を持つリポジトリを実現できる。

R.2 フィードバック期間

フィードバック期間は、親リポジトリに子リポジトリの成果を反映させる期間を設定可能にする。親リポジトリは手動でリポジトリの成果を取り込むのではなく、決められた期限、または一定時間ごとに自動的に成果が反映される。

R.3 コンフリクト通知

コンフリクト通知は、親リポジトリの版に対して、子リポジトリの版に設定値以上の矛盾が生じた場合、親リポジトリの管理者に通知を行う。

この関連は、親リポジトリのユーザが子リポジトリの開発状況を把握する指標として利用されることを想定している。

R.4 開発の委譲

開発の委譲は、親リポジトリで行われている開発の全部、または一部を子リポジトリに委譲できる。これは、前述したロック制御方式を用いて行う。

この関連により、開発が収束し主にメンテナンスを行う場合、開発チームからメンテナンスチームに管理の委譲を行うことができる。

5 プロトタイプシステムの設計

5.1 システムの概要

前述したモデルを実現するために、従来の CVS と CVSup プログラムを再利用し、リポジトリ間の関連を維持する親/子リポジトリサーバと、CVS プロキシを導入する。親/子リポジトリサーバは、互いに協調動作し、親–子リポジトリ間の版の複製、一貫性制御、リビジョン番号の操作、リポジトリの関連の処理を担当する。

CVS プロキシは、CVS サーバ–クライアント間のプロキシとして機能し、CVS コマンドであれば CVS サーバに、親/子リポジトリコマンドであれば親/子リポジトリサーバに命令を転送する。

図 7 にシステムの構成図を示す。

リポジトリ間の関連を操作する命令は、従来の CVS コマンドに親/子リポジトリコマンドを追加する。CVS プロキシは、コマンドによって各デーモンプログラムに命令を転送する。CVS のユーザは、従来の CVS と同様に CVS クライアントからそれらの命令を発行することにより、親–子リポジトリ間の関連を操作することを可能にする。

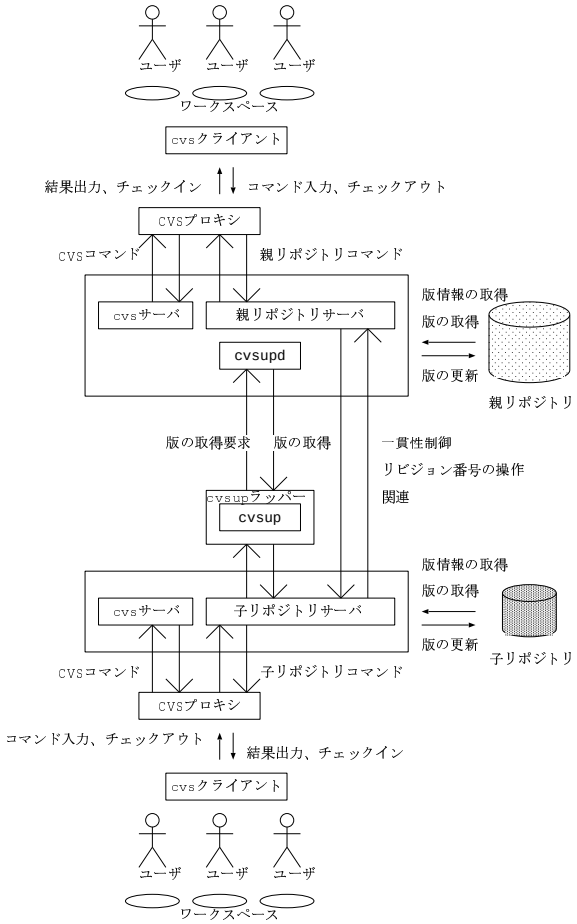

図 7: プロトタイプシステムの構成図

5.2 再利用するソフトウェア

CVS サーバ/クライアント

CVS クライアントは、CVS コマンドや親/子リポジトリコマンドを発行する。各ユーザは、CVS クライアントを使用してそれぞれのリポジトリにアクセスを行う。

CVS サーバは、CVS プロキシから転送された CVS コマンドを受け、リポジトリに対してチェックイン、チェックアウト処理を担当する。

CVSup サーバ/クライアント

親リポジトリから子リポジトリに版の複製を行うために、CVSup を利用する。CVSup のクライアント cvsup は、子リポジトリサーバから起動される。親リポジトリサイドの CVSup サーバ cvsupd は、cvsup からの要求を受け、親リポジトリから子リポジトリに指定された版を複製する。

5.3 CVS プロキシ

CVS プロキシは、CVS サーバ–クライアント間のプロキシとして機能し、ユーザからの命令を受け、命令によって各デーモンプログラムに処理を依頼する。CVS コマンドであれば、CVS サーバに命令と引数を転送する。親/子リポジトリコマンドであれば、親/子リポジトリサーバに命令と引数を転送する。

5.4 親/子リポジトリサーバ

親リポジトリサーバ

親リポジトリサーバは、親–子リポジトリ間の関連を維持する親リポジトリサイドのデーモンプログラムである。CVS プロキシから転送された親リポジトリコマンドに基づき、版の取得、一貫性制御、リビジョンの圧縮、リポジトリ間の関連の処理を担当する。

表 1 に親リポジトリコマンドの一覧を示す。

表 1: 親リポジトリコマンド

命令	引数	説明
show	*REPOSITORY*	版情報の表示
push	*REPOSITORY* *MODULES* *FILES* [*REVS*]	版の複製
pull	*MODULES* *FILES* [*REVS*]	版の取得
accept		反映要求の承認
sync	[-t *sec*] *REPOSITORY*	版の同期の設定
term	[-t *sec*] *MODULES* [-t *sec*] *FILES*	フィードバック期間の設定
confict	[-c *percent*] *MODULES* [-c *percent*] *FILES*	コンフリクト通知の設定
delegation	*MODULES* *FILES*	開発の委譲の設定

※ REPOSITORY (子リポジトリ)、FILES (1 つ以上のファイル)、MODULES (1 つ以上のモジュール)、REVS (全部、特定の部分、特定のリビジョン番号)、- t (時間設定オプション)、-c (コンフリクト値設定オプション)

各コマンドは、CVS の管理者グループ (cvsadmin) に属するユーザが実行権限を持っている。各コマンドは、従来の cvs コマンドと同様に、dcvs コマンドの引数として指定可能にする。

コマンドの実行例 (子リポジトリの情報取得)：

dcvs show *REPOSITORY*

子リポジトリサーバ

子リポジトリサーバは、親–子リポジトリ間の関連を維持する子リポジトリサイドのデーモンプログラムである。CVS プロキシから転送された子リポジトリコマンドに基づき、版の複製、一貫性制御、リビジョンの圧縮、リポジトリ間の関連の処理を担当する。

表 2 に子リポジトリコマンドの一覧を示す。親リポジトリコマンドと同様に dcvs コマンドの引数として指定可能にする。

表 2: 子リポジトリコマンド

命令	引数	説明
show	*REPOSITORY*	版情報の表示
pull	*REPOSITORY* *MODULES* *FILES* [*REVS*]	版の取得
request	*MODULES* *FILES*	反映要求
push	*MODULES* *FILES* [*REVS*]	版の複製

※REPOSITORY (親リポジトリ)、FILES (1 つ以上のファイル)、MODULES (1 つ以上のモジュール)、REVS (全部、特定の部分、特定のリビジョン番号)

6 おわりに

本稿では、リポジトリごとに運用方針と管理方針を設定可能な CVS リポジトリの階層型分散モデルを提案した。このモデルでは、子リポジトリの開発成果を親リポジトリに反映可能にするために、リポジトリ間の関連を維持する機構と、版の矛盾を防ぐ機構を導入している。このことより、メインプロジェクトとサブプロジェクト間で、協調した並行開発や開発の委譲が可能になる。本モデルにより、オープンソースソフトウェアの開発プロセスをより適切に支援可能な版管理システムを構成できる。

現在、プロトタイプシステムの実装を行っている。次の段階では、実装したシステムを用いて、オープンソースソフトウェア開発のシミュレーションを行い、その有効性を評価する。さらに、開発プロジェクトに応じて、柔軟にリポジトリ構成を取れるようにシステムを再実装する。また、本システムを利用した場合の作業モデルを作成し、それを本モデルに反映させることを予定している。

参考文献

[1] The Apache Software Fundation
http://www.apache.org/

[2] The Mozilla Organization
http://www.mozilla.org/

[3] The GNOME project
http://www.gnome.org/

[4] Karl Fogel: *Open Source Development with CVS: Learn How to Work With Open Source Software.* Coriolis Group Books, October 1999.

[5] ClearCase
http://www.rational.com/
products/clearcase/index.jsp

[6] David B.Leblang.: *The CM Challenge: Configuration Management that Works.* Configuration Management, 1994, pp1-37.

[7] Larry Allen, Gary Fernandez, Kenneth Kane, David Leblang, Debra Minard, John Posner: *ClearCase MultiSite: Supporting Geographically-Distributed Software Development.* SCM lecture notes LNCS 1005, pp 194-214.

[8] BitKeeper
http://www.bitkeeper.com/

[9] CVSup
http://www.polstra.com/
projects/freeware/CVSup/

[10] Walter F. Tichy: *RCS–A System for Version Control.* Software–Practice and Experience, Vol.15, No.7, 1985, pp 637-654.