# バージョンアップ・バージョンダウン機能を備えたシステム構成管理の実現

堀江 夏子† 後藤 滋文† 松本 大貴† 高橋 由樹† 中村 章人‡ 塚本 享治†
東京工科大学メディア学研究科† 産業技術総合研究所‡

## 1 はじめに

Web サーバなどの外部に公開するコンピュータでは、充分なセキュリティ対策を行うために、頻繁にシステムの構成管理を行う必要がある[1]。従来では、試行錯誤的にソフトウェアをインストール/アンインストール/バージョンアップ/バージョンダウンすることでシステムの構成管理が行われてきた。しかし、ソフトウェアは他のソフトウェアに依存したり、分散システムの構成要素として利用されるコンピュータでは稼働環境に依存したりするなど、ソフトウェアのみに注目した管理だけではシステム全体の管理までカバーしきれないのが現状である。そこで、複数のソフトウェアにまたがってシステム全体で構成管理を行うために、システム構成のバージョンアップ/ダウンを行う研究を進めている。

本稿では、LAN 内で稼動する基本的なサーバシステムを対象としたシステム構成管理について述べる。

## 2 従来のシステム構成管理の問題点

従来のシステム構成管理手法のひとつに、ソフトウェア本体と構成管理に必要な情報をパッケージ化し、パッケージ単位で管理を行う方式がある。そのようなパッケージ方式として、RedHat Linux の RPM(Redhat Package Manager)、Debian の deb(DEBian package)などがある。これらのうち RPM パッケージを利用した構成管理手法の問題点を以下に挙げる。

**(1) ソフトウェア単位の管理手法である**
ソフトウェア単体に対してのみの管理機能を持つ。

**(2) システム構成の移行が煩雑である**
現在の設定を引き継いで、システムを新しい構成に移行する、あるいはある時点の状態に戻すには手作業で複雑な手順を踏む必要がある。

**(3) 設定ファイルが複雑で不統一である**
ソフトウェアの設定ファイルは、記述形式、ファイルフォーマットが不統一である。

**(4) 設定間の整合性を論理的に確認する手段がない**
ソフトウェアの設定ファイルの中には、異なるソフトウェア間で同じ値を設定する場合がある。そうした設定ファイルの依存関係が論理的に表現されていない。

## 3 提案するシステム構成管理

### 3.1 問題の解決方針

前述した問題点を解決するシステム構成管理の仕組みを実現するにあたって、前章の(1)～(4)の問題点に対して次に挙げる①～④の解決方針を設定した。

**① ソフトウェアをまたがったシステム構成管理**
ソフトウェアの依存関係を考慮した構成管理を行う。

**② システム構成の円滑な移行**
システム構成の履歴を蓄積し、現在の設定を引き継いだまま新たな構成へ移行、あるいは過去のシステム構成を再現することを可能にする。

**③ 設定ファイル管理の効率化**
設定を統一的な記述形式で記述し、システム構成の変更に応じて、最適な設定ファイルを生成・適用させる。

**④ 設定の依存関係の論理的な記述**
ソフトウェア間、システム-ソフトウェア間、システム-システム間での設定ファイルの依存関係を論理的に記述し、適切に設定されているか検証する。

### 3.2 システム構成管理の概要

本稿では、RedHat Linux9 がインストールされた、LAN 内で稼動するコンピュータを想定し、ソフトウェアとして Web サーバの Apache, サーブレットコンテナの Tomcat, ファイル共有サーバ Samba, RDBMS の MySQL, 全文検索システム Namazu を対象にしたシステム構成管理を取り上げた。このシステム構成管理の流れを図 1 に示す。

**(1) システム構成管理の実行**
コンピュータのシステム管理者がシステムの構成管理を実行する。

**(2) 依存関係/影響範囲の調査**
構成管理の対象となるソフトウェアの依存関係を調査して依存の集合を求め、システムに及ぼす影響の範囲を検出する。構成管理の実行に必要なソフトウェア、妨げとなるソフトウェアをリスト化する。

**(3) システム構成の状態登録**
前フェーズで求めた影響範囲をもとに、現在の状態をリポジトリにチェックインする。チェックインした情報はバージョンダウンの際に利用される。

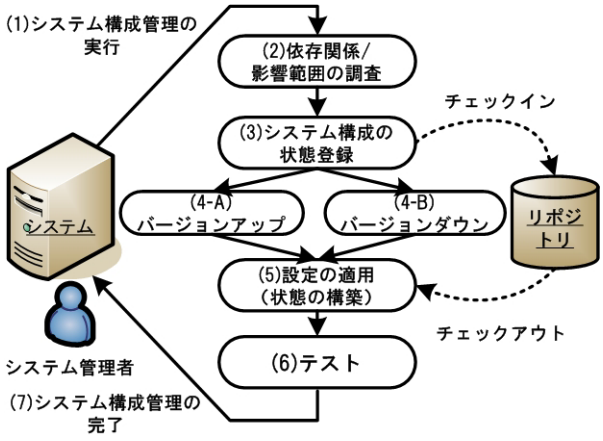


図 1:システム構成管理の流れ

"Version Up & Down Management of System Configuration", by Natsuko HORIE†, Shigefumi GOTO†, Taiki MATSUMOTO†, Yuki TAKAHASHI†, Michiharu TSUKAMOTO† , Akihito NAKAMURA‡
†Tokyo University of Technology, ‡National Institute of Advanced Industrial Science and Technology (AIST)

**(4-A) バージョンアップ**
(2)で求めたリストをもとに必要なソフトウェアをシステムに追加し、バージョンアップを実行する。
**(4-B) バージョンダウン**
(2)で求めたリストをもとに妨げとなるソフトウェアをシステムから削除し、バージョンダウンを行う。
**(5) 設定の適用（状態の構築）**
バージョンアップ/ダウン後のシステムに対して、適切な設定を生成・適用する。
**(6) テスト**
正常な構成管理が行われたことを確認するためのテストを行う。
**(7) システム構成管理の完了**
構成管理が正常に行われたとき、システム構成管理は完了する。

# 4 実現方法の検討

## 4.1 バージョンアップ/ダウンの実現

ソフトウェアの依存関係とシステムへの影響範囲を考慮した構成管理を行うために、ソフトウェアに関する詳細情報（依存関係、配置場所、設定ファイル所在）が必要である。その手段として、ソフトウェアに対して強力な問い合わせ機能を持つ RPM と APT(Advanced Package Tool)[2]を利用する。

- **バージョンアップ**
APT が RPM パッケージに対して依存関係の問い合わせを行い、システムに存在しないソフトウェアを必要とする場合は自動的にダウンロードしてインストールする。
- **バージョンダウン**
現バージョンのソフトウェアを APT でアンインストールし、指定されたバージョンの RPM パッケージをインストールする。

## 4.2 Subversion による構成履歴管理

構成管理によるシステム構成の状態変化を履歴として蓄積・管理するために、履歴管理システムを利用する。広く利用されているものに CVS(Concurrent Version System)と Subversion がある。ここでは、HTTP プロトコルでの通信が可能であり、履歴ごとに任意のメタ情報を付加することができる Subversion を利用することにした。履歴管理の対象となるリソースは、構成管理で影響が及ぶ範囲のソフトウェアの設定ファイル群である。それらのリソースには関係するソフトウェアの名前とバージョンの情報をメタ情報として付加し、Subversion のリポジトリにチェックインする。

## 4.3 XML を用いた設定ファイル管理

ソフトウェアの設定ファイルは、ソフトウェアごとに固有の文法規則で記述されている。たとえば Apache は、変数名に対して唯一の値を設定する記述パターンと、タグを用いた記述パターンが混在した文法規則で記述されている。MySQL や Samba では変数設定にグループの概念を導入している。また、Tomcat では XML 記述形式が採用されている。一方、ソフトウェアのバージョンによって設定できる項目が異なる場合もある。こうした文法規則の違いを吸収し、構成管理のたびに行われる設定を書き直す作業を効率化するために、各ソフトウェアの設定ファイルを XML の統一的な記述形式に変換し、管理する手法をとる（図 2）。

- **設定ファイルの XML 形式への変換**
設定ファイルを XML に変換するための変換器が必要である。変換器を作成する手段として、定義した文法規則から文字列解析プログラムを生成するコンパイラ・コンパイラを利用する。設定の文法規則はソフトウェアごとに異なるため、それぞれの設定の文法規則を調査して定義し、コンパイラ・コンパイラにより生成された変換器を利用して設定ファイルを XML 形式に変換する。一方、XML 形式から設定ファイルへの変換は XSLT 変換によって行う。
- **設定ファイルの文法規則の検証**
設定の中には、定められた候補の中から値を選択することを求めるものもある。XML では、そうした文法規則を XMLSchema で定義することで、文法の検証を行うことが可能である。そこで XML 化した設定ファイルの文法を XMLSchema で定義し、単一の設定ファイル内で値が正しく設定されているか検証を行う。
- **設定ファイル間の依存関係の検証**
ソフトウェア間、あるいはソフトウェア-システム間、システム-システム間で設定ファイルの依存関係が存在する。例えば、Apache と Tomcat を連携させて Web アプリケーションを動作させる際には、Apache の httpd.conf で Alias および Location の設定を行い、Tomcat の server.xml で Context の追加を行う。このとき、Apache 側で設定するディレクトリパスと URI が、Tomcat 側の設定と一致している必要がある。こうした設定ファイル間の関係を XML で記述して、設定が正しく行われているかを論理的に検証する。

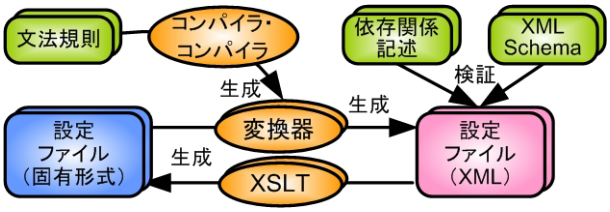

図 2 ：設定ファイル管理の関係

# 5 今後の課題

本稿で述べてきたシステム構成管理を実現するために、まず単一のサーバを対象とし、ランタイムシステムの構築やデーモンの作成といった実装を行う。その後対象を、ネットワークを介した PC クラスタ[3]へと広げ、分散環境あるいは並列マシンでの統一的なシステム構成管理を実現する。また、設定ファイル管理で扱えるソフトウェアの範囲を広げ、ソフトウェアの設定の体系化ならびに統一的管理を実現したい。

# 参考文献

[1] 中村章人,他:XML と SOAP によるセキュリティ関連情報 Web サービス,第 65 回全国大会論文集,2003
[2] APT:http://www.debian.org/doc/manuals/apt-howto/ch-novas.html
[3] 後藤滋文,他:chroot を用いたセキュアな Linux サーバシステムの構築,第 66 回全国大会論文集,2004