変更履歴を利用したパッケージ管理のためのバージョン管理システム

藤井 亮太　岩崎 英哉

パッケージとは，ソフトウェアを構成する複数のファイルを一つにまとめたものであり，ソフトウェアの管理単位として広く用いられている．パッケージのユーザは，自分の動作環境に合わせるために，インストールしたパッケージに対して変更を加えることがある．しかし，一般的なパッケージ管理システムは，ユーザによる変更を把握していないため，新しいバージョンのパッケージにユーザの変更をマージしたり，ユーザによる変更を別の環境で再現したりすることができないという問題がある．本研究は，バージョン管理システムと既存のパッケージ管理システムを連携させることで，上の問題点を解決する．提案するシステムは，パッケージに対する変更の履歴をパッケージの情報と対応付けて管理し，パッケージ管理システムから利用できるようにする．これにより，ユーザがパッケージに対して加えた変更を，通常のパッケージ管理の枠組の中で扱うことを可能にする．

1　はじめに

ソフトウェアの配布やコンピュータ上での管理を行う際に，ソフトウェアをパッケージという単位で扱うことが広く行われている．例えば，多くの Linux ディストリビューションでは，Linux カーネルからアプリケーションに至るまで，様々なソフトウェアがパッケージの形で提供されている．パッケージは，プログラム本体とそのドキュメントなど，ソフトウェアを構成する複数のファイルを一つのファイルにまとめた形で提供され，パッケージ管理システムと呼ばれる専用のツールを用いることで，ユーザのシステムにインストールして利用できる．パッケージには，内容物であるファイルに加えて，パッケージの名称やバージョン，パッケージ間の依存関係などのメタデータが含まれている．パッケージ管理システムは，このメタデータを利用してパッケージを適切に管理する．

パッケージに含まれるファイルの中には，設定ファイルのように，インストール後にユーザが自分の環境や利用目的に合わせた変更を加えて使用することが想定されるものがある．このような変更を加えることで，ユーザごとの環境や要求に合わせてパッケージを利用することが可能になる．その反面，設定ミスによってパッケージが正常に機能しなくなるなど，ユーザの変更に起因する問題も生じうる．さらに，パッケージを更新したり，環境の再構築を行う際には，ユーザによる変更が失われないように配慮する必要がある．そのため，パッケージに対して加えた変更を把握し，適切に管理することは，パッケージを利用する上で重要である．

しかし，一般的なパッケージ管理システムは，インストールされたパッケージに対するユーザによる変更を把握しておらず，その管理にも関与しない．そのため，パッケージに対して加えた変更を何らかの形で記憶ないし記録し，その情報を必要に応じて利用することは，ユーザの責任において行わなければならない．このことは，定期的にパッケージを更新しながら継続的にパッケージを利用する上で，ユーザの負担となる．

このような問題に対して，etckeeper [1] や Conary [2] といった，バージョン管理システム (VCS) とパッ

Ryota Fujii, Hideya Iwasaki, 電気通信大学大学院情報理工学研究科, Department of Communication Engineering and Informatics, The University of Electro-Communications.

ケージ管理システムを組み合わせることで，ユーザによる変更の履歴を管理できるようにするツールが存在する．しかし，これらのツールには，VCS によって記録した変更履歴をパッケージ管理に利用するための機能が不足していたり，他のパッケージ管理システムと共存できないなど，機能性や利用可能な環境に関する問題がある．

このような問題点を解決するため，本研究では，既存のパッケージ管理システムと連携しながら，ユーザがパッケージに対して加えた変更の履歴を管理し，利用することを支援する VCS を提案する．既存のツールと比較した際の本システムの特長は，変更履歴の記録と利用の両面においてユーザを支援する点，さらにこれを既存のパッケージ管理の枠組を変えることなく実現する点である．

本稿の構成は次の通りである．2 節では，パッケージに対する変更と，変更履歴の必要性について述べる．3 節では，既存のツールの特徴と問題点を述べる．4 節では，本研究で提案するシステムの概要と動作の流れについて述べ，5 節ではシステムの実装について述べる．最後に 6 節で本稿をまとめ，今後の課題を述べる．

2　パッケージに対する変更

2.1　ユーザによる変更

ユーザによる変更の最も典型的な例は，設定ファイルの書き換えである．図 1 に，プログラム app とその設定ファイル app.conf からなるパッケージ app を例として，パッケージに対する変更の様子を示す．

パッケージに含まれる設定ファイル (図 1 の (a)) はパッケージ提供者が作成した標準的な雛形であり，そのままでユーザの要求を満たすとは限らない．そのため，ユーザはパッケージをインストールした後，設定ファイルを自分の環境や要求に合うように編集して利用する (図 1 の (a'))．

2.2　変更履歴の必要性

設定変更により，パッケージをユーザごとの環境や要求に合わせて様々に利用することが可能となる．しかし，ユーザによる誤った設定変更は，システム障害

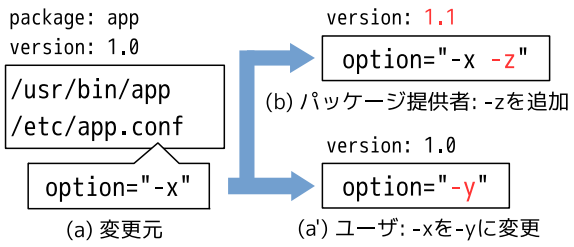

図 1　パッケージに対する変更

の主要な原因であることが知られている [3]．誤った設定変更による障害への対処の一つとして，問題が発生する前の設定内容を復元することが挙げられるが，通常，設定変更は設定ファイルを上書きする形で行われるため，どのような変更を行ったか記憶しているか，コピーを残すなどして古い内容を保存していない限り，過去の設定を復元することはできない．

また，OS の再インストールなどで一度構築した環境を構築し直す場合は，パッケージをインストールし直した上で，過去にパッケージに対して加えた変更を再度行う必要がある．しかし，パッケージに対する変更内容をユーザが把握していない場合，過去に行った変更を再現することはできない．

このように，パッケージを正しく動作させるためには，そのパッケージに対する変更内容を常に把握しておく必要がある．また，その際には，

- 変更は複数回行われる可能性があり，個々の変更は必ずしも密接に関連しているとは限らない;
- 特定の変更に問題がある場合，それだけを取り消すことが可能になっていることが望ましい;

といった理由から，単にパッケージをインストールした直後の未変更のファイルだけを残すのでは，複数回の変更のうち，特定の一つだけの内容を確認したり，取り消すことが難しいため，不十分である．そのため，個々の変更を行った時点での状態を逐一記録することで，変更の履歴が分かるようにしておくことが必要である．

2.3　パッケージ提供者による変更

ユーザがパッケージをインストールした後に独自の変更を行う一方で，パッケージの提供者は，古いバー

ジョンのパッケージに対して変更を加えることで新しいバージョンのパッケージを作成する (図 1 の (b)).

ソフトウェアのバージョンアップに伴い，設定項目の追加，削除やデフォルトの設定値の変更などが頻繁に行われる [4]．そのため，図 1 に示した，ユーザの環境 (ユーザによる変更) とパッケージ本体 (パッケージ提供者による変更) の双方で変更が行われるような状況は発生しやすい．

2.4 変更のマージ

ユーザがパッケージを更新する際に，新バージョンのパッケージに含まれるファイルは，旧バージョンのファイルを上書きする形でインストールされる．しかし，図 1 の `app.conf` のようにユーザが変更を加えたファイルについては，そのまま上書きしてしまうと，ユーザによる変更が失われてしまう．ユーザによる変更を保持しつつ，パッケージの新バージョンにおける変更を取り込むためには，新旧両バージョンの 2 つのファイル (図 1 の (a') と (b)) を適切にマージする必要がある．

ユーザによる変更と，パッケージの側の変更とは，互いに相反する可能性がある．また相反しない場合であっても，パッケージ提供者による変更がユーザの意図に反している可能性がある．従って，マージ作業にはユーザ自身による判断が不可欠である．

多くのパッケージ管理システムは，更新対象のファイルに対してユーザが変更を加えていた場合，旧バージョンのファイル (図 1 の (a')) を残したまま新バージョンのファイル (図 1 の (b)) を別名で保存し，対処をユーザに委ねる．しかし，これら 2 つのファイルを比較しても，どの部分をユーザが変更し，どの部分をパッケージ提供者が変更したのかが分からなければ，どのようにマージすればよいか判断することが難しい．

ユーザによる変更とパッケージ提供者による変更を区別するためには，ユーザの環境とパッケージそれぞれにおける変更内容 (図 1 における (a) と (a') の差分および (a) と (b) の差分) を把握する必要がある．

3 既存のツール

3.1 etckeeper

etckeeper [1] は，Git[†1] などの既存の VCS を用いた設定ファイルのバージョン管理を支援するツールである．図 2 のように，パッケージのインストールや更新を行うたびにパッケージ管理システムによって起動され，VCS を起動して設定ファイルの変更を記録する．

パッケージ管理システムに連動した動作の実現には，多くのパッケージ管理システムに備わっているフック機能が用いられている．これは，パッケージ管理システムの動作中，パッケージのインストール前後などの特定の時点で，任意のプログラムを起動することのできる機能である．etckeeper 自体は特定のパッケージ管理システムに依存しておらず，パッケージ管理システムに応じてフックを設定することで，複数のパッケージ管理システムに対応することが可能となっている．

etckeeper の問題点は，記録した変更履歴の利用には一切関与しない点である．変更履歴を利用する際，ユーザは etckeeper が呼び出している VCS を直接使用する必要がある．そのため，

- 特定のパッケージに関する変更履歴だけを取り出す;
- 変更履歴を利用して，2.4 項で述べたマージ作業を行う;

といったような作業については，全てをユーザが行わなければならない．

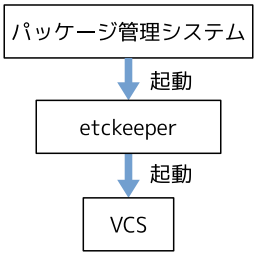

図 2 etckeeper の動作の流れ

†1 `https://git-scm.com`

3.2 Conary

Conary [2] は，VCS が統合されたパッケージ管理システムである．Conary におけるパッケージは全てバージョン管理されており，ユーザは，パッケージに対して自らが行った変更を記録することが可能である．また，パッケージを更新する際には，VCS の機能を用いてユーザが行った変更とパッケージの側の変更とをマージすることが可能であるなど，変更履歴をパッケージ管理に利用することが可能である．

Conary は，VCS とパッケージ管理システムを密に結合させることで，変更履歴の記録と利用をパッケージ管理に統合している．しかし，Conary がパッケージ管理システムであるということは，他のパッケージ管理システムとは共存できないということでもある．OS 標準のパッケージ管理システムが存在する場合，これを Conary に置き換えることは，対応しているパッケージ形式の違いや，OS 開発元によるサポートの有無といった点が問題となり，現実的ではない．そのため，Conary が利用できる環境は，初めから Conary をパッケージ管理システムとして採用しているシステムに限定される．

4 提案システム

4.1 設計上の要件

本研究では，Conary のように変更履歴の記録と利用をパッケージ管理に統合しつつ，etckeeper のように様々な環境で利用することが可能なシステムを提案する．提案システムに要求される機能は以下の通りである．

- ユーザがパッケージに対して加えた変更の履歴を記録する．
- 変更履歴をユーザが利用することを助ける．

後者は，etckeeper に不足していて，Conary には備わっている機能である．

また，ユーザの負担を軽減するために，システムを利用する上でユーザが新たに行わなければならない作業は，可能な限り少なくすべきである．特に，通常のパッケージ管理に付随して行うべき作業については，ユーザが明示的に実行するのではなく，パッケージの操作に統合された形で自動的に行われることが望ましい．

変更履歴の記録と利用をパッケージ管理に統合する上で，Conary のようにパッケージ管理システムを置き換えて固定する方法は，利用可能な環境が限定されてしまうため，望ましくない．また，パッケージ管理システムを置き換えないとしても，通常のパッケージ管理の手順を変えてしまうと，ユーザの学習コストが大きくなってしまう．そのため，パッケージ管理への統合は，既存のパッケージ管理システムを置き換えず，また可能な限り既存のパッケージ管理システムのユーザインタフェースを維持した形で実現すべきである．

4.2 システムの概要

本研究で提案するシステムは，etckeeper や Conary と同様に，ユーザがパッケージに対して加えた変更を管理する VCS であり，ユーザが変更履歴を記録したり，記録した変更履歴を取り出すためのユーザインタフェースを提供する．この VCS は，変更対象のファイルに加えてパッケージのメタデータを記録しており，名前やバージョンなどのパッケージに関する情報を元に，特定のパッケージに関する変更履歴を取り出すことが可能である．

また，提案する VCS は，既存のパッケージ管理システムと連動して動作することで，変更履歴の記録と利用を通常のパッケージ管理に統合する．例えば，ユーザはパッケージに対して変更を加えるにあたって，変更前の状態を VCS に記録しておく必要があるが，設定ファイルのようにユーザが変更を加える可能性が高いものについては，パッケージをインストールした時点で自動的に記録しておくことで，ユーザの行う作業を削減できる．また，2.4 項で述べたマージ作業は，パッケージを更新した直後に行うことが望ましい．このような，通常のパッケージ管理作業に付随して行うべき作業は，パッケージ管理システムの動作に連動する形で，可能な限り自動的に実行する．

以上のように，提案する VCS は，変更履歴の記録と利用の両面においてユーザを支援する．また，既存のパッケージ管理システムに連動して動作することにより，システムを利用する上でのユーザの負担を低減

する．

本研究で提案するシステムは，パッケージ管理システムに連動して動作するものであり，既存のパッケージ管理システムを置き換えるものではない．また，本システムを導入することで，通常のパッケージ管理操作の手順が変わることもない．ユーザはシステムの導入前と同様に，使い慣れたパッケージ管理システムのユーザインタフェースを利用して，パッケージ管理を行うことが可能である．

4.3 システムの動作の流れ

図 3 を用いて，2.4 項で述べたパッケージに対する変更のマージを例に，パッケージをインストールしてからマージ作業を行うまでのシステムの動作の流れを説明する．

4.3.1 パッケージを初めてインストールしたとき

ユーザがパッケージ管理システムを用いてパッケージをインストールすると，この操作に連動して VCS が起動し，パッケージのメタデータ M1 と，パッケージに含まれる設定ファイルの内容 C1 を記録する．このとき，M1 と C1 の対応関係を記録することで，ファイルの内容がパッケージのどのバージョンに基づくものであるか判別できるようにする．

4.3.2 ユーザがパッケージに変更を加えたとき

ユーザは，パッケージをインストールした後，設定ファイルを自分の要求に合わせて編集する．その後，VCS が提供するユーザインタフェースを用いて，変更後のファイル内容 C1’ を記録する．このとき，C1’ は VCS によって変更元のパッケージのメタデータ M1 と対応付けられる．

4.3.3 パッケージを更新したとき

新しいバージョンのパッケージが公開され，ユーザがパッケージ管理システムを用いてパッケージの更新を行うと，パッケージをインストールした時と同様に VCS が起動し，新しいバージョンのパッケージのメタデータ M2 と新しいバージョンのパッケージに含まれる設定ファイルの内容 C2，および M2 と C2 の対応関係を VCS に記録する．

ここで，古いバージョンのパッケージのメタデータ M1 に対して C1 と C1’ という複数のファイル内容

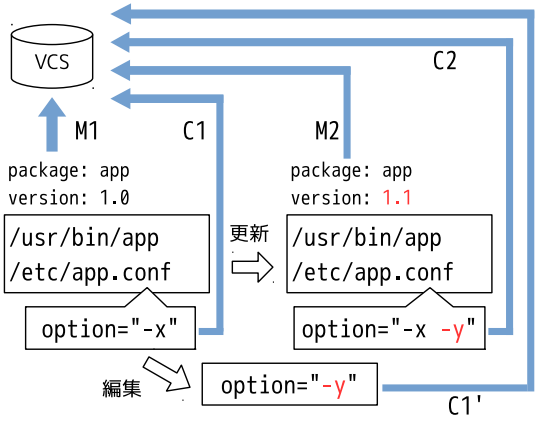

図 3 システムの動作の流れ

が対応付けられていることから，ユーザがパッケージに対して変更を加えたことがわかる．また，M1 に最初に対応付けられた C1 と，新しいパッケージのメタデータ M2 に対応付けられた C2 を比較することで，パッケージの側でも変更が行われたことがわかる．このように，ユーザの環境とパッケージの双方で変更が行われたことが判明した場合，VCS はこれらの変更をマージする必要があると判断し，ユーザに対してマージを行うよう要求する．

このマージの必要性の判定は自動的に行われるため，ユーザは確実にマージの必要性を認識することができる．また，C1 と C1’ を比較することでユーザ自身の変更による差分が得られ，C1 と C2 を比較することでパッケージの側の差分が得られるので，これらをユーザに提示することで，ユーザは自分が行った変更の内容と，パッケージ提供者が行った変更の内容を明確に区別して認識することができる．これは，ユーザによるマージ作業の助けとなる．

5 実装

5.1 VCS の実装

VCS の実装は，Git をベースに，独自のユーザインタフェースを実装する形で行っている．ただし，Git そのものを拡張したり，あるいは外部プログラムとして呼び出すのではなく，Git と同等の機能を提供する

ライブラリである libgit2[†2] を用いることで，Git 形式のリポジトリを操作している．これにより，本システムは，Git が導入されていない環境でも利用可能である．

4.2 項や 4.3 項で述べた通り，VCS は変更対象のファイルだけでなく，パッケージのメタデータを記録する．これには，

- パッケージの名称
- パッケージのバージョン
- パッケージに含まれるファイルの一覧

が含まれる．ここで，ファイル一覧に含まれるそれぞれのファイルは，Git リポジトリに記録されている具体的なファイルの内容と対応付けられている．パッケージの情報をもとに変更履歴を取り出す際は，この対応関係を利用する．

以上のメタデータは，通常のファイルとして Git リポジトリ上の特別なディレクトリに格納する．このとき，パッケージのメタデータの形式はパッケージの形式ごとに異なるので，特定のパッケージ形式に依存しない形に変換してからリポジトリに格納する．この変換処理のようなパッケージ管理システムに依存する処理は，VCS 本体から呼び出される外部プログラムとして，パッケージ管理システムごとに実装する．これにより，複数のパッケージ管理システムに対応することが可能である．

現在行っている実装では，パッケージ管理システムとして，Debian GNU/Linux 等で用いられている Apt[†3] を対象にしている．Apt は，より下位のパッケージ管理システムである dpkg に対するフロントエンドであり，パッケージファイルを直接操作する処理は，dpkg によって実現されている．そのため，パッケージのメタデータを取得する際は，dpkg を外部プログラムとして呼び出す．

5.2 パッケージ管理への統合

4.2 項や 4.3 項で述べたように，本システムの VCS は，パッケージ管理システムに連動して動作する．これには，etckeeper と同様に，パッケージ管理システムのフック機能を利用する．

6 おわりに

本稿は，パッケージに対してユーザが加える変更に起因する問題を解決するために，ユーザによる変更の履歴を管理し，パッケージ管理に利用することを支援するシステムを提案した．提案したシステムは，既存のツールにおける機能性や利用可能な環境の問題を考慮した上で設計されており，変更履歴の記録と利用の両面でユーザを支援することを，既存のパッケージ管理の枠組を変えることなく実現している．

現状は，提案したシステムの実装を行っている途上であり，システムの実装を完了し，実用性を検証することが目下の課題である．また，現在は特定のパッケージ管理システム (Apt) を対象として実装を行っているが，これが完了し次第，他のパッケージ管理システムへの対応作業を行うことを通して，設計上の問題が存在しないかどうか確認することも必要である．

参 考 文 献

[1] Hess, J.: etckeeper, `http://etckeeper.branchable.com/`.

[2] Johnson, M. K., Troan, E. W., and Wilson, M. S.: Repository-based System Management Using Conary, *Proceedings of the Ottawa Linux Symposium*, Vol. 2, 2004, pp. 557–571.

[3] Nagaraja, K., Oliveira, F., Bianchini, R., Martin, R. P., and Nguyen, T. D.: Understanding and Dealing with Operator Mistakes in Internet Services, *Proceedings of the 6th Conference on Symposium on Opearting Systems Design & Implementation - Volume 6*, OSDI'04, USENIX Association, 2004, pp. 5–5.

[4] Zhang, S. and Ernst, M. D.: Which Configuration Option Should I Change?, *Proceedings of the 36th International Conference on Software Engineering*, ICSE 2014, ACM, 2014, pp. 152–163.

†2 `https://libgit2.github.com/`

†3 `https://wiki.debian.org/Apt`