3W-02 階層的モジュール管理に基づく ソフトウェアリポジトリにおける変更伝搬制御

二木 邦尚　織田 健

電気通信大学大学院電気通信学研究科電子情報学専攻

1 はじめに

近年、ネットワークを介したソフトウェアの共同開発や再利用が盛んになっている。そのため、ソフトウェアを管理し開発者間で共有させる開発支援環境が求められ、さまざまな研究がなされているが、再利用の対象がある程度まとまった大きさであったり、1組織で閉じたものになっている [1]。

我々は、様々な大きさのソフトウェア部品をモジュールとして管理し、個人から多組織にまたがるモジュールの共有をサポートするソフトウェアリポジトリ [2] の研究を進めている。

本稿ではモジュール間で変更を伝搬させ、品質を高める変更伝搬機構について述べる。

2 階層的モジュール管理

我々の提案するリポジトリは、ソフトウェア部品を表すためのモジュールと、リポジトリ利用者であるグループ、そしてグループのモジュールに対するアクセス権等といったソフトウェア開発に固有の情報を管理する情報システムである。

グループは1人以上のリポジトリ利用者と1人の管理者から構成され、利用者は複数のグループに所属できる。グループは階層的に構成することができ、個人から多組織までを単一の概念で扱える。モジュールの属性には、版番号、所有するグループ、使用するグループがある。所有グループの管理者がモジュール管理者になる。

このリポジトリではソフトウェアの構成要素を木構造に分解したときの任意の部分木をモジュールとして捉え管理することで、様々な大きさの部品管理を行う。この木構造の親子関係をモジュールの使用関係と捉え、モジュールを任意の節点であると考えると、あるモジュールをその上位モジュールが使用していると見ることが出来る。そのようなモジュール構造では、あるモジュールに変更を加えて出来た新しいモジュールを上位モジュールに使わせることが可能で、その結果、新モジュールを使用した全てのモジュールの品質の早期向上が期待できる。さらに、木構造をグラフに一般化することでより多彩な部品構造を表現できるようになる。よって本稿ではモジュールを「使用関係を辺に持つ有向グラフにおける任意の節点」と定義する。

Update Propagation on Hierarchical Module Management Based Software Repository
Kunitaka Futatsugi, Takeshi Oda
Course in Communications and Systems, Graduate School of Electro-Communications

3 変更伝搬機構

変更伝搬は、モジュールが変更する以前と以後の版を更新前モジュール、更新後モジュールと呼ぶ時、登録により生じる更新後モジュールの存在を、更新前モジュールの利用者に通知するとともに、更新後モジュールを更新前モジュールを使用していた上位モジュールに使用させ、新しい上位の版を発生させることである。

3.1 伝搬条件

変更伝搬はその実行後に、リポジトリの保全性を保つため、以下の2つの条件を満たす必要がある。

- 更新前モジュールの上位モジュールの所有者が更新後モジュールに読み込みアクセス権を持つ。
- 更新前モジュールをたどることができる全てのモジュールが更新後モジュールを使用する。

3.2 伝搬の形態

変更伝搬作業を1つ1つ人の手によって行うのは大規模なリポジトリにおいて現実的ではない。そのため自動化が求められるが、更新モジュール採用の際、使う側に変更が必要であったり、採用に開発者の判断が必要である場合は人の介入が不可欠になる。以下、変更伝搬の形態を自動と手動に分けて説明する。

自動変更伝搬では、モジュールの更新によって更新前モジュールを利用する全てのモジュールに更新後モジュールを使用した新たな版が作成されるとともに、更新前モジュールの利用者に更新後モジュールの利用を促す伝搬通知が送られる。新しい版の上位モジュールにも同様に変更伝搬が発生し、最上位モジュールまで再帰的に変更伝搬が繰り返される。

手動変更伝搬では、モジュールの更新が起こると更新前モジュールの上位モジュールの管理者に対して変更情報を含んだ更新通知がされ、更新モジュールの採用が検討される。採用が決定した場合は、所有者がモ

表 1: 変更の種類と伝搬形態

更新モジュールに加えられた変更	手動	自動
環境変化への対応	○	
変更情報が必要な改良	○	
変更情報が必要でない改良		○
修正		○

ジュールに手を加えてリポジトリに登録し、新しい版が発生する。

3.3 伝搬形態の選択

変更伝搬形態は、更新モジュールを採用するモジュールの加工及び採用時のユーザ判断の必要の有無で決まる。モジュールに加えられる変更から伝搬形態を選択するために、「修正」「環境変化への対応」「改良」での伝搬動作について考える。

修正作業で行われるのは誤りの修正であり、一般には仕様の変更は生じない。そのため使用する側に変更の必要がない。環境変化に対応するために更新されたモジュールを使用するには、使用する側もその環境に対応するための変更が必要になる。改良のうち、速度や保守容易性の向上等では、その更新モジュールを利用するだけで効果が得られるため、利用する側に変更の必要がない。機能の追加や拡張等では、変更情報を得てその効果が得られるように利用する側のモジュールに変更が必要になる。また、改良した結果ある部分においては前の版より劣ってしまう場合がある。そのときは変更情報から、採用を判断することになる。

以上より、モジュールに与えられた変更が自動・手動どちらの変更伝搬を生じさせるかは表 1 により示される。変更の種類は複数設定可能なものとし、伝搬形態の決定は、表 1 の上から順に調べ、該当する行があった時点で決まる。ただし、モジュールのインタフェースが変更された場合はどの変更であってもが加工が必要であり、伝搬は手動になる。

3.4 通知増加対策

変更伝搬では新しいモジュールの発生や、モジュールの更新がモジュール利用者に通知されるが、大規模なリポジトリでは通知が大量になる恐れがある。これを避けるために、必要性が少ない通知を無くす必要がある。

自動伝搬での新しい版の利用を促す伝搬通知は、上位に向かって次々と作られる更新モジュールごとに発生する。このとき多くの下位モジュールを持つ上位モジュールほど伝搬の回数が増大し、送られる伝搬通知も莫大なものになってしまう。自動伝搬では取り込むべき更新モジュールがあることを知るだけで良く、どのモジュールから伝搬が発生したかは問題ではないため、個々の版全ての通知をする必要はない。よって、更新が発生し通知がされてから利用者が取り込むまでは新たな伝搬通知を抑制する。

また、手動伝搬の更新通知を伝搬元と伝搬先のモジュール管理者が同じときに行う必要はなく、更新通知をそのモジュールの管理者以外にする。

4 考察

変更伝搬の形態を区別するためにモジュールに与えられる変更の種類を調べ、更新モジュールを利用して無条件で良い効果が得られるときの変更を識別した。これは、変更伝搬自動化への前進になった。

しかし、変更の種類を選んでいるのは開発者で、それが正しい選択だという保証はない。また、1 度の更新に変更の種類が複数ある場合、優先順位の高い変更に低い変更が取り込まれてしまう。これらの問題は開発者の判断に依存しており、リポジトリ側で対処するのは困難であると思われる。また、正しく変更の種類を選択したとしても変更内容に誤りがある場合があり、変更伝搬で次々と作られた版から誤りを見つけ修正する必要が生じてくる。版が多くなるほどその作業は困難になるため、登録時に自動でモジュールテストを行うような措置が必要になると思われる。

変更の種類指定の意味として伝搬形態の区別の他に変更の性質ごとに版管理を行えることがある。それにより、ある環境に対応したモジュールの最新版の取り出しや、版番号から変更の推移の理解等が可能になる。

5 おわりに

本稿では、モジュール間に依存関係が存在するリポジトリにおいて変更伝搬動作を自動・手動の 2 つの形態に分けて考え、変更の種類によって伝搬形態を区別し、ある種の変更において更新モジュール採用の判断と新しい版の生成を自動化する変更伝搬機構を提案した。

今後は変更の性質に基づく版管理機構を含んだ変更伝搬システムの詳細設計及び実装を行い、その後適用実験を行う必要がある。

参考文献

[1] 青山 幹雄 “分散開発環境：新しい開発環境像を求めて” 情報処理, Vol.33, No.1 pp.2-13 (1992)

[2] 雲切 啓太, 織田 健 “ビュー提示機構を導入した広域型ソフトウェアリポジトリ” ソフトウェア科学会第 15 回大会論文集, pp105-108 (1999)