実身／仮身モデルを用いたバージョン管理・構成管理

3L-7

高田 広章　　坂村 健
（東京大学　理学部）

１．はじめに

TRON プロジェクトにおける TIPEシステム（TRON Integrated Programming Environment）の一部である，実身/仮身モデルを用いたバージョン管理・構成管理に関する研究について述べる．TIPEシステムはBTRONの環境において，大規模なシステムの多人数による分散開発にたえるソフトウェア開発環境の構築をめざすものである．

２．実身/仮身モデル

実身/仮身モデルとはBTRONの統合操作環境における情報の蓄積/表現/管理のためのモデルであり，ハイパーテキストに基づいた一種のファイルシステムといえる[1]．情報のまとまりのことを実身と呼ぶ（従来のＯＳのファイルに相当する．以下ファイルという言葉を実身の意味で用いる）．仮身は実身を参照するためのタグであり，文字や図形と同様に実身の中に埋め込むことができる．同じ実身を指す仮身はいくつあってもよく，実身と仮身がなす構造は一般にネットワーク構造になる（図１）．

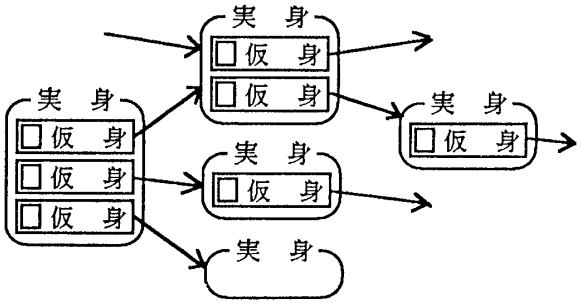


図１．実身/仮身モデル

３．既存のバージョン管理・構成管理の問題点

大規模なソフトウェアの開発時におけるバージョン管理・構成管理の必要性は広く認められており，多くのツールないしはシステムが作られてきた．UNIXのSCCSやMAKEといった比較的単純なツールに始まり，現在ではほとんどのソフトウェア開発環境において，不可欠な要素としてサポートされている[2][3]．

ここでは，これらのツール，システムの多くに共通する問題点として次の２点を指摘し，それらに対する１つの解答を示す．

(a) 構成記述に含まれる各コンポーネントに対するバージョン情報を，構成記述とは別に管理するというように，構成情報をバージョンによらない静的なものとしているシステムが多く，バージョン改訂に伴う構成の変更についての扱いが不明確．

(b) 概念が複雑または冗長なシステムが多く，強力な記述力とひきかえにわかり易さを失っている．例えば，構成記述が作る階層構造と，階層ディレクトリの構造とが重複した情報を持っている．

４．実身/仮身モデルを用いたバージョン管理・構成管理

４．１　バージョン管理

本システムでは，個々のファイルのバージョン管理は次のように行なわれる．すなわち，あるファイルの各バージョンを，１つのファイル（これを管理ファイルと呼ぶ）中に差分を使うことにより効率良く保存し，必要なときにバージョンを指定して元のファイルを再構成し，読み出す．

ファイルの内容を修正したい時は，管理ファイルから元となるバージョンを修正モードで取り出す（リザーブ）．この時，管理ファイルに作成中のバージョン番号が記録され，同じバージョンをリザーブすることを禁止する．修正作業が終われば，新しくできたバージョンを管理ファイルに登録する（デポジット）．この際に新バージョンの作成理由・修正点を記述したテキストも一緒に登録する．新バージョンに関するより詳細な情報は別の実身に書き，登録するテキスト中にはその実身を指す仮身をいれる．

バージョン管理されているファイルをバージョンの指定なしでオープンすると，管理されているバージョンの一覧と各バージョンの作成理由・修正点がウィンドウに表示される．この一覧表から必要なバージョンの読み出しやリザーブができる．また，バージョン番号をあたえてオープンすることにより，指定したバージョンの内容がウィンドウに開かれる．この際，バージョン管理されているファイルを指す仮身中に，読み出したいバージョン番号をバージョン管理固有のデータとして入れておくことにより，ユーザにはバージョン管理されていることを意識させないことも可能である．

Version and configuration managemanet on the basis of real object/virtual object model

Hiroaki TAKADA and Ken SAKAMURA
The Univ. of Tokyo

４．２　構成管理

大規模なシステムを開発/管理する際は通常，階層的なモジュール分けが行なわれる．この構造を最も自然に管理するために，モジュールの階層構造の中間ノードに構成ファイルを置く．構成ファイルには，そのなかに含まれる各コンポーネント（下位の構成ファイルまたはソースファイル）のバージョン番号やコンパイル/リンクオプション，ソースファイル間の依存関係（ソースファイルから自動判別される場合は必要ない）が書かれる．実身/仮身モデルを用いて構成ファイルを作ると，各コンポーネントを指す仮身が構成ファイル中に含まれることになり，構成ファイルが自然にディレクトリの役割を兼ねていることになる．

構成ファイルには各コンポーネントのバージョン番号が含まれているため，コンポーネントのうちのどれかが修正されるたびに構成ファイルも修正されることになる．そのため，構成ファイル自身も通常のソースファイルと同様にバージョン管理される．このように，構成ファイル自身もバージョン管理されるという考え方をとることにより，バージョンアップにともなう構成の変更が柔軟に記述できる．

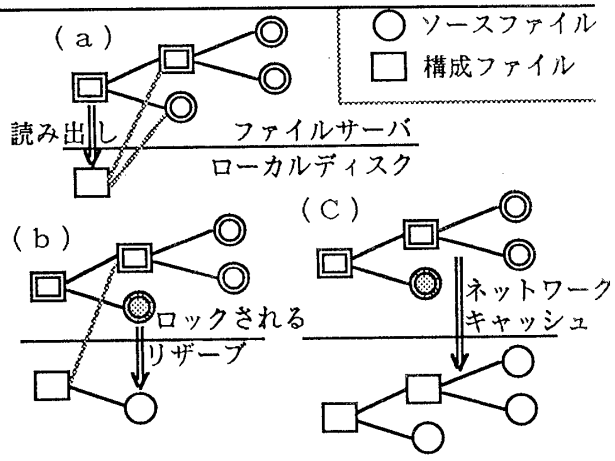
各コンポーネントのバージョン番号の記述には，具体的な番号で指定する以外に，あるバージョン系列の中で最新のものを指定したり，上位の構成ファイルでの記述に従うという指定も可能である．

４．３　多人数による分散開発

この節では，TIPEプロジェクトの目標である多人数による分散環境での開発支援が，バージョン管理・構成管理の中でどのように実現されているかについて述べる．

分散開発環境においては，開発中のシステムを構築する全ファイルは，ネットワーク上のファイルサーバ中に（ワークステーションのうちの１台がサーバを兼ねることも可能）バージョン管理されて置かれている．システム中のあるソースファイルの開発/保守を担当しているプログラマは，まずファイルサーバ中にある必要な構成ファイルをローカルなディスクに読み出す（図２ａ）．次に修正したいソースファイルをリザーブする（図２ｂ）．リザーブされたソースファイルはローカルディスクに取り出され，その上で修正を行なう．コンパイルなどの際に読み出されたファイルは，ローカルディスクにキャッシュされる（図２ｃ）．

ここで，関連する構成ファイルをリザーブせずに単に読み出したのは，次のような理由による．構成ファイルは，その下のすべてのコンポーネントがコンシステントにバージョンアップされ，なんらかの検証をした後に新バージョンを登録すべきもので，その作業は構成ファイル管理者によって行なわれるべきである．そのため，各ソースファイルの作成者または下位の構成ファイルの管理者は，新バージョンをデポジットした時点で，そのコンポーネントを直接含む構成ファイルの管理者に電子メイルで知らせなければならない（この作業は，半自動化される）．さらに大規模なシステムの開発のために，バージョン管理データベースの階層的な管理も可能である．



※二重枠はバージョン管理されていることを示す

図２．分散開発の流れ

多人数による開発の際に，もう一つ重要となる要件として，パーミッションの管理がある．コンポーネントのリザーブ/デポジットができるのは，そのコンポーネントの担当者（複数も可能）と特権ユーザに限られる．そのために，バージョン管理ファイルの中にそのコンポーネントの担当者も記述される．読み出しに関するパーミッションについては，BTRONの通常のパーミッション管理機構で十分な管理が可能である[4]．

５．まとめ

BTRONの実身/仮身モデルの特長を活かすことにより，従来のファイルシステムの下では複雑になりがちだったバージョン管理・構成管理が，記述力を保ったまま簡潔に実現できることを示した．実身/仮身モデルはバージョン管理・構成管理のみにとどまらず，ソフトウェア開発環境を構成する他の部分にも有効に活用できると考えられる．TIPEシステムの研究を通して，BTRONアーキテクチャが開発用のマシンとしても有効であることを示していきたいと考えている．

参考文献

[1] Sakamura, K., "BTRON: The business-oriented operating system", *IEEE Micro*, vol. 7, no. 2, pp. 53-65, Apr. 1987.

[2] Leblang, D. B. and Chase,R. P. Jr., "Computer-aided software engineering in a distributed workstation environment", *SIGPLAN Notice*, vol. 19, no. 5, pp. 104-112, May 1984.

[3] Belkhatir, N. and Estublier, J., "Experience with a data base of programs", *SIGPLAN Notice*, vol. 22, no. 1, pp. 84-91, Jan. 1987.

[4] 坂村健, "BTRONのファイル管理システム", 第一回リアルタイムOS-TRON研究会資料, 電子情報通信学会, pp. 2-21, Apr. 1987.