モジュール管理におけるバージョンとアクセス権の扱いの一方法*

5J−1

根岸聖†　佐渡一広‡
群馬大学§

1　はじめに

モジュールを部品として扱い、部品の依存関係と部品の変化への対応策を用いた、プロジェクトにまたがった依存関係を持つ部品のバージョン管理とプロジェクトを使ったアクセス権の扱いの一方法を述べる。また、この方法でのシステムをUNIX上で試作した。

2　プロジェクトと部品

プロジェクトとは、部品を扱い登録する単位であり、プロジェクトの構成例は図1の様になる。プロジェクトはプロジェクトが持つ部品の使用により依存関係の木を作る。そして、その木のルートとなるプロジェクトをルートプロジェクト(最初に生成されたプロジェクトでAやCなど)とする。これが一つだけ含まれる木をプロジェクトグループとする。プロジェクト名は親プロジェクトの中で閉じている。しかし、ルートプロジェクトの場合はそれが存在しないので他のプロジェクトから見ることができる。また、この依存関係では子プロジェクトで利用できる部品はその親プロジェクトでも利用できるが、逆のことはできない。このことを、プロジェクト間の部品の継承とする。このために、AがBにある部品を使用する時にはBをAの子プロジェクトとして登録することで行なう。AがDにある部品を使いたい場合は、AはDを直接見ることができないのでDのルートプロジェクトCをAの子プロジェクトとして登録することで行なう。

DからCやEの部品を使う様な部品を作るためには、その両方の部品が見えるプロジェクト(なければ作る)でそのような部品を作り適当なプロジェクトの方へそれをおく。この様にすることでプロジェクト間の部品の継承関係を壊すことなしに使用上のループを実現できる。

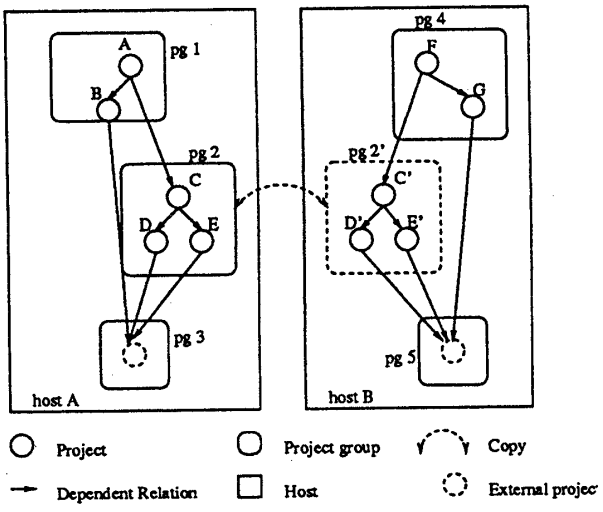

図1: プロジェクトと分散環境

プロジェクトの結合は分散環境において他の計算機やファイルシステム上でも行なうことができる。host B上のFがhost A上のCを子プロジェクトとして登録すると、それが属するプロセスグループpg2がhost Bにコピーされ、pg2'が作られる。pg2'の内容の変更を禁じ、pg2が更新された時にシステムがpg2'を更新することでデータの一貫性を保つ。host Bからのpg2'の更新は新しいバージョンの部品を要求する時に起きる。さらに、各々の計算機上に特別なプロジェクト“外部プロジェクト”が存在する。それは部品管理システムの外に存在するライブラリ等をシステム内で他の部品と同様に扱うためのものである。計算機が変わるとその計算機の外部プロジェクトが使われる。

3　バージョン管理

部品のバージョンを次の三つのもので表す[1]。

仕様版番号: 部品の仕様(機能、入出力データ構造、実現するためのアルゴリズム)のバージョンを表す。

実現版番号: ある特定の仕様版番号を実現するための手段(記述手段、言語、ターゲットとするアーキテク

---

*A Method of Managing Version and Access Right of Modules
†Satoshi Negishi
‡Kazuhiro Sado
§Gunma University

チャ) のバージョンを表す。

構成版番号: ある部品の構成要素である部品のバージョン構成を表すバージョン。

さらに、局所的なバージョンを管理するために使われる所属版番号を加える。この四つの番号を合わせて版番号とし、ある特定のバージョンの部品を指定するのに部品名と一緒に使う。仕様版番号により部品の仕様の情報が管理され、実現版番号により個々の実現の違いを管理できる。構成版番号は、部品が依存している部品のバージョン変化による自己変更の履歴を表す。この版番号の表記において、「全、新、現在」の様な曖昧な表現を許す。部品の依存関係がプロジェクトにまたがっている時にバージョン変化が起きると、依存している部品に伝える必要がある。

この伝搬は次の様にして行なう。まず、依存関係の情報 (依存情報) とバージョン変化への対応を記述した情報 (アクション情報、これはプログラム可能な言語で記述されている) を部品に持たせる。依存情報は、依存する情報と依存される情報の二種類からなる。次に、部品 pA' を部品 pA の新しいバージョンとして登録する時に、pA と pA' を指定して登録する。これによって pA が pA' に変化したことを検出する。これらを用いる例として今、D と E にある部品 pA などから C でライブラリ libC を作るとする。この時に、ユーザは使っている pA を部品名と版番号で指定し、pA が変化した時に「pA を持ってきて再コンパイルし、作り直し、登録する」などの対応策をアクション情報として libC に登録する。前者は依存情報として保存され、pA には libC に依存されているという情報が登録される。pA が pA' に変化すると pA の依存情報から libC に pA と pA' の部品名と版番号が送られる。libC では pA の指定が曖昧な記号を使わずに書かれているならば使われる部品の変化に対して自分が変化しないことを意味して何も起きないが、それ以外の時は使う部品の変化に対して自分が変化することを意味していて libC が持つアクション情報が実行される。この時に作られた libC' は libC の構成版番号を更新した版番号で libC の代わりに登録される。この登録により、pA と libC の間で起きたことが libC を使っている部品に対して起きる。このようにして、バージョン変化の情報が部品の間を伝搬していく。

4 アクセス権

アクセス権の情報はプロジェクトと部品にあり、それは読み込み権、書き込み権、実行権の三種類からなる。実行権はアプリケーション等を部品として登録した時にそれが使えるかどうかを示し、プロジェクトでは結合を許すプロジェクトを示す。これらの組合せにより、プロジェクトの結合、部品の継承、部品への書き込みなどを制限する。これらは、プロジェクト名、ホスト名、データベース名、ユーザ名等を登録することで行なう。

部品の継承において子プロジェクトとそれがもつ部品の両方が読み込み権で許していないと親プロジェクトで部品が使用できない。これは、継承してきた部品に対しても適応される。部品への書き込みはプロジェクトの書き込み権と部品の書き込み権とで制限する。また、書き込まれている最中は、他のアクセスが制限される。部品への書き込みができてプロジェクトへの書き込みができない時には、pA' を pA があるプロジェクトに登録する代わりにそれを登録しようとしたプロジェクトに登録し局所的な部品とする。この pA' により pA が引き起こすバージョン変化の情報の伝搬は起こらない。これは、局所的な領域 (局所的なバージョンの部品を継承しているプロジェクト) でその部品に関して局所的なバージョンが作られたことを意味する。前述の pg2' に対しての書き込みの禁止は pg2' のプロジェクトへの書き込み権をなくすことで実現している。より細かいアクセスの制限には部品毎に持つアクション情報を実行することで行なう。この様にすることで、アクセス権を増やすことなしに必要に応じたきめ細かなアクセスの制限ができる。

5 終りに

これらにより、プロジェクトにまたがった依存する部品のバージョン変化によって、部品のバージョン変更が可能となる。しかし、問題点があり、バージョン変化情報の獲得の問題や、アクション情報の記述作業の繁雑さがある。

参考文献

[1] Randy H.Katz. Toward a Unified Framework for Version Modeling in Engineering Databases. In *ACM Computing Surveys*, Vol. 22, No. 4, pp. 375–408. December 1990.