3R−3

ソフトウェア構成の動的変更およびその版管理の一方式

岡田　世志彦　　飯田　元　　井上　克郎　　鳥居　宏次
大阪大学　基礎工学部

１．はじめに

ソフトウェア開発過程を形式的に記述しようという試みが盛んになされている．それによって，開発工程の管理を容易に行うことや，生成物の品質がある水準以上に保たれることが期待されるからである．

また，その一方でソフトウェア開発中の生成物の管理は重要な問題である．我々は，構成管理の一プロセスの記述をこれまでに試みた[3]．

本研究では，ＵＮＩＸ上でのＣ言語によるソフトウェア開発を前提とする．そして，ソースプログラムの構成の変化に対応した版管理や実行プログラムの生成を行うための支援系を，モジュールの構成管理とファイルの構成管理の２点から考え，開発過程記述言語ＰＤＬ（Process Description Language）[1]で記述する．

ＰＤＬは関数型言語であり，ツールの起動，ウィンドウのオープン・クローズなどの操作を行う基本関数を備えている．これらの関数を組み合わせて開発過程の支援を行うプログラム（以後，スクリプトと呼ぶ）を記述できる．この記述はＰＤＬインタプリタ[2]で実行可能である．

２．支援の概要

ソフトウェア開発においては，その過程で様々な生成物が作成される．仕様書，設計文書，ソースコード，オブジェクトコード，実行プログラム，テストデータなどがそれにあたる．また，そのそれぞれには多くの版が存在し，これらをうまく関連づけて，管理するための機構が求められる．必要に応じて適切な生成物を保管場所から取り出すことや，その変更に対しての履歴を保存したり，それに伴って他の新しい生成物を作成するような作業を効率よく行えるようにすることは，ソフトウェア開発支援系の重要な役割である．

本研究ではＵＮＩＸ上でのＣプログラム開発の支援を前提としているが，ＵＮＩＸでは，版・構成管理を支援するツールとして様々なものが提供されている．版管理を行うツールとしてＳＣＣＳ（Source Code Control System）[5]，ＲＣＳ（Revision Control System）[6]等が代表的である．ＳＣＣＳ・ＲＣＳは，ファイルの差分を保存することによって版管理を行う．しかし，これらは，個々のソースに対しての版管理機能は提供しているが，その集合（モジュールなど）に対してはそのまま適用することはできない．

また，構成管理を行うためのツールには，Ｍａｋｅ[4]が一般に用いられている．Ｍａｋｅは，Makefile に記述されたファイルの依存関係に基づいて，実行プログラムを生成するといった機能を提供しているが，動的にファイルの依存関係が変化したり，任意の版を指定して再構成を行いたいような場合には向かない．

A Method of Dynamic Software Configuration Management
Yoshihiko OKADA, Hajimu IIDA, Katsuro INOUE, and Koji TORII
Faculty of Engineering Science, Osaka University

そこで，本研究で記述するスクリプトでの支援は，特に次のことに重点をおいて考えている．

・ソースプログラムの構成に対応する版管理の支援
・構成の変化に伴う新しい実行プログラムの生成の支援
・任意の版の構成を取り出して実行プログラムを再生成する場合の支援

以下では，この支援を

・モジュールの構成管理
・ファイルの構成管理

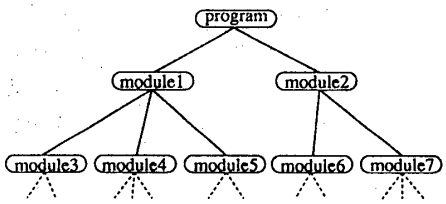
という二つのレベルから述べる．

また，このスクリプトを実行すると，作業者は，各生成物に対して行うべき作業をメニューによって選択することができる．

３．モジュールの構成管理

ソフトウェアはモジュールのような論理的な階層構造で表現される．開発者もこの構造を念頭において作業を進めて行くと考えられる．したがって，このような論理構造に対しての構成管理が必要となる．

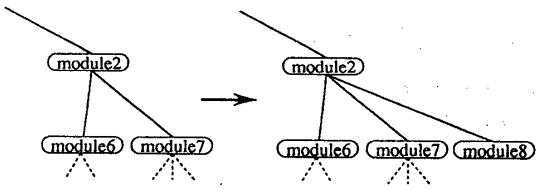
そこで，モジュールの構成管理に対する機能について述べる．



program : module1, module2
module1 : module3, module4, module5
module2 : module6, module7
module3 : ……

図１　モジュールの構成情報

構成情報を利用した実行プログラムの生成・・・ここでは，図１のように，program というモジュールは子のモジュールとして，module1, module2 を持ち，module1 は子のモジュールとして，module3, module4, module5 を持つというような構成情報を管理する．実行プログラム program を生成するためには，module1, module2 を生成しなければならず，module1 を生成するためには，moddle3, module4, module5 を生成しなければならない．このように，生成は構成情報を用いて再帰的に行う．また，あるモジュール以下の部分についてのみ，生成を行うことも可能である（モジュールテストなど）．



module2(1) : module6, module7

module2(2) : module6, modure7, module8

図 2　モジュールの構成の変更

構成情報の版管理・・・図２のように，モジュールの構成は開発中に設計のバックトラックなどにより変化すると考えられる．そこで，構成自体の版管理を行う．この例では，module2 には２つの構成が存在する．

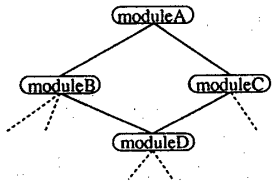
program(1) : module1(1), module2(1)
program(2) : module1(2), module2(1)
program(3) : module1(2), module2(2), module2'(1)

module1(1) : module3(1), module4(1), module5(1)
module1(2) : module3(1), module4(2), module5(2)

module2(1) : ·····

図 3　モジュールの複数の構成

任意の版の実行プログラムの生成・・・実際には，各モジュールが複数の構成を持つ可能性があるので，図３のようにその構成は再帰的に定義される．program には，現在３つの構成が存在しており，下位モジュールの別の構成を使用することなどで，新しい構成を得ることもできる．構成情報は後に自由に取り出すことができ，それを用いて，或いは，その一部を変更して，実行プログラムの再生成が可能である．



moduleA() : moduleB(1), moduleC(2)

moduleB(1) : moduleD(1)×, ·····
moduleC(2) : moduleD(2)×, ·····

図 4　構成の衝突

構成の衝突の検出・・・図４では，moduleA をmoduleB の構成１とmoduleC の構成２を用いて再構成しようとしているが，moduleB とmoduleC がmoduleD の異なる構成を用いているため，矛盾が生じる．このような場合には，エラーメッセージを出力し，すべてのコンパイル・リンクなどは実行されない．

エラーを含むモジュールの指定・・・エラーなどの原因となったモジュールの版がわかっているような場合には，開発者がそれを指定しておけば，後にその版を用いて再構成をしようとした場合は警告する（それでも再構成したい場合には，強制実行も可能である）．

４．ファイルの構成管理

モジュールとファイルの対応・・・ソフトウェアをモジュールという論理的な単位で扱う場合に問題となるのは，ファイルなどの物理的な存在との対応である．ソース自体の版管理は，実際にはファイルに対して行われる．エディタ，コンパイラなどのツールも，ファイルに対して起動される．したがって，あるモジュールのある版に含まれているファイル名とそのファイルの版番号を管理する必要がある．

インクルードファイルの管理・・・あるファイルがインクルードしているヘッダファイル名とその版番号を管理することで，１つのヘッダファイルを異なる版番号で参照しようとしている場合には，再構成時にこれを発見することができる．

関数の二重定義などの警告・・・ファイルに含まれる関数名など名前の管理することで二重定義を検出できる．

コンパイルエラーを含む版の警告・・・コンパイルに失敗するファイルの版が指定された場合には警告する．

ファイルの２段階版管理・・・ファイルの版管理には差分管理をもちいるが，ここでは，２段階の版管理を考えている．ひとつは開発者が明示的に版の登録を行った場合のものであり，実行ファイルの再生成などを行うときなどに使用されるのは，この版である．もうひとつは，開発者がエディタを閉じたときに自動的に登録されるものである．これは，バックアップや作業履歴の記録として意味を持つ．

５．おわりに

以上，Ｃのソースプログラムの構成の変化に対応した版管理や実行プログラムの生成を支援するＰＤＬスクリプトについて述べた．このスクリプトは現在作成中である．

作成後は，実際に使用して，さらに検討を重ねていきたいと考えている．

また，多人数による開発を念頭においた拡張も行いたい．

［参考文献］

[1] K.Inoue, T.Ogihara, T.Kikuno, and K.Torii: “A Formal Adaptation Method for Process Descriptions”, Proceedings of the 11th International Conference on Software Engineering, pp.145-153 (1989).

[2] 荻原，井上，鳥居：“ソフトウェア開発を支援するツール起動自動制御システム”，電子情報通信学会論文誌(D-I)，Vol.J72-D-I, No.10, pp.742-749(1989).

[3] 岡田，飯田，井上，鳥居：“開発過程記述言語ＰＤＬによるＰＤＬ処理系管理スクリプトの記述”，情報処理学会第４１回全国大会論文集(1990).

[4] D.E.Perry: “Version Control in the Inscape Environment”, Proc. 9th ICSE, Neeth, pp.316-325(1987).

[5] M.J.Rochkind: “The Source Code Control System”, IEEE Trans. Software Eng. 11,3, pp.259-266(1975)

[6] W.F.Tichy: “RCS - A System for Version Control”, Software - Practice and Experience 15,7, pp.637-654(1985).