図式表現を用いたソフトウェアプロセス構成・実行システムの試作

非会員　松永　泰明[†]　　非会員　飯田　元[†]
正　員　荻原　剛志[†]　　正　員　井上　克郎[†]
正　員　鳥居　宏次[††]

Process Description and Enaction System Based on Graphical Representation

Yasuaki MATSUNAGA [†], Hajimu IIDA [†], *Nonmembers*, Takeshi OGIHARA [†], Katsuro INOUE [†] *and* Koji TORII [††], *Members*

† 大阪大学基礎工学部情報工学科，豊中市
Faculty of Engineering Science, Osaka University, Toyonaka-shi, 560 Japan
†† 奈良先端科学技術大学院大学情報科学研究科，生駒市
Graduate School of Information Science, Advanced Institute of Science and Technology, Nara, Ikoma-shi, 630 Japan

あらまし　ソフトウェアの開発工程(プロセス)の記述を図で定義し，直接実行する開発支援システム Process View を試作した．Process View はプロセスの定義が容易に行える上，開発作業中にもその定義を動的に変更できる機能を備えている．

キーワード　プロセス，開発支援システム，プロセス記述言語，図式表現

1. まえがき

ソフトウェアの開発工程(プロセス)を形式的に記述し，この記述に従って計算機上での作業を進める(「プロセス記述を実行する」という)試みがなされている[(1),(2)]．

これまでに提案されているプロセス記述の方法は，我々が開発したプロセス記述言語 PDL (Process Description Language)[(1)] をはじめとして，テキスト(文字列の集合)として記述を行うものが多い[(2)]．しかしこれらの方法では直観的にプロセスを表現・理解するのは容易ではなかった．そこで我々は図式表現を用いてプロセスを定義する図式エディタを開発し，その有効性を確認した[(3)]．但し，この方法では，開発作業を始める(プロセスを実行する)前に，あらかじめプロセスの完全な定義を図式エディタで作成した上で，テキスト形式(PDL)に変換しておく必要があった．しかし，実際には作業中にも開発対象や進ちょく状況に応じてプロセスを変更したいことが多い．

我々は今回，プロセスを図式表現を用いて定義(記述)し実行することができ，また，実行の途中でも，開発作業を中断させてプロセスの記述を変更することのできるシステム Process View を試作した．

以下では Process View で用いるプロセスを記述の方法とシステムの概要について述べる．

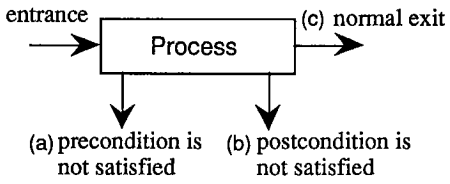

図1　一つのプロセスの図式表現
Fig. 1　Graphical representation of process.

2. Process View の概要

2.1 プロセス

Process View では，前提条件，プロセス本体，完了条件の三つ組をプロセスという[(4)]．一つのプロセスには一つの(前提条件への)「入口」および三つの「出口」がある．出口は，(a)前提条件が満たされない場合，(b)完了条件が満たされない場合，(c)正常にプロセスが終了した場合の三つである．通常一つのプロセスを箱で表し，各出入口を図1のように表す．また複数のプロセスの間の出口から入口へ有向辺で結んだグラフをプロセスの系列と呼ぶ．

プロセス本体は一つのツールの実行や人間の判断・作業などの基本作業，若しくはプロセスの別の系列である．一つのプロセスは必要に応じて階層的に構成することができる．従って，ある階層に対する変更が上位の階層に影響を及ぼさないように記述できる．また，開発対象や計算機環境，あるいは各開発者の習熟度や好みに応じて階層化の深さ(定義の詳細さ)を変えて記述することができる．

プロセスの系列は，その中に含まれる各プロセスの間の状態遷移を規定している．すなわち，あるプロセスの入口で前提条件を調べ，もし成り立てばそのプロセス本体に状態が移る．そして完了条件を調べ，成り立てば正常な終了出口で指定された次のプロセスに状態が移る．各条件が成り立たないときは，それぞれの出口で指定された先に状態が移る．プロセス本体が基本作業の場合は，その作業を実行し，別のプロセスの系列の場合は，その入口へ遷移する．

図2に簡単な階層化したプロセスの例(プログラム作成過程)を示す．

2.2 プロセスの図式表現

ここで言うプロセスの記述とは階層化されたプロセスの系列の集合を言い，これをテキスト(文字列の集合)ではなく箱や矢印の集合として表示・編集できるよう

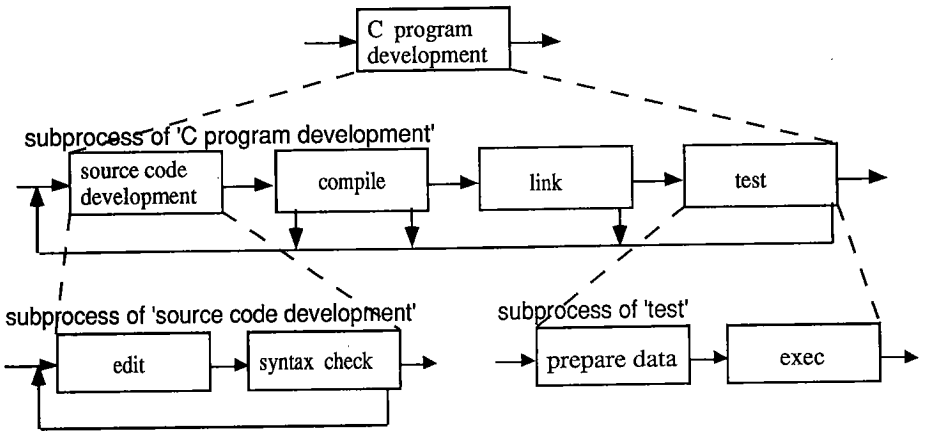

図2　プロセスの階層構造
Fig. 2　Hierarchical structure of processes.

にしたものを図式表現と呼ぶ．

テキストを用いてプロセスの記述を行う場合，その記法に習熟していなければ記述を行うのは困難である．一方，図式表現ではプロセスの構造が理解しやすく，初心者でも比較的容易にプロセスの記述を行えることが期待される．

上で述べた図式表現は，階層化された状態遷移図の一つであると見ることができる．但し，この図式表現には条件分岐や行うべき作業内容の情報が付加されており，実行可能なテキスト形式のプロセス記述と相互に変換が可能である．

プロセスを図で記述する他の方法として，データフロー図 (SADT[8] など)，ER ダイヤグラム[7]，STATEMATE[9] などを利用することも考えられる．しかし，データフロー図や ER ダイヤグラムは操作対象となるプロダクト (この場合は開発対象のドキュメントなど) を中心に記述が行われるため，作業の繰返しやり直しなど，関連するプロセスの特徴を直接記述しにくいという問題がある．STATEMATE は状態遷移グラフを用いてプロセスの振舞いをシミュレーションで調べることを主な目的として作成されたシステムであり，我々が目的としている開発支援にそのままでは適用できない．特に，以下で述べるような記述の動的な変更が行いにくいという問題点がある．

2.3　プロセス記述の動的な変更

実際のソフトウェア開発では，開発作業がいったん始まってからでも，進ちょく状況に応じてプロセスの定義を変更したいという場合が多い．また，プロセスの詳細を定義する前に開発作業を開始し，詳細化は実行中に行いたいこともあろう[5]．つまり，プロセスの実行を一時中断し，記述を追加・変更した後，再実行できるような機構が必要である．そのために，プロセスを定義・記述するシステムとそれを実行するシステムを別に開発するのではなく，プロセスの定義を行うユーザインタフェースを実行システムの一部としてこれらを一体化して作成し，密に連絡を取り合えるようにした．

プロセスの実行を中断しようとすると，システムは，各階層ごとに実行中のプロセスの名前や位置などの情報を保存する．記述に変更を加えて実行を再開しようとすると，中断が起こったときに実行中だったプロセスを探し，そのプロセスの実行を再開する．変更によっては，新たな記述と保存された中断前のプロセスの情報が一致しない場合が起こり得るが，この場合は問題が生じた階層の入口のプロセスから再開を試みる．そのプロセスからも再開できない場合は，順次，上の階層の入口のプロセスからの再開を試みる．しかし，中断されたプロセスの名前や階層関係の変更など，記述に大幅な変更を加えた場合は実行を再開できないこともある．また，システムが再開しようとするところが作業者が再開したい位置とは限らない．これらの場合には作業者が再開位置を明示しなければならない．

3.　Process View 処理系

今回作成したシステムは互いに通信を行いながら動作するインタフェース部と記述実行部から構成されている．

インタフェース部は X Window 上で実現されており，マウスを使ってプロセス記述を作成する．作成された

プロセス記述はLispのS式に変換され，記述実行部に送られて実行される．現在Process ViewではUNIXシステムのパイプを利用してインタフェース部と記述実行部の通信を行っている．

記述実行部ではXLISP†を用いてプロセス記述の実行と中断・再開の仕組みを作成した．中断・再開の仕組みおよびインタフェース部との通信に必要な機能を組込み関数としてXLISPに付け加え，記述の解釈実行を行っている．

4. むすび

現在，Process ViewはDEC Station上で稼動中である．Process Viewでは図式によってプロセス記述を行うため，視覚的に理解しやすく，また多くの操作をマウスのクリックのみで行うことができる．このため，はじめてプロセス記述を行う場合でも容易に操作できる．

また，記述の作成環境と実行環境の一体化によってプロセスの定義を作業の途中で変更することが可能になった．今後，プロセスの変更に伴って生じる問題，例えば，必要とするプロダクトが存在しない等の問題を扱う必要があると考えられる．このために我々は，プロダクトの振舞いを規定するプロダクトモデルを導入し，プロセスの変更と同時にプロダクトモデルとの整合性を調べることを検討している．

謝辞 本研究を進める上で貴重な議論および助言を頂いたさくらケーシーエスの石若道利氏，山本一成氏およびシステム作成に御協力頂いた大阪大学基礎工学部の森藤元氏に深謝します．

文献

(1) 荻原剛志，井上克郎，鳥居宏次："ソフトウェア開発を支援するツール自動制御システム"，信学論(D-I)，**J72-D-I**，10, pp. 742-749 (1989-10).

(2) Sutton S. M. Jr.: "A process-program in APPL/A for the software-process modeling problem", Sixth international software process workshop. Arcadia Document CU-90-06, University of Colorado, Department of Computer Science, Boulder, Colorado 80309 (Sept. 1990).

(3) 荻原剛志，飯田 元，井上克郎，鳥居宏次："ソフトウェア開発環境定義用スクリプトの生成法について"，情報処理学会CASE環境シンポジウム論文集，pp. 7-14 (1989-03).

(4) Williams L. G.: "Software process modeling: A behavioral approach", Proc. 10th Int. Conf. Software Eng., pp. 174-186 (April 1988).

(5) 柳瀬健一，荻原剛志，井上克郎，鳥居宏次："実開発におけるプロセス記述の試みと開発支援に関する考察"，ソフトウェア・シンポジウム '91 論文集，pp. 36-42 (1991-06).

(6) 松永泰明，荻原剛志，井上克郎，鳥居宏次："図式表現によるソフトウェアプロセスの構成と実行の試み"，日本ソフトウェア科学会第8回大会論文集，pp. 141-144 (1991-09).

(7) Chen P.: "The Entity-Relationship Model: Toward a United View of Data", ACM Trans. on Database Systems, **1**, 1, pp. 9-36 (1976).

(8) Marca A. David: "Augmenting SADA™ To Develop Computer Support for Cooperative Work", Proc. 13th Int. Conf. Software Eng., pp. 94-103 (May 1991).

(9) Harel D., Lachover H., Naamad A., Pnueli A., Politi M., Sherman R. and Shtul-Trauring A.: "STATEMATE: A Working Enviorment for the Development of Complex Reactive Systems", Proc. 10th Int. Conf. Software Eng., pp. 396-406 (April 1988).

(平成4年9月4日受付，5年1月5日再受付)

† David Micheal によって作成され，Common Lisp のサブセットとして広く利用されている．