1D－1

レジスタ付きペトリネットで書かれたソフトウェアプロセス記述の分散実行系

村岸 厳† 東野 輝夫‡ 谷口健一‡

†奈良先端科学技術大学院大学 情報科学研究科 情報システム学専攻
‡大阪大学 大学院基礎工学研究科 情報数理学専攻

1 はじめに

近年，グループによるソフトウェアの開発，変更，保守などの工程をプロセスとして明確にモデル化し，計算機による実行支援を行うことなどを目的としたソフトウェアプロセスの研究がさかんである．ソフトウェアプロセスを形式的に記述するためには，各技術者の作業の内容とその実行順序の指定，そして各技術者が指定通りに作業を実行するための技術者間の連絡作業を記述する必要がある．

我々の研究グループでは，文献[1]や文献[2]で，仕様記述言語であるLOTOSや，レジスタ付きペトリネットモデル (Petri Net Model with Registers, 以下PNRモデル) を用いて，ソフトウェアプロセス全体を技術者の作業内容とそれらの実行順序のみで記述し (これを**全体プロセス記述**と呼ぶ)，その記述から各技術者ごとの動作記述 (その技術者が行うべき作業と連絡作業，それらの実行順序が指定されたもの．これを**個人プロセス記述**と呼ぶ) を機械的に導出し，さらにそれらを分散システム上の複数の計算機で実行することにより，各技術者の開発作業を支援する方法についての研究を行っている．そこで，我々は文献[2]で提案されている手法に基づき，PNRモデルで記述された技術者の個人プロセス記述を分散システム上で協調実行し，技術者の開発作業を支援するシステムを作成した．

2 PNRモデルによるプロセス記述例

本章では例として，PNRモデルによる簡単なソフトウェア開発プロセスの全体プロセス記述例と，文献[2]で提案した手法に基づき，その全体プロセス記述から導出した各技術者の個人プロセス記述を示す．導出方法の詳細については文献[2]を参照のこと．

ここでは以下のような3人の技術者$QE, SE1, SE2$による一連の作業の繰り返しを考える (簡単のため，終了条件等は考慮していない)．

(1) QEが旧仕様書を変更する．
(2) $SE1$がその新しい仕様書をもとにコーディングを行う．
(3) (2). と並行に$SE2$が新仕様書をもとにテストデータを作成する．
(4) QEが (2). で作成されたオブジェクトコードを (3). で作成されたテストデータでテストを行う．

図1は，この作業の全体プロセス記述の例である．

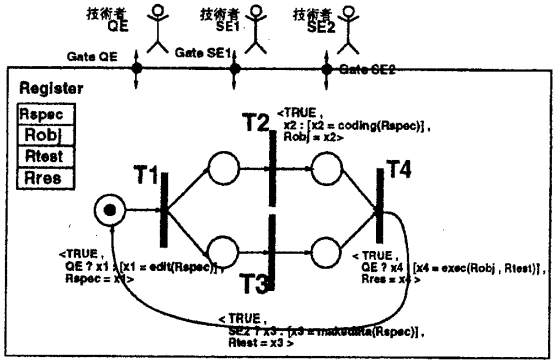

図1.PNRモデルによる全体プロセス記述例

トランジション$T1$の発火によりQEがツール**edit**を使って旧仕様書 (レジスタ$Rspec$の内容) を変更し，またもとの$Rspec$に格納する．この時点で初めて$T2$と$T3$が発火可能になり，$SE1$は更新された$Rspec$をもとにツール**coding**を使ったコーディング作業が，また$SE2$は$Rspec$をもとにツール**makedata**を使ったテストデータの作成作業がそれぞれ可能になる．$T2, T3$が発火するとそれらの作業が行われ，コーディングの結果 (オブジェクトコード) がレジスタ$Robj$に，また作成されたテストデータがレジスタ$Rtest$にそれぞれ格納される．この時点で$T4$が発火可能になり，QEが新しく作成されたオブジェクトコード ($Robj$の内容) を新しいテストデータ ($Rtest$の内容) でテストをする作業が行える．そして，テスト結果がレジスタ$Rres$に格納される．

以上の全体プロセス記述から導出された各技術者の個人プロセス記述の例を図2に示す．

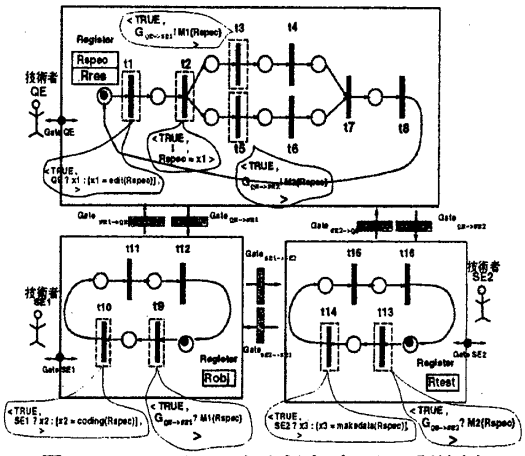

図2.PNRモデルによる個人プロセス記述例

各技術者にはそれぞれいくつかのレジスタが配置されている．図2では，QEにはレジスタ$Rspec$と$Rres$が，$SE1$にはレジスタ$Robj$が，$SE2$にはレジスタ$Rtest$がそれぞれ配置されている．各技術者は，自身に配置されておらず他の技術者に配置されてい

A System for Distributed Execution of Software Process Descriptions in a Petri Net Model with Registers
Tsuyoshi Muragishi, Teruo Higashino and Kenichi Taniguchi
†Nara Institute of Science and Technology
‡Osaka University

るレジスタ値を使って何か作業をしなければならない場合, そのレジスタ値をメッセージで他の技術者から受け取ってから作業を行うことになる.

たとえば $SE1$ や $SE2$ は QEが新たに作成した仕様書 (レジスタ $Rspec$の内容) をもとにコーディングの作業 (トランジション $t10$) やテストデータの作成の作業 (トランジション $t14$) を行うが, $SE1$ や $SE2$ にはレジスタ $Rspec$が配置されておらず, QEに配置されている. したがって $SE1$ や $SE2$ はそれぞれの作業を, QEがレジスタ $Rspec$の値を更新した ($t2$) 後, その $Rspec$ の値をメッセージとして QEから受け取って ($t9, t13$) から行う. 以上の動作は, 図 1 の全体プロセス記述において, $T1$ が発火した後に $T2$ と $T3$ が発火可能になって発火するという動作と同じである.

このように各技術者はお互いにメッセージを交換をしながら協調して全体記述に書かれた作業を行っていく必要がある. しかし, 全体プロセス記述の設計やモデル化を考える場合, 個人プロセス記述に書かれるようなメッセージ交換を考慮すると, そのソフトウェアプロセス自身の理解や評価が難しくなる. また, 技術者間の連絡作業の内容やタイミングを誤りなく設計, 実現するのも容易でない.

3 実行支援システム

3.1 実行支援システムの設計

我々が作成したシステムは, (1) 全体プロセス記述と技術者やリソース (レジスタ) の配置情報から, 対応する個人プロセス記述の組を生成する生成系と, (2) 生成された個人プロセス群をネットワーク上で分散実行する実行支援システムからなる. 以下では紙面の都合上, 実行支援システムの概略 (図 3) について紹介する. 実行支援システムは, 図 3 のように, 導出した個人プロセス記述をその技術者が使用するネットワーク上の計算機でそれぞれ解釈, 実行することにより技術者の作業支援を行う.

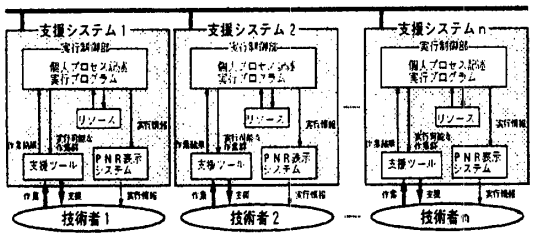

図 3. 実行支援システムの概略図

3.2 実行支援システムの支援機能

実行支援システムが提供する主な支援機能は以下のとおりである.

1. プロセスの実行制御および技術者間の連絡作業の自動実行機能： 各技術者が使用する計算機上で, 導出された個人プロセス記述を解釈, 実行することにより, 各技術者の作業の実行順序制御及び, 連絡作業の自動実行を行う. 前章の例で, $SE1$ や $SE2$ は, 自身に配置されていないレジスタ値を QEからメッセージとして送信してもらうことを述べたが, そのようなメッセージのやりとりを自動的に実行する機能が連絡作業の自動実行である. この機能により, 技術者は連絡作業の誤りを防ぐことができ, かつ本来の作業のみに集中できるため, 個人あるいは全体の作業効率をあげることができる.

2. 作業の誘導およびそれに必要なツールの自動起動機能： 現在のマーキングで発火可能なトランジションを発見し, そのようなトランジションから実行可能な作業を抽出し, それらを表示, 選択できる機能を与える. さらに, 選択した作業で必要となるツールを自動的に起動する機能を与える. 前章の例 (図 1) では, ツールとして **edit, coding, makedata, exec** があるが, これらのツールを自動的に起動する機能である. この際, 作業に必要なファイル等はツールの引数としてシステムが与えるため, 作業者はツールを起動する手間が省けるとともに, 作業の誤り (ファイルの読み込み誤りなど) を防ぐことができる. また, ユーザの環境, 開発するソフトウェアなどへの対応に柔軟性を持たせるため, **edit** などにより実際に起動されるツールは別に用意されたツールテーブルに指定することにより自由に設定できるようになっている.

3. プロセスの可視化： 図 4 にシステムの実行中画面を示す. 各技術者はこの表示機能により, 個人プロセス記述に関する情報を得ることができる. 例えば, 技術者は, 今自分がどの作業をしているのか, また次に何をすべきかといったプロセスの動的な情報を容易に把握することができる. 図 4 では, EXECUTABLE という欄の下に, 現時点での行うことが可能な作業が一覧表示されており, 技術者はこれらの作業の中から次に行う作業を選ぶことができる. さらに, どの連絡作業で遅延が生じているかといった管理情報もある程度把握でき, 技術者間の非公式なコミュニケーション (電話, 電子メールなど) により遅延を少なくすることもできる.

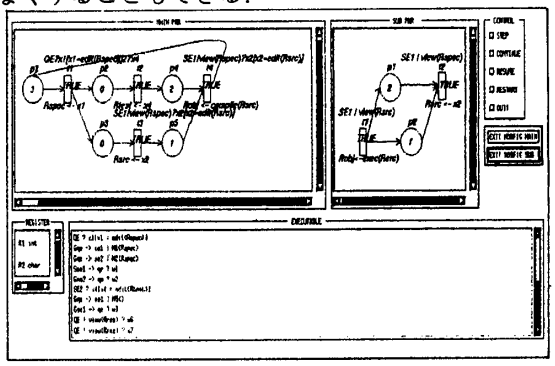

図 4. システムの実行中画面

4 おわりに

本稿では, 文献 [2] で提案した手法に基づき, PNR モデルで記述されたソフトウェアプロセスの各技術者の個人プロセス記述を分散システムの複数の計算機上で分散実行する支援システムの機能とその有用性について述べた. 本システムを用いることにより, 各技術者は他の技術者との連絡作業に煩わされることがなくなる. また実行可能な作業の誘導や作業状況の把握により, 作業を効率よく進めることができる.

今後の課題としては, 現実のソフトウェア開発には必ず期限があることを考慮し, 実行支援システムに作業の進捗状況を管理する機構などを組み入れることなどが考えられる.

参考文献

[1] 安本, 東野, 谷口: "LOTOS によるソフトウェアプロセスの記述とその実行", コンピュータソフトウェア, Vol. 12, No. 1, pp. 16-30 (1995)

[2] 山口, 岡野, 東野, 谷口: "レジスタ付きペトリネットモデルで記述されたソフトウェアプロセスの実行支援", 情報処理学会ソフトウェア工学研究会サマーワークショップ・イン・立山, pp. 89-96 (1995)