プロセスモデル化の例題の協調開発支援管理システムによる実行

5S－10

三村圭一† 飯田元† 井上克郎† 鳥居宏次‡

†大阪大学 ‡奈良先端科学技術大学院大学

1 はじめに

ソフトウェアの規模と需要の増大につれ，開発効率と信頼性の向上がソフトウェア工学における重要な目標のひとつとなっている．我々は，ソフトウェア開発の効率的な支援を目的として，開発作業の形式化やそれに基づく作業の誘導や自動化に関する研究を行なっており，これまでに，複数人によるソフトウェア開発過程の作業モデルを提案し，そのモデルに基づいた支援や管理を行なうシステム「はこにわ」を試作した．

本研究では，はこにわシステムを用いた開発を例題を用いて試験的に行ない，モニタリングによるデータの収集を試みた．

2 はこにわシステムの概要

2.1 作業モデル

はこにわシステムでは，複数人による開発プロセスを「作業モデル」と呼ぶ枠組で表現し，それに基づいた支援・管理を行なう．作業モデル (図1) は，並列に実行される複数の「タスク」によって構成される．タスクは基本作業の集合と，それを用いて定義される文法（ここでは正規文法に制限する）によって構成される．従って，各タスクは，それぞれの持つ基本作業をアルファベットとした正規表現で表すことができる．このとき，基本作業とは，対象となる開発過程をとらえる上での最小の作業のことである．

一つのタスクは，比較的単純な順序で行なわれる作業群を表す．複雑な作業順序は，タスク間のコミュニケーションによって実現される．このため，タスク間では簡単な非同期の通信を行なえることとする．

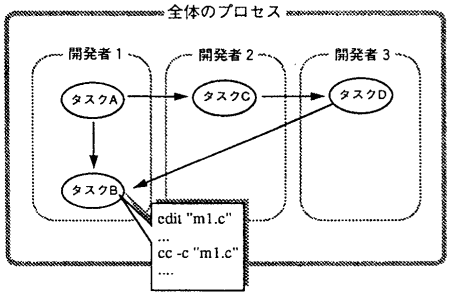

図 1: 作業モデルの概要

2.2 はこにわシステム

システムの構成を図2に示す．作業モデルに基づいた開発プロセスの記述から (a) タスクのモニタリングと，タスク間のコミュニケーションを行なう はこにわサーバと，(b) 個々の開発者が用いるための支援環境（タスクドライバとタスクマネージャ）が生成される．

Enacting Software Process Description with Hakoniwa System
Keiichi MIMURA, Hajimu IIDA, Katsuro INOUE, and Koji TORII
Osaka University
Advanced Institute of Science and Technology, Nara

このシステムでは次の三つの支援を行なう．

(1) 作業の誘導

実際の開発では，複数のタスクを一人の開発者が担当する．そこで，まず管理者がタスクの割当を行ない，その情報を元に，各開発者が用いるための支援インターフェース（タスクドライバ／マネージャ）が生成される．このインターフェースは，タスクごとに定義されている文法に基づいて，各タスクにおける作業の進行をメニューを用いて誘導する．メニューを通じて選択された作業のうち，開発ツールを用いるものなどについては自動的にツールの起動を行なう．

(2) モニタリング

一方で，開発者用インターフェースはメニュー選択によって進行して行く作業の履歴をはこにわサーバに通知する．管理者はこの情報をプロジェクト全体の状態の把握に役立てることができる．

(3) コミュニケーション支援

タスク間のコミュニケーションは，すべてはこにわサーバを介して行なわれ，作業の開始要請や，終了通知などといった，開発者間での単純なコミュニケーションは自動的に行なわれ，記録される．

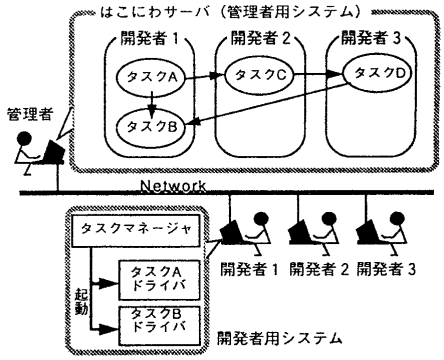

図 2: 支援システムの概要

3 運用実験

3.1 例題

はこにわサーバの運用実験として，例題を用いたプログラムの作成を行なった．

例題には酒屋問題 [1] を用いた．また，実験で用いた開発プロセスの原型には，Kellner らによって提案されているプロセスモデリングのための例題 [2] を用いた．この問題は，あるソフトウェアシステムの一つのモジュールに対して変更を加える作業を規定しており，8つのサブステップに分けられ，それぞれ様々な開始・終了条件などが付加されている．実験で用いたプロセスは，この例題を元に変更を加えたものである．主な変更点は次のようなものである．

- プログラムはモジュールに分割され，各モジュールは個別に作成される．

- モジュール毎の単体試験は行なわない．したがって，単体試験パッケージの作成・変更も行なわない．
- 最後に試験（統合試験）を行なう．そのための試験データの作成も行なう．

このプロセスの概要を図3に示す．図の中の「デザイン作成・変更1」,「コード作成・変更1」などがそれぞれタスクである．これらのタスクは並行に行なわれる．また，タスク間の進行に関する制限，例えば「デザインレビューが終了するまで，コード作成・変更は終了できない」,「コード作成・変更と試験データの作成・変更が終了するまで，試験は開始することができない」といったものが存在する．このような制限は，タスク間の通信等によって実現する．

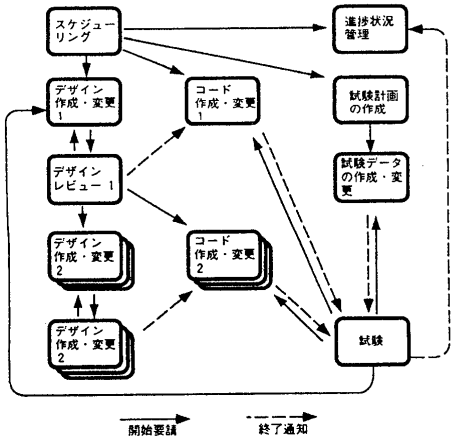

図 3: 実験で用いたプロセス

3.2 タスク内部の記述

タスクの内容の記述の例として，タスク「デザイン作成・変更1」の記述を示す．記述は次のようになる．

```
ModifyDesign1 =
        < CheckOut desgin1 > < Modify design1 > + < CheckIn design1 >
                                                < start ReviewDesign1 >
```

ここでは，まずデザインを CheckOut し，デザインの編集（< Modify design1 >）の繰り返しを+記号で指定した後，デザインを CheckIn し,「レビューデザイン1」に開始要請を送信（< start ReviewDesign1 >）している．

3.3 記述の実行

各タスクを次のように3人の開発者（大阪大学基礎工学部情報工学科の学生）に割り当て，実験を行なった．

- 開発者1：「タスクのスケジューリング及び割り当て」,「進捗状況管理」,「デザイン作成・変更1」,「デザインレビュー1」,「コード作成・変更1」
- 開発者2：「デザイン作成・変更2−1」,「デザインレビュー2−1」,「コード作成・変更2−1」
- 開発者3：「デザイン作成・変更2−2」,「デザインレビュー2−2」,「コード作成・変更2−2」,「試験計画の作成」,「試験データの作成，変更」,「試験」

4 収集データ

実験では (1) 基本作業の実行履歴: 各開発者が行なった基本作業（ツールなど）の実行履歴（図4）と，(2) タスク状態の履歴: タスク状態の遷移の記録を収集することができ，これらの情報から様々な時点でのタスク状態（図5）を得ることができる．

また，デザイン，コード，試験計画，試験データなどの生成物の変更履歴も，版管理ツールの自動化が行なえるため容易に収集できる．

```
92-08-17 16:11 ModifyCode 1      CheckOut 'module1' begin
92-08-17 16:11 ModifyCode 1      CheckOut 'module1' end
92-08-17 16:11 ModifyCode 1      editCode 'module1' begin
92-08-17 16:32 ModifyCode 1      editCode 'module1' end
92-08-17 16:32 ModifyCode 1      compile 'module1' end
92-08-17 16:33 ModifyCode 1      compile 'module1' end
92-08-17 16:34 ModifyCode 1      editCode 'module1' begin
92-08-17 16:41 ModifyCode 1      editCode 'module1' end
92-08-17 16:41 ModifyCode 1      compile 'module1' begin
92-08-17 16:42 ModifyCode 1      compile 'module1' end
92-08-17 16:42 ModifyCode 1      CheckIn 'module1' begin
92-08-17 16:42 ModifyCode 1      CheckIn 'module1' end
```

図 4: 基本作業の実行履歴（例）

```
----
ModifyCode 0
----
parameter : "a.out",("main","sub1","sub2")
status :
    seq n0 (start=92-06-07 22:40)
      rep n1 (1) (start=92-06-07 22:40)
        rep n2 (1) (start=92-06-07 22:40)
          *rep n3 (1) (start=92-06-07 22:40)
            term editCode '' (start=92-06-07 22:40 end=92-06-07 22)
          term compile '(null)'
        peek ReviewDesignOk
      recv ReviewDesignOk
----
```

図 5: タスク状態（例）

5 おわりに

協調開発支援管理システムはこにわを用いたソフトウェアの開発を例題を用いて実際に行なった．モニタリングによって基本作業の実行履歴やタスク状態の履歴，生成物の履歴を収集した．タスク状態の履歴から，各タスクの実行時間，繰り返し回数を収集することができる．これらを蓄積することによって，コストの予想や，進捗状況の把握に役立てることができると考えられる．

参考文献

[1] 二村良彦, 雨宮真人, 山崎利治, 淵一博: “新しいプログラミング・パラダイムによる共通問題の設計”, 情報処理, Vol. 26, No. 5, pp.458-459(1985).

[2] Marc Kellner: “Software process modeling example problem”, Proc. 1st Int. Conf. on Software Process, Redondo Beach, CA, pp.176-186,(1991).

[3] H. Iida, T.Ogihara, K.Inoue and K.Torii: “Generating a menu-oriented navigation system from formal descriptions of software development activity sequence”, Proc. 1st Int. Conf. on Software Process, Redondo Beach, CA, pp.45-57,(1991).