ソフトウェア工学 131－3
（2001．6．1）

ソフトウェア分散共同開発のための チーム構造モデルの定義と調整支援への応用

落水 浩一郎

北陸先端科学技術大学院大学 情報科学研究科

〒923-1292 石川県能美郡辰口町旭台 1-1
Phone: 0761-51-1260, Fax: 0761-51-1149
e-mail: ochimizu@jaist.ac.jp

概要

本稿では、分散共同ソフトウェア開発に対するソフトウェアプロセスモデルを構築するにあたっての基本的要素となるチーム構造モデルについて考察する。組織構造と規則、役割と責任範囲、地理的分散、コミュニケーション等の問題を主に論じる。また、チーム構造モデルを利用した能動的調整支援の方針についてまとめる。

和文キーワード

ソフトウェア分散共同開発、ソフトウェアプロセスモデル、チーム構造モデル

Design of a Team Model and it's application to co-ordination support for a Distributed Co-operative Software Development

Koichiro OCHIMIZU

School of Information Science, JAIST

Asahidai 1-1, Tatsunokuchi, Ishikawa 923-1292

Abstract

In this paper, we will consider a team model in constructing a software process model for a distributed co-operative software development. We will discuss and define the model mainly from the oragnizational sturucture and rules, role and responsibility, geographical distribution, comuunication. We also propose an active co-ordination method based on the team model defined in this paper.

英文 key words

distributed cooperative software development, software process model, team model

1 はじめに

今日、地理的に分散した人々がネットワークを介して共同でソフトウェア開発に従事する機会が増大しつつある。ネットワークを介した共同作業の支援のためには、**不安定さの共有、矛盾や曖昧さの漸増的解消、作業効率の改善**等にわたって新しい技術の開発が必要である。われわれは、ネットワークを介した共同作業の支援において、分散共同作業の制御方式に理論的基盤を与えることを目標に、以下のような解構築の基本方針を立てている [1, 2]。

- 実世界の開発状況とその変化を忠実に反映する情報リポジトリを関係者に共有させ (**状況の不安定さの共有と awareness 支援**)、一貫性と確実さの漸増的補強を支援する機能を提供する (**co-ordination 支援**)。これにより、矛盾や不確実さでの作業遂行を可能にする (図. 1)。

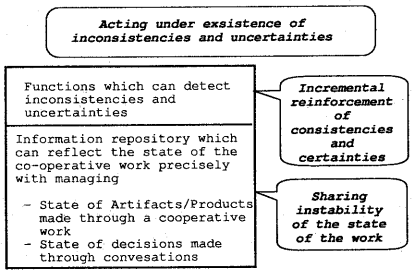

図 1: 解構築の基本方針

上記方針に基づき、ソフトウェプロセスモデルとソフトウェア開発環境の構築を試みている。すでに、**上記情報リポジトリの一実装**を実現中である (図. 2)。以下のプログラムやモデルを開発することにより、必要な情報を収集可能なデータから自動的に抽出する手段を開発中である [3]。

- 自然言語処理による会話分析の手法を拡張することにより、メーリングリスト中のメールをコントリビューション木の集合として整理し、**話題ごとの討議スレッドを自動抽出するプログラム** [4][5]。決定事項などの作業のコンテキストを与えることを目標とする。
- 版管理システム CVS のログ情報から、「更新頻度の高い情報の特定」、「特定の開発者による作業内容」、「特定の期間に行なわれた作業内容」などの**開発状況に関する統計情報を収集するプログラム** [6]。作業負荷の不均衡や、作業の遅延などの検知によるリスク管理を目標とする。
- Java ソースコードをシステム依存グラフ (抽象構文木) に変換し、システム依存グラフ間の差分**抽出や波及解析をおこなうプログラム** [7]。ある人の変更が他人におよぼす影響を検知することを目標とする。
- 上記 3 つのデータ構造をチーム構造モデルにより統合し、抽象度のレベルに応じて実世界をシミュレートしつつ、**解消すべき矛盾や曖昧さの存在に基づいて、オブジェクト関係活動やコミュニケーションを駆動・制御する** Active Coordinator[8]。能動的調整支援を目標とする。
- **切断された状況での作業継続や先を見越した作業を可能にする未来版管理機構** [9, 10]。作業の独立性と並行性を高めることを目標とする。

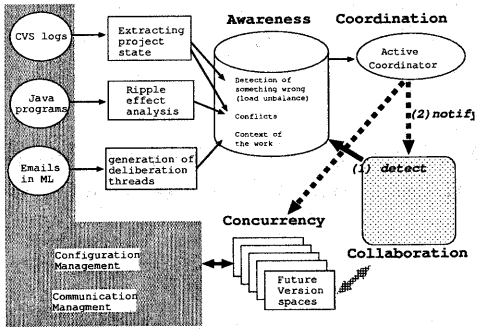

図 2: 自在プロトタイプの構成

我々は、分散共同ソフトウェア開発を調整・制御するソフトウェアプロセスモデルの要件を以下のようにとらえている。

- 個人の責任範囲と他人との作業インタフェースの明確な定義
- 個人の活動とチームの活動の関係の定義
- オブジェクト中心の活動とコミュニケーションプロセスの融合
- 分散共同作業に固有の不安定さにともなって発生する矛盾や曖昧性の管理

すでに [3] において、上記要件を、Coplien による組織パターン [11]、Activity Theory[12] の知見に基づき整理することにより **Active Coordinator** の必要性を指摘した。Active Coordinator は、チーム構造モデルを内蔵するオブジェクトであり、チーム構造モデルの抽象レベルに応じて、解消すべき矛盾や曖昧さを検知し、オブジェクト関連活動やコミュニケーション関連活動を駆動することにより、実世界の制御を行なう。本稿では、チーム構造モデルについてさらに詳細に考察する。

2 自在環境の Unified Process による分析

図. 2に示した自在環境の分析と設計を Unified Process に従って実施した。図. 3に分析結果 (ユースケース図と分析クラス図をマージした図) の一部を示す。

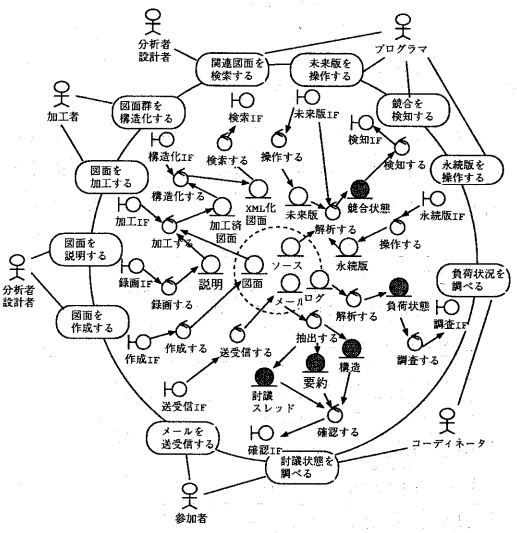

図 3: 自在環境の Unified Process による分析

図. 3において、黒く塗り潰した実体オブジェクトは、図. 1 における情報リポジトリが管理する状態情報に対応する。

3 チーム構造モデル

図.4にチーム構造モデルの概要を示す。図.4の各要素と相互関係について以下の各節で説明する。

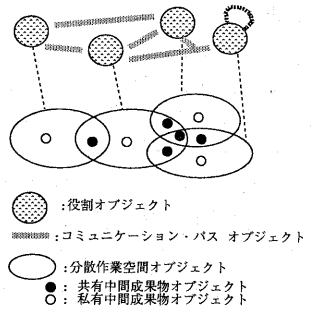

図 4: チーム構造モデル

3.1 役割オブジェクト

Coplien によると [11]、「密接に関係する活動 (互いに組になっている実装、同じ中間成果物を操作する活動、同じ領域に対して意味的に関係している活動など) をグループ化して、役割として定義する。役割に結合された活動を責任として定義する」とある。本研究では、図. 3における、Unified Process におけるアクターと Coplien による役割を対応させる。また、アクターが利用するユースケースに含まれる活動を Coplien による責任と対応させる。責任は、オブジェクト関連責任とコミュニケーション関連責任に分類する。さらに、各役割 (ロール、アクター) 毎にワークフローを定義し、進捗状況表現の基礎とする。ワークフローはアクティビティ図で表現する。

Role: role name
The Person-in-Charge: name
Activity-in-Use: Workflow
Object-related-Responsibility: activity-name-1(artifact, access right) ... activity-name-n(artifact, access right)
Communication-related-Responsibility: communication-path-1(group, artifact) ... communication-path-n(group, artifact)

ソフトウェアエンジニアリング諸活動には、オブ

ジェクト関連活動とコミュニケーション関連活動の2種類がある。役割オブジェクトの定義にあたっては、両者の融合について考慮する必要がある。両者の関係を図.5に示す。共同作業の参加者は、コミュニケーションを通じて、共同作業のゴール、中間成果物やその要素の機能・役割、実装法に関する共通認識に達する。共同作業のバックグランドとなる共通認識は、まず、状況の把握からはじまり、認識のズレの解消へと進む。コミュニケーションの結果、各自は新しい認識状態に達し、その状態を各自の責任範囲下にある中間成果物オブジェクトに反映させる。このような中間成果物の状態の変化は、再び、関係者間のコミュニケーションを引きおこす。上記の活動群はActive Coordinatorによって連携させる予定であるが、実行制御の詳細は今後の課題である。

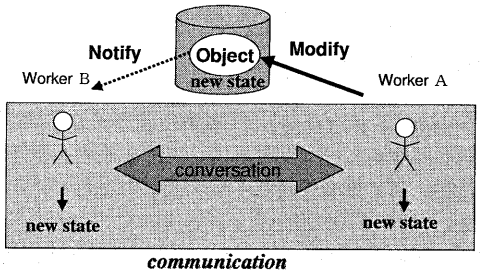

図 5: オブジェクト関連活動とコミュニケーション関連活動の関係

3.2 コミュニケーション・パス・オブジェクト

Coplien[11] によれば、「不必要なコミュニケーションの発生を防ぐためには、チーム構造とコミュニケーションのパスは役割における責任 (役割に結合された活動) 間の意味的結合に従って設計する」とある。本研究では、分析モデルまたはコラボレーション図におけるイベントの流れに従って、(通知する、通知される)、(質問する、答える)、(共に検討する) などの、アクター (役割) 間の結合を定義し、設計の基礎とする。これにより、役割オブジェクト間にコミュニケーション・パスがはられる。コミュニケーション・パスもオブジェクトとして表現する。また、コミュニケーション・パス・オブジェクトは、後述する、複数の討議スレッドオブジェクトを内蔵する。

Communication-Path : number
Realted-Roles: roles
Related-Delibereation-Thread: deliberation-thread-1 ... deliberation-thread-n
Participants: group
State: (active, inactive)

コミュニケーション・パスの実際の設計にあたっては、さらに以下の点を考慮する必要がある[1]。

- 開発構成員の地理的配置に従ってアーキテクチャの構造を設計せよ。
- アーキテクチャの構造に従って開発構成員の地理的配置を決定せよ。
- マーケットの構造をチーム構造、すなわちアーキテクチャに反映させよ。
- チーム外部とのコミュニケーションは Gate-Keeper を介して行なえ。

3.3 討議スレッドオブジェクト

討議スレッドとは、電子メール中に含まれる複数の討議の流れを話題毎に構造化したデータ構造である [4]。下記のようなオブジェクトとして表現する。

Deliberation-Thread : number
Comunication-Path: number
Topics: content
History: content
Decision: content
Summary: content
Participants: group
State: (active, completed, left)

3.4 役割と責任範囲の定義

役割オブジェクト毎に、以下に定義する分散作業空間 (責任範囲) を配置する (図. 4破線)。

[1] 最初の3つの項目を同時に充たす解は存在しない場合もありうる

3.5 分散作業空間オブジェクト

役割毎の作業の独立性を可能な限り保証する立場をとり、役割毎の作業の責任範囲と他の役割との関係を記述することにモデル定義の力点を置く。具体的には、役割毎の仕事の責任範囲を中間成果物オブジェクトの集合としてとらえる (以後タスクと呼ぶ)。**分散作業空間** [13] は，一部の中間生成物が共有されているタスクである。分散作業空間の積集合の要素が**共有中間成果物オブジェクト**であり、他の役割との作業インタフェースである。複数の分散ワークスペースの重なりの時間軸上での動的変化をプロセスの実行として定義する。

Distributed-Work-Space : dws name
Related Role: role name
Private Artifacts: artifact-1(future-version-space-number) ... artifact-n(future-version-space-number)
Shared Artifacts: artifact-1(trunk-number) ... artifact-m(trunk-number)

3.6 中間成果物オブジェクト

UML を利用した各種方法論における各種中間成果物 (ユースケース図、コラボレーション図、クラス図、シーケンス図、ステートチャート図、配置図、コンポーネント図、テストケース) と Java プログラムを対象としている。中間成果物間には UML における trace 依存関係が設定されていること、版管理がなされていることを前提する。

Artifact : name
Type: (diagram, class, interface)
Version: revision-number
dependency: list
state: STD

中間成果物オブジェクトの状態遷移を図. 6のように定義する。

3.7 分散作業空間と未来版管理機構

作業の並列性を高め、非同期性を向上させるようなソフトウェアプロセスを可能にするには、

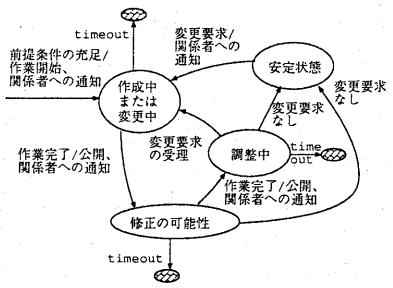

図 6: 中間成果物オブジェクトの状態遷移

- 予想される変更に備えて、先を見越して作業を進める
- 共有中間成果物の変更要求が承認されていなくても私有作業空間で改訂作業を先行する
- 移動計算環境で独自に作業を進め得る

等の作業を支援するモデルと機構の開発が必要である。我々はすでに、「未来デルタ機構と汚染マーキング機構」の基本的構成を定義しプロトタイプを開発した [9, 10]。**未来版管理機構**は、未来版空間、永続版空間、変更要求空間からなる。**未来版空間**は、変更要求が関係者によって承認されていなくとも、先を見越して作業を進めるための作業空間である。

Future-Version-Space : fvs name
Branches: branch-1(artifact) (side effect: notification of conflicts) ... branch-n(artifact) (side effect: notification of conflicts)

Change-Request-Space : crs name
Change-Request: approved-changes: list not-approved-changes: list

Persistent-Version-Space: pvs name
Trunks: trunk-1(artifact) (side effect: notification of conflicts) ... trunk-n(artifact) (side effect: notification of conflicts)

分散作業空間の共有中間成果物オブジェクトを永続版空間におき、私有中間成果物オブジェクトは未来版空間におく。永続版空間から複数の未来版空間に、同じ中間成果物や、依存関係にある中間成果物がチェックアウトされた場合は、未来版空間におけるある人の改訂作業が、別の未来版空間に影響を与えることが予想される (**競合状態**)。このような影響を自動的に解析・把握することを支援する。競合解析 [7] はその基礎となる研究である。

ここで開発すべき技術は、未来版空間と永続版空間の一貫性保証である。本研究における矛盾の定義は以下の通りである。「矛盾とは、ソフトウェア中間成果物間に偶発的に発生する定義のくいちがい」であり、「汚染」とは競合解析すなわち、ある人の変更が他人におよぼす影響を指す。一貫性保証技術の開発は今後の課題である。

3.8 チーム構造オブジェクト

役割オブジェクト、コミュニケーション・パス・オブジェクト、分散作業空間オブジェクトからなる構造 (図.4) をチーム構造オブジェクトとして定義する。チーム構造オブジェクトは、プロセスの静的側面を定義するモデルであり、様々な事象の発生を引き起す土台となる。

Team : name
Roles: role-1 ... role-n
Communication-Paths: communication-path-1 ... communication-path-n
Distributed-Work-Spaces: distributed-work-space-1 ... distributed-work-space-n

3.9 オブジェクト中心の活動に対応する自在アーキテクチャ

図. 2に対応する自在アーキテクチャの設計方針を図. 7 に示す。以下のような簡単なアーキテクチャにより、ここまで述べてきた種々の機能を統一的の実現できる予定である。すなわち、クライアントは、共同作業の状況把握に関する副作用を与えるプロキシを介してCVS サーバにアクセスする。各プロキシには、状態情報がそれぞれ格納されている。

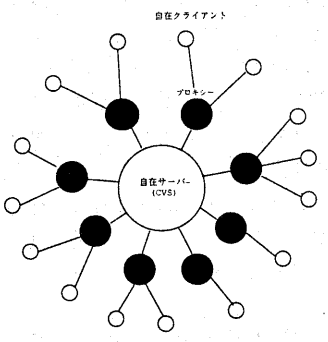

図 7: 自在環境のアーキテクチャ

4 開発状況の不安定さを共有させる情報リポジトリと Active coordinator オブジェクト

4.1 Active coordinator と実行制御機構

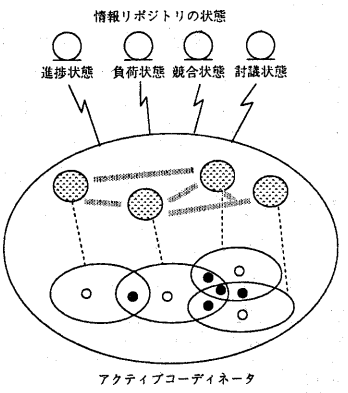

図 8: Active coordinator

チーム構造 (プロセス構造) の実行 (enact) 制御の方針を、「分散共同作業に固有の不安定さにともなって発生する矛盾や曖昧性の存在の検知」を活動のトリガーとするという立場をとる。Active coordinator オブジェクトは、役割、コミュニケーション・パス、責任範囲からなるチーム構造を内部モデルとして持

ち、矛盾や曖昧性を検知し関係者に通知する能動的主体である。

Active coordinator オブジェクトはチーム構造モデルを構成する各種オブジェクトの状態を、情報リポジトリに蓄積されたデータに基づいて設定・変更する(図. 8)。Active coordinator オブジェクトは実世界の抽象モデルであり、抽象度のレベルに応じて実世界をシミュレートする。また、チーム構造モデルの抽象レベルで表現された範囲で、解消すべき矛盾や曖昧さを検知し、オブジェクト関連活動 (object-related activity) やコミュニケーション関連活動を駆動することにより、実世界の制御を行なう。

4.2 予想検知可能事象

本節で定義したチーム構造モデルにより検知可能な事象と検知手段を予想する。

- 各アクターの作業進捗状況はワークフロー上の作業位置で把握する。また、作業の繰り返しや停滞も作業履歴情報より抽出できる。
- 新しい共有中間成果物オブジェクトの出現により、状況の進展が観測されうる。
- 中間成果物オブジェクトに割当てられた状態遷移図より作業進捗状況が把握できる。
- 中間成果物オブジェクトに割当てられたタイムアウト事象の発生により、作業または変更に時間がかかりすぎる、作成・変更したままほっておかれている、変更に対する調整作業が難航している、などが検出できる。
- 特定の人に負荷がかかりすぎているなどの、アクター間の負荷分散の適切度も負荷状態をもとに判断できる。
- ある人の変更が他人に影響を与えている度合を、未来版空間における競合状態より判断できる。
- 新しいメールが到着する毎に、そのメールは話題毎に発言に分解され、該当する討議スレッドに接続される。討議スレッドオブジェクトのタイムアウト事象の発生により、返答を要求しているのに応答しない、期限がきても物事が決まらないなどが検知可能であり、催足メッセージを発送することも可能である。また、指定された期日までに「決定」の項目が設定されないものについては警告メーッセージが発送することもできる。
- コミュニケーション・パスについて切断が発生したことを検知できる。
- 新しい話題が発生するたびに討議スレッドオブジェクトが生成される。
- チーム構造モデルオブジェクトの状態は、上記各種オブジェクトの状態設定・変更の複合効果として定義する。

5 達成点の評価と今後の課題

Coplien による組織パターン [11]、プロセスパターンを Activity Theory[12] の分類に基づき整理し、分散共同作業の支援に関する筆者の研究成果成果をあてはめると図. 9のようになる。図. 9 の実線部分についてはすでに基本的な考察が完了した部分である。図. 9の点線部については今後の課題である。とくに、各作業グループ毎に異なる規則を明示的に定義するための研究を今後進める必要がある。

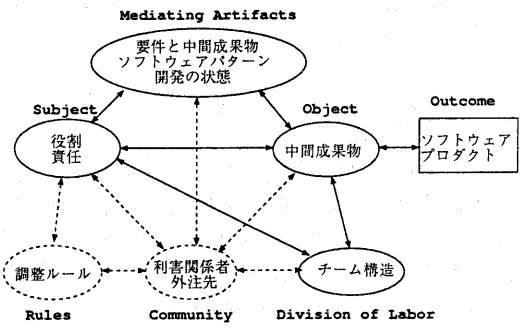

図 9: 行動モデル：ソフトウェア開発版

Subject Pattern5: 密接に関係する活動をグループ化して役割とせよ。役割に結合された活動は責任となる。Pattern13: アーキテクトの役割を開発者の役割をアドバイスし制御するものとせよ。Pattern23：文書化には専門家を雇え。

Division of Labor Pattern9: アーキテクチャの構造は開発構成員の地理的配置を反映させよ。Pattern10: マーケットの構造を開発チームの構造に反映させよ。

Object と Division of Labor 間

Pattern14:組織の構造とソフトウェアアーキテクチャの構造を一致させよ。

Subject と Division of Labor 間 Pattern11: 繰り返しが開発のキーであるときプロジェクトのコミュニケーションの中心は開発者とせよ。Pattern15: アーキテクトも実装に参加せよ。Pattern19: QA を中心的な役割として定義せよ。Pattern20: 顧客満足度を達成するには、顧客の役割をアーキテクトや開発者の役割と関連づけよ。Pattern26:コミュニケーションは役割における責任 (役割に結合された活動) 間の意味的結合に従え。適切な責任と権限を与えよ。

Rule Pattern16: 一つのモジュールは一人の開発者によって管理させよ。

Subject と Community 間 Pattern12: マネージャーには、プロジェクトレベルの障害の除去とプロジェクトの士気の維持に責任を持たせよ。Pattern24:プロジェクト外部からの干渉から開発者を防ぐことをプロジェクトマネージャーの役割として定義せよ。Pattern25:プロジェクト外部とのコミュニケーションは GateKeeper を介して行なえ。

6 おわりに

分散共同作業に固有の不安定さにともなって発生する矛盾や曖昧性の存在の検知を活動のトリガーとするという立場に基づく、分散共同作業を支援するためのソフトウェアプロセスモデルと開発支援環境についての筆者の中間研究成果を報告した。これまでの研究により定義・改良してきた、開発状況の不安定さを共有させる情報リポジトリの存在を前提として、役割、コミュニケーション・パス、責任範囲の 3 要素からなるチーム構造モデルを提案し、それに基づいて、Active coordinator によるプロセス実行制御 (調整支援) の方針ををまとめた。

謝辞

本研究は文部省科研費基盤 (C)(2)「分散共同開発に適したソフトウェアプロセスモデルに関する研究」(課題番号 12680339) の援助の基に実施された。記して謝意を表する。また、本学情報科学研究科助手 藤枝和宏氏の日頃の討論に感謝する。さらに、図. 2の分析における落水研究室諸氏の貢献に感謝する。

参考文献

[1] K.Ochimizu, C. Kadowaki, M.Hori:"Design of an Information Repository to Support Cooperative Works over a Computer Network", IPSJ International Symposium on Next-Generation of Information Technologies, September, 1997.

[2] 落水:漸増的ソフトウェア設計・実現のためのプロセスモデル -ソフトウェア分散共同開発における調整支援-, 日本ソフトウェア科学会 コンピュータソフトウェア, Vol. 15, No. 4, pp. 73–77, 1998.

[3] K.Ochimizu, M.Murakoshi, K.Fujieda, M.Fujita: "Sharing instability of a distributed co-operative work", Proc. of the ISPSE, November, pp.33–42, 2000.

[4] H.Murakoshi,A.Shimazu,K.Ochimizu: "Construction of Deliberation Structure in E-mail Communication", International Journal of Computational Intelligence, Vol.16 No.4, pp. 570–577, 2000.

[5] 村越、島津、落水 "メーリングリストを利用した共同作業における討議構造の自動構築法", コンピュータソフトウェア, Vol. No., pp. –, 2001.

[6] 藤田、藤枝、落水、:オープンソース開発におけるプロジェクトモニタリング手法, ソフトウェアシンポジウム 2000, pp. 126– 130, 2000.

[7] 林崎, 落水:"未来版管理機構における競合解析の一手法", 情報処理学会, ソフトウェア工学研究会, SE-127, pp. 15–22, 2000.

[8] 落水:"分散共同ソフトウェア開発に対するソフトウェアプロセスモデルに関する基礎考察", 電子情報通信学会, ソフトウェアサイエンス研究会資料, 2000.

[9] M.Hori, Y.Shinoda, K.Ochimizu: "Management of Future Version Spaces by Pollution Marking", Proc. of 1st IWPSE, pp. 74–83, 1998.

[10] M.Hori, Y.Shinoda, K.Ochimizu: "Management System of Future Version Space: Design and Implementation", Proc. of 2nd IWPSE, pp. 48–54, 1999.

[11] J.Coplien:"A Development Process Generative Pattern Language", Pattern Languages of Program Design, 1995.

[12] Yrjo Engestrom: "Activity theory and individual and social transformation", Perspectives on Activity Theory, pp. 19–38, Cambridge University Press, 1999.

[13] 堀, 落水:"ソフトウェア開発における自己反映オブジェクト指向モデルに基づく共有情報の管理法", コンピュータ・ソフトウェア, Vol. 13, No. 1, pp. 37–54, 1996. M.Hori, Y.Shinoda, K,Ochimizu: "Shared Data Management Mechanism for Distributed Software Development Based on a Reflective Object-Oriented Model", LNCS 1080, Advanced Information Systems Engineering, pp.362-382, 1996.