ソフトウェア工学 127－3
（2000．5．26）

未来版管理機構における競合解析の一手法

林崎 浩典　　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: {hironori,ochimizu}@jaist.ac.jp

概 要

本報では、分散共同作業における状況検知の一手段とそれに基づく調整支援の一方式についての研究成果を報告する。版管理システムにおけるある版を複数人がそれぞれ私有空間 (未来版空間) にチェックアウトして独立に作業を進める場合を例にとり、ある未来版空間に対する他のすべての未来版空間における改訂作業の影響を解析・検知する機構を提案する。機構は、オブジェクト指向システム依存グラフの差分抽出と波及解析に基づいて開発された。

キーワード： オブジェクト指向依存グラフ、差分抽出、波及解析、競合解析、版管理、未来版空間、分散共同作業、アウエアネス支援、調整支援

Impact Analysis among Future Version Spaces

Hironori Hayashizaki　　Koichiro Ochimizu

School of Information Science, JAIST
Asahidai 1-1, Tatsunokuchi, Ishikawa 923-1292

Abstract

In this paper, we will report on the awareness support of matters in a distributed cooperative work to coordinate the participants of the work. We assume the situation that multiple people check-out the same version or versions being dependent mutually from a persistent version space to private work spaces(Future Version Spaces) and revised them concurrently and independetly. We propose an mechanism that can analyze and extract the impacts of revisions in Future Version Spaces to the specific Future Version Space. The mechanism was developed based on a differential analysis and an effect analysis between two Object-oriented System Dependence Graphs.

keywords: Obeject-oriented System Dependence Graph, differential analysis, effect analysis, impact analysis, version control, future version space, distributed cooperative work, awareness support, coordination support

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。ネットワークを介した共同作業には以下の固有の問題点が存在する [15]。

「共同」作業に関する問題点：共同作業においては作業者間の合意事項や中間成果物の内容の認識に関するズレが発生し、そのようなズレは担当者の地理的分散によって拡大しその収束が遅延する。このような状況は開発状態の不安定さをもたらし、中間成果物の内容や作業のコンテキストとして存在する合意事項に多くの矛盾や不確実さを発生させる

「分散」作業に関する問題点：分散環境でソフトウェア開発を行なう場合、我々は、「互いに交信できない」という状況に対処する必要がある。そのような問題状況として、ネットワークの障害、交信相手の不在、交信相手が移動計算環境下で作業中でありネットワーク環境から切り離されている等の例がある。これ等の問題は、単に、ネットワークの接続性を改善するだけでは解決されず、他人と接続されていない状況下でも開発作業を継続させ得る機構が必要である

われわれは、共同作業の支援においては、**状況のawarenessの支援とそれに基づくco-ordinationの支援が重要である**との認識のもとに、以下のような解構築の基本方針を立てている。

- 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**)。共有された情報中に存在する矛盾や不確実さを漸次解消していけるようなモデルと支援環境を開発することにより、分散共同作業の制御方式に理論的基盤を与える。

現在、以下のような成果を得ている。中間成果物間の粗粒度レベルの依存関係を土台にして、共有中間成果物の変更管理と決定事項の管理を目標にした、ソフトウェア開発環境の**参照モデル** CSCSD モデルを定義した [14]。共同作業を通じて生成・変更されていく中間生成物や決定事項と、それらの状態を、実世界の状況を正確に反映しつつ管理する**情報リポジトリ**を設計した [9, 8]。CSCSD モデルに基づき、

- **共有中間成果物の変更管理**に関する機能とオブジェクト群を開発した。各自の仕事の責任範囲を中間成果物の集合 (タスク) としてとらえ、複数のタスクで共有される中間成果物の動的発生を作業の進行と定義する。このとき、書き込み権限のない共有中間成果物オブジェクトの内容を変更したいという要求を持つ人が，関係者とのネゴシエーションの後に，書き込み権の保有者から，書き込み権を一時委譲される**協調書き込み**の機能を開発した [3, 10]。

- また、共同作業を通じてなされる**決定事項の記録スキーマ**を開発した [9]。さらに、**討議スレッドの自動抽出法**を開発した [7, 13]。「すでに決定されたことと、これから決定されるべきこと」、「全体像に対する，現状の位置づけ」、「次に話し合うべき話題」などの会話のコンテキストを共有させることが目標である。

現在、上記**情報リポジトリの一実装**を実現中である。以下の 3 つのプログラム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検討中である。

- 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から、**話題ごとの討議スレッドを自動抽出するプログラム**

- 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**開発の定常状態と異常の予知情報を検知するプログラム** [17, 16]。

- 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**

本報告は最後の項目に関する研究の中間成果を報告するものである。

2 未来版管理機構とは

近年のソフトウェア開発では、ソフトウェアプロダクトの迅速な提供を実現するために、「予想される変更に備えて、先を見越して作業を進める」ことの支援が必要になってきている。例えば、仕様が常に改変され完了することのないネットワーク関連やPC関連分野の標準化指向のソフトウェアはそれらの代表例である。

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、「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た[11, 4]。**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。**永続版空間**は、承認された変更要求に対応する中間成果物を管理する。**変更要求空間**は**承認済変更要求**と**未承認変更要求**からなる。変更要求が承認されたとき、対応する未来版空間の中間成果物は、永続版空間にチェックインされる。

Bob Balzer によって提案され、Carlo Ghezzi によってソフトウェアプロセスに応用された汚染マーキングの理論と技術を拡張して、上記未来版空間と永続版空間の一貫性保証のための形式化の準備を整えた。改訂作業は、「変更内容は関係者に承認されていなければならない」という**前提条件**を持つ。これに違反して作業を進める場合は、中間成果物を永続版空間から未来版空間にチェックアウトして行なう。ただし**不変条件**に違反することは許されない。未来版空間での作業において、前提条件に違反してチェックアアウトされたオブジェクトおよびその派生オブジェクトには汚染マーキング規則により**汚染マーク**が付加される。これにより、一貫性と確実さを漸増的に補強しつつ、矛盾と不確実さの存在を前提とした作業遂行を支援する機能の実現について見通しを得ることができた。

以下の機能を有する未来版管理システムを開発した[12, 5]。

- 永続版空間から未来版空間へのチェックアウト
- 未来版空間から永続版空間へのチェックイン
- 未来版の構成のコピー
- 2つの未来版のマージ
- 競合の検知・解消支援
- 版の進化に関する情報の提示
  - 未来版空間と永続版空間の差の通知
  - 未来版空間と永続版空間の差分の提示

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・把握できるような機構が必要である。以下にのべる競合解析はその基礎となる研究である。

3 競合解析

競合解析は、後に説明するオブジェクト指向システム依存グラフの差分抽出と波及解析を基にして行なう。差分抽出、波及解析共に2つのオブジェクト指向システム依存グラフの対応関係の解析を土台とする。**差分抽出**は、版管理システムにおける同じブランチ上の2つのオブジェクト指向システム依存グラフ間の対応関係を解析し、対応しない部分を差として検出する。**波及解析**は、異なるブランチ上の2つのオブジェクト指向システム依存グラフ間の対応関係を基に波及を決定する。**競合**を、あるブランチのある版に対する波及の和集合として定義する。

図1は、開発者Aと開発者Bがバージョン管理システムにおけるブランチを利用し、並行して作業を進めている様子を示している。開発者Aと開発者Bが互いに相談・合意することなく、同じバージョン、または、依存関係にあるバージョンのソースプログラムに対するブランチを作成し、例えば、機能の拡張や保守などの作業を並行して進めている場合には、ブランチは未来版空間に作成されることになる。この時、例えば、開発者Aの改訂作業が開発者Bに与える影響を、開発者Bの作業空間において観測できることが望ましい。互いに影響のない部分は

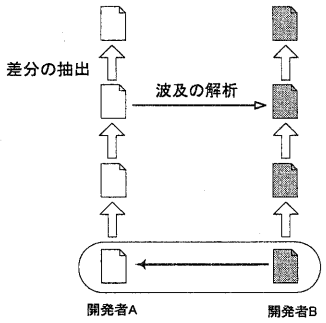

図 1: 差分抽出、波及解析、競合解析

安心して改訂作業を進めることができるし、また、影響を受ける (競合する) 部分についてはその対応を考慮にいれて作業を進めることができる。

3.1 競合解析の手順

以下に、競合解析解析の手順をまとめる。

1. **オブジェクト指向システム依存グラフの作成**
   テキストベースの差分抽出ツールとして、UNIX diff がよく知られている。しかし、テキスト行の差分を見ただけでは、変更内容を理解することが困難な場合もある。一方、開発者が理解しやすい意味的な差分を抽出する技術として、ソースプログラムの構文木に基づいた差分抽出法がある [18]。

   本研究では、依存グラフの差分抽出をおこなう。このため、ソースプログラムを構文・意味解析し、オブジェクト指向システム依存グラフ (*Object-oriented System Dependence Graph*: OSDG) に変換する必要がある。

2. **差分抽出**
   あるブランチ上の OSDG 同士の対応づけを行い、対応のない部分を差分として取り出す。

3. **波及解析**
   異なるブランチ上の OSDG 間の対応づけを行う。例えば、図 1 において、開発者 A が改訂中のシステム依存グラフと、開発者 B のブランチ上にあるシステム依存グラフとの間にメソッド呼び出しなどの依存関係がある場合を対応があるといい、この対応を抽出する処理を波及解析と呼ぶ。

4. **競合の解析**
   差分抽出と波及解析の結果を利用して、競合を検知する。競合は、あるブランチ上のバージョンに対して、すべての他のブランチからの波及の全体として定義する。

3.2 オブジェクト指向システム依存グラフ

依存グラフは、プログラムの最適化、理解などにおける問題を解決するために有効な技法として活用されている。本研究では、オブジェクト指向言語 Java を解析対象としているので、OSDG を利用する。OSDG とは、Harrold 他 [6] が提案した依存グラフにをもとに、蜂須 [2] により提案された、メンバ変数や戻り値に関して、依存関係を推移的に追跡できる等の特徴を持つ依存グラフである。

OSDG の例を図 2に示す。グラフは、クラス、メソッド、文などの構文情報から得られる節や、データ依存関係などのプログラムの意味的な情報を含む辺から構成される。本研究では、異なるブランチのクラス間で依存関係の成立を解析する必要があるため、メソッド内の局所的な解析に加えて、メソッド呼び出し関係などのメソッド間に渡る解析を行なう。例えば、図 2に示すように、メソッド呼び出しを表わす辺を作成する。

メソッド呼び出しについて OSDG を利用した処理を正確に行うためには、プログラム実行時に動的に呼び出されるメソッドを解析し、グラフを構築する必要がある。多相型によるメソッド呼び出しは、静的解析によっては完全に解析できない。現在、データフローを解析し、変数に代入されているオブジェクトの型を決定する手法や、インスタンス化されていないクラスを呼び出しの対象からはずす Rapid Type Analysis[1] 等の手法を用いて近似し、精度を向上させる手法を検討中である。

3.3 OSDG 間の差分抽出

OSDG 間の差分抽出は、2 つの OSDG の共通な節を求めることにより実現する。その対応づけは、

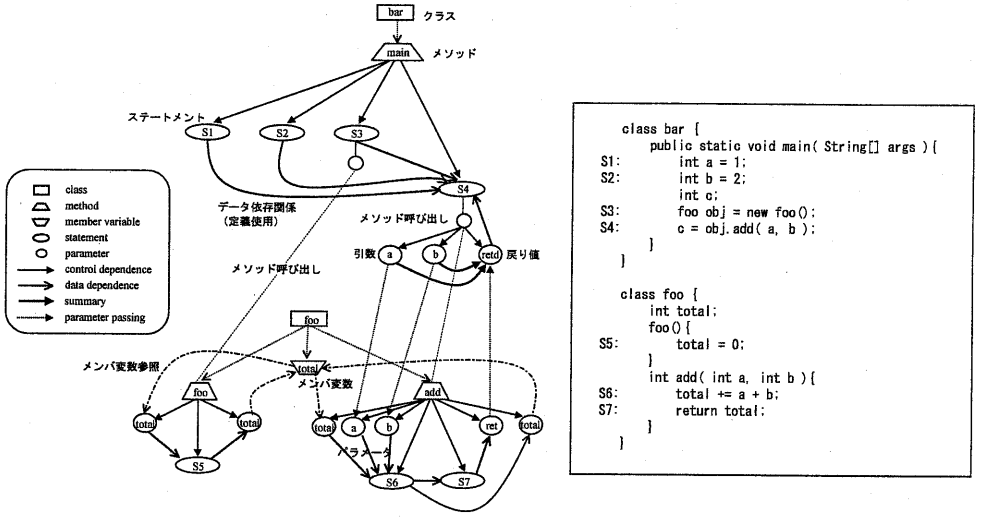

図 2: OSDG の例

以下に示す 3 つのステップを順に行うことで実現する。

1. **クラス名**
   まず、クラス名を比較し、名前が同じ場合には対応づける。図 3 において、クラス`foo` の間を結ぶ辺が、対応箇所を示している。

2. **メンバ変数名、メソッドのシグネチャ**
   つぎに、クラスを構成するメンバ変数やメソッドの対応づけを行う。メンバ変数は、クラスの場合と同様に、単純に名前のみの比較を行う。図 3の例においては、メンバ変数`var` 間に対応がつけられる。また、メソッドの場合は、シグネチャが同じである場合に対応をつける。

3. **メソッド内の文**
   文の場合にはメソッド名を根として、そこから葉に向かって、同じ頂点であるかどうか調べて行く。例えば、図 3の例においては、*add* を頂点とするサブツリーを比較する。最初に $S1$ と $S3$ を比較する。$S3$ は`if` 文なので $S1$ と一致せず、それ以降も一致する文はない。よって、挿入された文であると判断し、さらに、$S3$ から制御依存のある節も同じく追加された文とみなす。$S2$ と $S6$ は一致する。

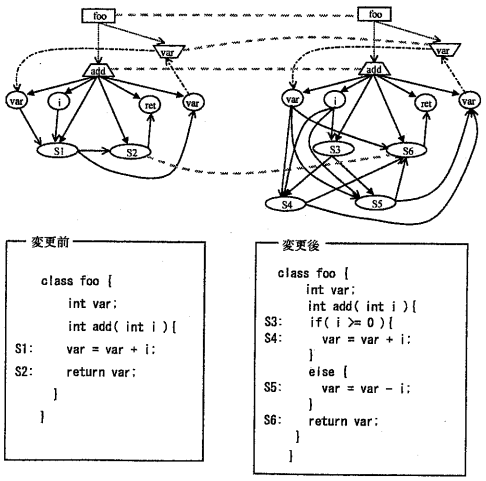

図 3: OSDG の対応づけ

以上のステップを踏むことにより、対応が無いと判断された節を、以下のように解釈する。変更作業の前のグラフにおいては、削除された節であると判断し、変更後のグラフにおいては、挿入された節であると判断する。また、名前 a が名前 a' に変更された場合、現在のところ、a が削除され、a' が追加され

たと判断するようにしている。今後、改良の必要がある。

3.4 ブランチ間の波及解析

異なるブランチ上のOSDG間の対応関係として、変更の波及があるような依存関係を抽出する。そのような依存関係として、以下を考えている。

- **メソッド呼び出し:** あるブランチ上のクラスのメソッドが、別のブランチ上のクラスのメソッドを呼び出している。
- **メンバ変数の参照:** あるブランチ上のクラスのメソッドが、別のブランチ上のクラスのメンバ変数を参照している。
- **継承:** あるブランチ上のクラスが、別のブランチ上のクラスを継承している。

3.5 差分抽出と波及解析に基づく競合の検知

競合の検知とは、あるバージョンに対する波及の全体を検知することとする。図4の例において、開発者Aと開発者Cのシステム依存グラフには変更が加えられている。この結果、開発者Bのプログラムに対して波及効果1,2,3の影響を及ぼす結果となっている。波及効果1,2,3の和集合が競合である。現在、競合解析については、下記に示す簡単なもの

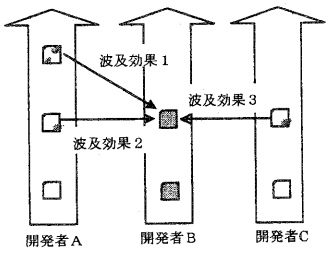

図 4: 競合の検知

として定義している。これ以外にも、プログラムの動作に影響を与える変更は考えうるが、今後の課題である。

クラス宣言の変更

- **挿入または削除**
  クラス、メソッド、メンバ変数などを参照している場合に、参照している名前の挿入・削除により影響を受ける。例えば、メソッドのシグネチャを変更した場合は、それまでの呼び出し法ではアクセスできなくなる。
- **クラスメンバの型**
  メソッドの戻り値の型やメンバ変数の型が修正されたときは影響を受ける。プリミティブ型のメンバ変数を参照型へ修正した場合を考える。その変数を参照しているプログラム文は、型が異なるので、そのメンバ変数への整合性が保てない。
- **スコープの変更**
  クラス、メソッド、メンバ変数のスコープが修正された場合には影響を受ける。例えば、スコープが`public`から`protected`に変更されたときにはアクセスが不可能になる場合がある。

継承・実装関連の変更 クラスの継承・実装関係が変更された場合、型の互換性を失い、呼び出すメソッドが意図するものと異なる動作をする恐れがある。

戻り値に影響を与える変更 メソッドの戻り値に影響を与えるような変更が行われた場合、戻り値（例えば、参照型など）が利用者の期待と異なるかもしれない。図5に、図3のプログラム例を使って戻り値に影響する変更例を示す。検知法としては返り値の型が`void`型を除き、*ret*パラメータからデータ依存関係にある文を全て取り出し、その中から挿入または削除された文を探し出せばよい。図5においては、濃い色の文が取り出される。

4 実装

現在のところ、競合検知の部分的な解析（クラス宣言の変更）を対象として、部分的実装がなされている。

- システム依存グラフの解析は、クラス、メンバ変数、パラメータ、メソッド呼び出し、メンバ変数の参照については実装したが、ステートメ

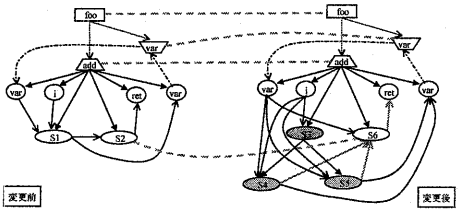

図 5: 戻り値に影響する変更例

ントの解析およびデータ依存関係は未実装である。

- 差分抽出については、クラス、メンバ変数、メソッドについて実装したが、ステートメントについては未実装である。

- 波及解析については、メソッド呼び出しおよびメンバ変数の参照の参照については実装したが、継承については未実装である。

- 競合検知については、クラス宣言の変更については実装したが、戻り値の変更および継承の変更については未実装である。

プロトタイプシステムでは、変化しつつある状況をブランチ間において追跡可能にするユーザ・インタフェースを実現した。このユーザ・インタフェースには、ソースプログラムへのマーキング機能、詳細情報提示機能が備わっており、数多くある競合の種類を利用者は容易に見分けることができる。ここで、

- マーキング機能は、ソースプログラム中の影響を受ける部分をマーキングすることで、影響の存在を利用者に通知する。図6は、出力結果の例を示しており、WWW ブラウザの右フレーム内の HTML にハイパーリンクとして記述されている。

- 詳細情報提示機能は、マーキングされた部分に対応する競合箇所、バージョン名、競合の発生理由を表示する (図6の左下フレーム)。さらに、変更の影響を与える側と受ける側 (左上フレーム) を対応させて表示することで、比較しながらの把握を可能にしている。

実装には CASE ツールプラットホームの Japid[2] を利用した。最終的な出力は、HTML として得られるため WWW ブラウザ上で閲覧可能である。

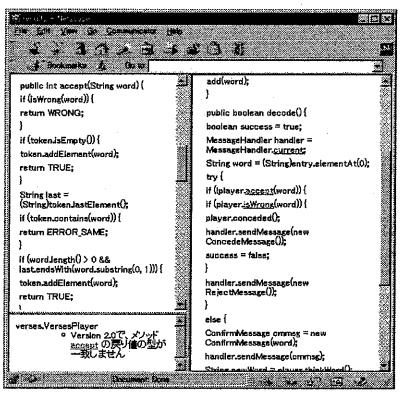

図 6: 実行例

5 おわりに

われわれは、オブジェクト指向方法論[1]を前提としながらも、決定事項の不確実さや中間成果物間に存在する矛盾の存在を前提とした共同作業の遂行を支援するソフトウェア開発環境の構築を目指している。次の目標は、**一貫性と確実さの漸増的補強を可能にするようなソフトウェアプロセスモデルを開発し、矛盾と不確実さの存在を前提とした作業遂行に関して理論的基盤を与えることである。**このような、ソフトウェアプロセスモデルの構築にあたっては、認識のズレの認知とその結果の中間生成物への反映、共有情報の動的発生や変更による関係者間のコミュニケーションの発生等のセマンテイクスを表現する融合ソフトウェアプロセスモデルを開発する必要がある。

参考文献

[1] David Francis Bacon. *Fast and Effective Optimization of Statically Typed Object-Oriented*

[1]ここでいうオブジェクト指向方法論とは、SRA の青木氏が定義したコンセプト「現実世界をそのまま計算機の中に投影し、投影された仮想世界を現実世界とのミスマッチや現実世界の進化に応じて逐次的、漸増的に修正・進化させていく」を指す。

Languages. PhD thesis, Computer Science Divison, University of California, Berkeley, 1997.

[2] Yoshinari Hachisu. *A CASE Tool Platform for Object-Oriented Program - Japid: Its Design and Implementation.* PhD thesis, School of Engineering, Nagoya University, 1999.

[3] Masakazu Hori, Yoichi Shinoda, and Koichiro Ochimizu. Shared data management mechanism for distributed software development based on a reflective object-oriented model. In *LNCS 1080, Advanced Information Systems Engineering*, pages 362–382, 1996.

[4] Masakazu Hori, Yoichi Shinoda, and Koichiro Ochimizu. Management for future version space by pollution marking. In *Proc. of the 1st International Workshop on the Principle of Software Evolution*, pages 74–83, 1998.

[5] Masakazu Hori, Yoichi Shinoda, and Koichiro Ochimizu. Management system of future version space: Design and implementation. In *Proc. of the 2nd International Workshop on the Principle of Software Evolution*, pages 48–54, 1999.

[6] L. D. Larsen and M. J. Harrold. Slicing object-oriented software. In *Proc. 18th Int. Conf. Software Engineering*, pages 495–505, 1996.

[7] Hiroyuki Murakoshi, Akira Shimazu, and Koichiro Ochimizu. Construction of deliberation structure in e-mail communication. In *Proc. of Pacific Association for Computational Linguistics*, pages 16–28, 1999.

[8] Koichiro Ochimizu. Toward a software process model for distributed cooperative software development. In *Proc. of the 2nd International Symposium on Future Software Technology*, pages 55–62, 1997.

[9] Koichiro Ochimizu, Chie Kadowaki, and Masakazu Hori. Design of an information repository to support cooperative works over a computer network. In *Proc. of the IPSJ International Symposium on Next-Generation of Information Technologies*, pages 79–86, 1997.

[10] 堀 雅和 and 落水 浩一郎. 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):37–54, 1996.

[11] 堀 雅和, 篠田陽一, and 落水 浩一郎. 汚染マーキングによる未来版空間の管理. In ソフトウェア・シンポジウム *'98*, pages 161–170, 1998.

[12] 堀 雅和, 篠田陽一, and 落水 浩一郎. 未来版管理システムの設計と実装. In ソフトウェア・シンポジウム *'99*, pages 8–17, 1999.

[13] 村越 広享, 島津 明, and 落水 浩一郎. 電子メールを利用したコミュニケーションにおける討議の流れの自動抽出法について. In 情報処理学会自然言語処理研究会資料 *127-2*, pages 31–36, 1998.

[14] 落水 浩一郎. ネットワークを介した共同作業に対する支援環境の参照モデルに関する一考察. In 情報処理学会ソフトウェア工学研究会資料 *115-10*, pages 73–80, 1997.

[15] 落水 浩一郎. 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 –. コンピュータソフトウェア, Vol.15(No. 4):73–77, 1998.

[16] 藤田 充典, 藤枝 和宏, and 落水 浩一郎. 共同開発の作業履歴から観測される事象とその考察. In 電子情報通信学会 ソフトウェアサイエンス研究会 *SS99-34*, pages 33–40, 1999.

[17] 藤田 充典, 藤枝 和宏, and 落水 浩一郎. オープンソース開発における作業履歴からの情報の抽出と視覚化. In 電子情報通信学会 ソフトウェアサイエンス研究会 *SS99-50*, pages 1–8, 2000.

[18] 吉田 敦, 山本 晋一郎, and 阿草 清滋. 意味を考慮した差分抽出ツール. 情報処理学会論文誌, *Vol.38, No.6*, pages 1163–1171, 1997.