ソフトウェア工学 127－1
（2000．5．26）

ワークフローと構成管理の統合環境を実現したソフトウェア開発支援システム

煤孫 統一郎† 倉谷 和彦††

本稿では，分散環境でのソフトウェア開発を支援するための，ワークフローと構成管理の統合環境を提案する．実際のソフトウェア開発事例を調査し，議事録が重要な役割を果たしていることを確認した．議事録に注目して，議事録をシステムに取り込んだワークフローと構成管理の統合環境のモデルを構築した．さらに，モデルを実現するシステムを開発した．本システムの評価として，ソフトウェア開発3事例について思考実験を行った．担当者間の確実な情報伝達と成果物間の整合性確保ができなかった例として，ある障害の修正が新たな障害を引き起こした現象に注目した．これらの障害に本システムを適用した場合の回避可能性を考察した．この思考実験の結果から，本システムの有効性が期待できる．

A Software Development Support System which Integrates Workflow and Configuration Management

TOICHIRO SUSUMAGO† and KAZUHIKO KURATANI††

This paper proposes a software development support system, which integrates workflow and configuration management. We found out that "minutes" played an important role through the case study of three software development projects. We made a workflow and configuration management model, which contains "minutes". We developed and verified a software development support system which is based on the model. The verification shows that the system is effective in software development.

1. はじめに

分散環境でのソフトウェア開発は，担当者間の確実な情報伝達と，仕様書，プログラムソースなどの整合性確保が困難である．このことに起因するソフトウェアの品質低下が大きな問題となっている．特に，関係者が大人数で複数の会社に所属し，地理的に分散しているようなソフトウェア開発では，以下の理由で障害が多発する．

- 変更通知が，関係者に確実に配布されない．
- 配布された通知内容が自分に関係するものなのか，判断が難しい．
- 変更された仕様書，ソースがすぐに入手できない．
- 関連部分の変更が確実に行われたのか，管理者側で把握しきれない．

上記の問題を解決するためには，担当者間の確実な情報伝達と，成果物☆間の整合性確保を同時に行うことが重要である．ワークフローは担当者間の情報伝達を確実に行うために有効である．構成管理は成果物間の整合性を確保するために有効である．本研究では，ワークフローと構成管理の統合環境を提案する．

統合環境の構築に先立ち，実際のソフトウェア開発の事例調査を行った．事例調査から議事録等の中間成果物が重要な役割を果たしていることを確認した．この結果に基づき，議事録を成果物として取り込んだモデルを構築した．このモデルを実現するソフトウェア開発支援システムを設計，実装，評価した．

以下本稿では，関連研究，事例調査，統合環境のモデル，統合環境の実装，統合環境の評価について述べる．

† 富士通エフ・アイ・ピー株式会社
Fujitsu FIP Corporation
†† 中電コンピューターサービス株式会社
Chuden Computer Service Co.,Ltd.

☆ 本稿では，仕様書，プログラムソース，実行形式ファイル，議事録等，ソフトウェア開発で生成されるものを総称して成果物とする．

2. 関連研究

関連研究には以下の3種類があった.

- ワークフローによるソフトウェア開発支援の研究
- 構成管理によるソフトウェア開発支援の研究
- ワークフロー・構成管理によらないソフトウェア開発支援の研究

本章ではこれらの研究を紹介し，本研究との相違点について述べる．最後に本研究の特徴を述べる．

2.1 ワークフローによるソフトウェア開発支援の研究

ワークフローによるソフトウェア開発支援の研究に，オブジェクト指向開発方法論を適用したソフトウェア開発プロセス支援システムの研究がある[5][6][7].

この研究では，開発ドキュメント管理機構とワークフローシステムにより，各種ドキュメント（プロダクト）の開発状況という視点でプロジェクトの進捗を管理する．開発ドキュメント管理機構がドキュメント間の依存関係を管理し，それらの開発状況を制御する．ワークフローシステムが，開発作業の流れを支援する.

中規模のソフトウェア開発の支援を想定していること☆，成果物の依存関係を管理する機能がワークフローと関連づけられていることなど，本研究との共通点が多い．しかし，オブジェクト指向開発方法論を適用してドメインモデルを構築した点が異なる．また，ドキュメントをベースとしたワークフローの視点から管理されていて，構成管理についての記述はない．成果物には仕様書，プログラムソース，テストスーツの例があげられているが，議事録等の中間成果物についての記述はない.

2.2 構成管理によるソフトウェア開発支援の研究

構成管理によるソフトウェア開発支援の研究に，発展型開発を支援するソフトウェア開発支援環の研究がある[2][3][4].

発展型開発とは，ベースラインとなる標準システムに対して変更を加え，複数の製品を開発する開発形態である．このシステムでは構成管理を基盤として，成果物の変更の管理機能，ソフトウェア・メトリックスの計測機能，開発者間の情報共有機能を実装・評価している.

構成管理の機能を実装していることは本研究と共通する．しかし，発展型開発の支援を目的としている点は異なる．また，ワークフローの機能・視点についての記述や，議事録等の中間成果物についての記述はない.

2.3 ワークフロー・構成管理によらないソフトウェア開発支援の研究

ワークフロー・構成管理によらないソフトウェア開発支援の研究に，プロセス・プロダクト・プロジェクトの視点から，ソフトウェア開発のマクロ構造モデルを構築した研究がある[1].

このモデルでは，ワークフローの機能を実装せず，あそびの概念と管理水準の設定によるソフトウェア開発支援環境を実現している．あそびの概念と管理水準の設定とは，重複や手戻りを許容し，設定された管理水準より詳細なレベルでは実行順序などを規定しないことである．ワークフロー機能を実装しなかったのは，妥当な管理水準を設定し，管理できることと管理できないことを明確にすることが重要であるという認識による.

分散環境でのソフトウェア開発の支援を目的としている点は本研究と共通する．しかし，ワークフローの機能を実装しなかった点が異なる．本研究では，分散環境でのソフトウェア開発で問題となる担当者間の情報伝達に，ワークフローが有効であるという認識に基づき，ワークフローと構成管理の統合環境を実装した.

2.4 本研究の特徴

関連研究から以下の点が本研究の特徴といえる.

第1に，議事録等の中間成果物に注目したことである．本研究ではソフトウェア開発の事例調査から，議事録等の中間成果物が重要な役割を果たしていることを確認し，それらに注目したモデルを構築した（3章，4章で詳述）．このような研究は従来なかった.

☆ 大規模プロジェクト全体の管理は，既存のプロジェクト管理ツールと連携して行う.

第2に，ワークフローによる管理の視点と構成管理の視点の両方があることである．本研究では，両方の視点から管理できる統合環境を実現した（4章，5章で詳述）．このような研究は従来ほとんどなく，あってもどちらかの機能が弱かった．

第3に，ワークフローと成果物管理を関連づけていることである．多数の成果物が生成されるプロジェクトでは，ワークフローと成果物管理の関連づけが重要になる．本研究ではワークフローと構成管理を同時に実現するので，ワークフローと成果物管理は常に関連づけられている（4章，5章で詳述）．

3. ソフトウェア開発の事例調査

3.1 事例調査の概要

統合環境のモデル構築に先立って，実際のソフトウェア開発事例について調査を行った．関係者が複数の会社に所属し，地理的に分散しているソフトウェア開発を3事例調査し，成果物の概要，関係者，成果物が生成・改訂・参照されるフェーズ，成果物間の相互関係，成果物管理の問題点，ソフトウェア・プロセスの標準的モデル（ウォータフォール・モデル，スパイラル・モデル）との相違点を検証した．

調査にあたって，開発規模がピーク時で10名から100名の中規模のソフトウェア開発を選択した．調査したソフトウェア開発の概要を表1に示す．

開発規模が1名から数名程度の小規模のソフトウェア開発では，分散環境でのソフトウェア開発の問題がそれほど大きくない．分散環境でのソフトウェア開発の問題が大きくなるのは，中規模以上のソフトウェア開発である．大規模のソフトウェア開発支援は，中規模のソフトウェア開発支援の発展と考えられる．したがって本研究では，中規模のソフトウェア開発の支援を想定した．

3.2 事例調査の結果

3.2.1 成果物は何回も改訂される

調査したソフトウェア開発事例において，各種成果物はウォータフォール・モデルに則した時期に生成されていた．しかし成果物はその後何回も改訂されていた．

表1 事例調査したソフトウェア開発の概要

システム	小売業者向け店舗システム	メーカ向け生産ライン制御システム	リース会社向け基幹システム
開発言語	COBOL Visual Basic	MS Visual C++	COBOL
プログラム・ステップ数	約 300K	約 600K	約 900K
ピーク時の開発人数	15 名	40 名	100 名
人月	約 50 人月	約 150 人月	約 5 年

小売業者向け店舗システムにおける，成果物が生成・改訂・参照されるフェーズを図1に示す☆．この図から以下のことがわかる．

クライアント側の仕様書類は設計フェーズで生成されているが，改訂・参照は実運用フェーズまで行われている．ソースプログラム群，実行ファイル群は，開発フェーズで生成されているが，改訂・参照は実運用フェーズまで行われている．障害連絡票は設計フェーズから実運用フェーズまで生成・参照されている．打合せ議事録は全てのフェーズで生成・参照されている．このような傾向は他の成果物，他のプロジェクトについても同様にみられた．

このように，成果物はウォータフォール・モデルに則した時期に生成されるが，その後何回も改訂されること，打合せ議事録は全てのフェーズで生成・参照されていることが観察された．

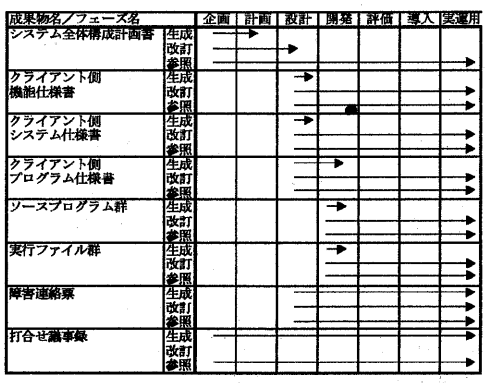

図1 成果物が生成・改訂・参照されるフェーズ一覧

☆ このシステムはクライアント／サーバ型システムである．この図では，クライアント側の成果物を主に抜粋してある．

3.2.2 成果物は作業結果にあわせて変更される

作業指示が，正規の成果物のみで行われることはまれであった．実際にはレビューの議事録，ホワイトボードのコピーなどによって，作業指示が行われていた．

そのため，成果物間の整合性が確保されないまま作業が進められていた．その結果，作業結果にあわせて成果物を変更するケースが多く観察された．たとえばシステム試験完了の後，システム設計書が現在のオブジェクトコードと整合するように改訂されていた．

3.2.3 中間成果物の存在

ウォータフォール・モデルにおける正規の成果物の他に，ヒアリング・メモ，技術検討メモ，議事録などの中間成果物（以下まとめて議事録と記す）が作成されていた．

議事録には以下の特徴があった．

- レビュー結果等の重要な情報が記載されている．
- 正規の成果物の内容を補完している．
- 作業指示書として使われる場合がある．
- 作業確認書として使われる場合がある．

議事録の内容が正規の成果物に正確に反映されず，後で問題が起こる場合がある．このようなときに，議事録までさかのぼって確認するケースが多く観察された．

このように実際のソフトウェア開発では，議事録が重要な役割を果たしていることを確認した．

3.3 事例調査の考察

実際のソフトウェア開発では恒常的に仕様変更が発生し，成果物間の整合性の確保が難しくなる．事例調査では，この不整合を議事録で補っているケースが多く観察された．

このようなソフトウェア開発の状況を考えると，議事録に注目したソフトウェア開発支援システムを構築することが，現状に適したソフトウェア開発の支援になると考えられる．

4. 統合環境のモデル

事例調査の結果をふまえ，議事録に注目したワークフローと構成管理の統合環境のモデルを構築した．

4.1 ソフトウェア開発の定義

ソフトウェア開発を，成果物を変更（生成，修正，削除）することと定義する．ソフトウェア開発における作業とは，ある成果物が参照している成果物が変更された場合，整合性を確保するように成果物を変更することと定義する．

成果物がシステムに入力されると，自分が担当している成果物であっても変更することはできない．成果物を変更できるのは，参照している成果物が変更されるか，議事録の変更（生成）によって成果物の変更を要求された場合だけである．

4.2 アクティビティの定義

アクティビティを，一人の担当者が連続して行う作業の単位と定義する．アクティビティにおける作業とは，そのアクティビティに割り当てられた成果物が参照している成果物が変更（生成，修正，削除）されたときに，そのアクティビティに割り当てられた成果物を変更することと定義する．

原則として，1つのアクティビティに1人の担当者，1つの成果物が割り当てられる．アクティビティ，成果物，担当者は1：1：1である．1つのアクティビティに1人の担当者を割り当てるのは，責任を明確にするためである．複数の担当者がいる場合には，責任者を割り当てる．1つのアクティビティに1つの成果物を割り当てるのは，ワークフロー・プロセスの管理と成果物の依存関係の管理を関連づけるためである．

4.3 レビュー・アクティビティと議事録の定義

本モデルでは，議事録をシステムに取り込むために，議事録を成果物とするレビュー・アクティビティを定義する．レビュー・アクティビティはワークフロー・プロセスの任意の位置に定義できる．成果物を変更する場合には，レビュー・アクティビティを定義し，議事録と成果物の依存関係を定義しなくてはならない．これによりレビューがワークフロー・プロセスのアクティビティとして管理される．同時に議事録が成果物としてシステムに入力される．

レビュー・アクティビティと議事録には以下の特徴がある．

レビュー・アクティビティの特徴

通常のアクティビティはワークフロー・プロセスの直前のアクティビティによって起動されるが，レビュー・アクティビティは他のアクティビティによらずに起動することができる．このためレビュー・アクティビティはワークフロー・プロセスの任意の位置に定義できる．

議事録の特徴

本モデルでは，成果物を変更（生成，修正，削除）する場合は，その成果物が参照している成果物が変更されない限り，変更することができない．しかし，議事録が生成されてその成果物との依存関係が定義されれば，成果物を変更することができる．したがって，議事録の生成が他の成果物の変更のトリガーとなる．

4.4 リリースの定義

本モデルでは，アクティビティのバージョンをリビジョン，ワークフロー・プロセスのバージョンをリリースと定義する．例を図2に示す．

(1) レビュー・アクティビティが定義され，アクティビティ1R1が起動される．
(2) アクティビティ1R1に変更がおこる．
(3) アクティビティ1のリビジョンは2になる．
(4) 下流のアクティビティもワークフローの定義にしたがって順番に起動される．

新しいリリースのワークフロー・プロセス（アクティビティ1R2→アクティビティ2R2→アクティビティ3R2）はリリース2として管理される．

このようにワークフロー・プロセスをリリース管理することによって，ワークフローと構成管理を同時に実現する．

5. 統合環境の実装

5.1 統合環境の概要

本モデルをWebベースのシステムとして実装した．システム構成の概要を図3に示す．サーバのOS上に，Webサーバ，アプリケーション・サーバ，データベースが常駐している．ワークフローの定義情報，成果物間の依存関係定義情報はデータベースに格納し，ワークフロー&構成管理エンジンはこれらを参照・更新しながら処理を行う．分散環境での利用を前提として，クライアント側はブラウザのみとした．

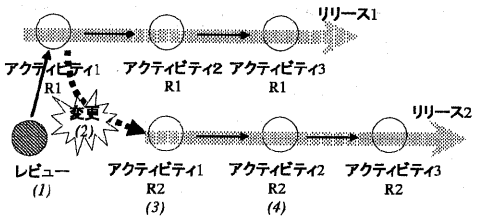

図2 ワークフロー・プロセスのリリース管理

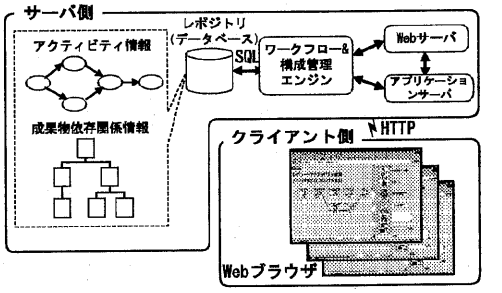

図3 システム構成の概要

5.2 レビュー・アクティビティ登録機能

レビュー・アクティビティ登録画面を図4に示す．

画面左側の領域にワークフローの一覧を表示する．ワークフローの一覧の中から，レビュー・アクティビティを定義する位置を選択する．アクティビティのアイコンをクリックすると，画面右側の領域にアクティビティ名と担当者名の情報が表示される．納期等を入力し，議事録を登録する．

この機能により，レビュー・アクティビティをワークフローの任意の位置に定義し，議事録をシステムに入力することができる．

5.3 リリース・モニタリング機能

リリース・モニタリング画面を図5に示す．

画面左側の領域にワークフローの一覧を表示する．デフォルトで最新のリリースが表示される．リリースアップボタンをクリックすると，リリース番号が一つ新しいリリースが表示される．同様に

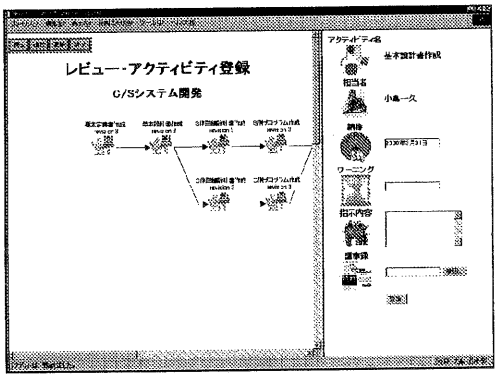

図4　レビュー・アクティビティ登録画面

リリースダウンボタンをクリックすると，リリース番号が一つ古いリリースが表示される．

アクティビティのアイコンをクリックすると，画面右側の領域にアクティビティの詳細情報が表示される．成果物名をクリックすると，画面右側の領域に成果物が表示される．図5では，レビュー・アクティビティの詳細情報が表示されている．レビュー・アクティビティの成果物は議事録なので，成果物名は「議事録」と表示されている．成果物名をクリックすると，議事録が表示される（図6）．

この機能により，各リリースを起動したレビュー・アクティビティの議事録を，容易に検索・参照することができる．

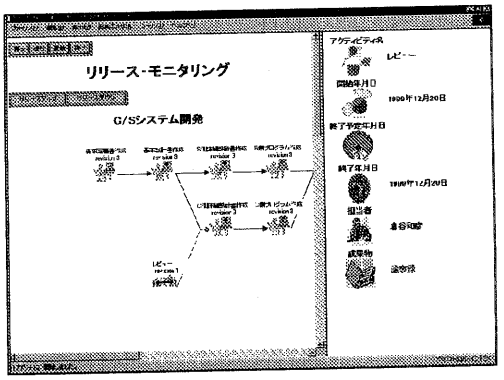

図5　リリース・モニタリング画面

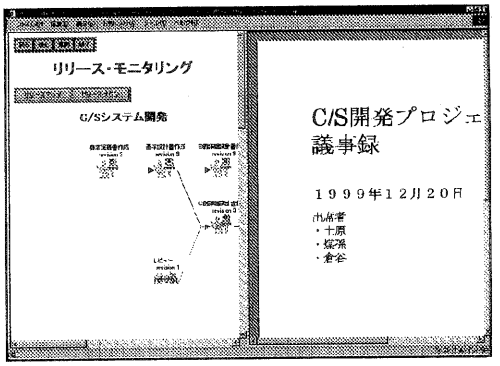

図6　議事録の表示

6. 統合環境の評価

6.1 評価の概要

本システムの評価として，ソフトウェア開発3事例について思考実験を行った．担当者間の確実な情報伝達と成果物間の整合性確保ができなかった例として，ある障害の修正が新たな障害を引き起こした現象（以下2次障害と書く）に注目した．これらに本システムを適用した場合の障害の回避可能性を考察した．

6.2 2次障害を伴った障害事例

2次障害を伴った障害事例を以下に示す．この事例は，販売サブシステムと発注サブシステムの間で起こった障害である．

販売サブシステムの売価決定ルーチンは，インタフェースを公開していた．発注サブシステムから販売サブシステムの売価決定ルーチンを call したところ，Visual Basic のスタック・オーバフロー☆の障害が発生した．この時，販売計画サブシステムと発注サブシステムの開発リーダ同士が個別に相談し，以下の対応をした（図7）．

(1)　共通変数領域☆☆を使ってパラメータを渡すことにし，販売プログラムと発注プログラムを修正した．

(2)　**この変更を仕様書に反映しなかった．**

この後，以下の仕様変更があった．当初，商品詳細情報テーブルの売価フィールドの属性は金額型

☆ 引数にある一定バイト以上のデータが渡されたときに発生するエラーである．

☆☆ 複数の画面から更新・参照が可能な変数領域．

と規定され，null 値は含まれなかった．売価フィールドに null 値が含まれることに仕様が変更された．この時，以下の 2 次障害が発生した（図 8）．

(1) 仕様変更通知が回覧された．

(2) 販売設計書の担当者は，仕様変更の通知を見て null 値に対応するよう設計書を修正した．

(3) 発注設計書の担当者は，仕様変更の通知を見たが設計書を修正しなかった．

(4) 発注プログラムから，売価フィールドに null 値が入ったデータが共通変数領域に書き込まれた．

(5) 販売プログラムは，売価フィールドの null 値が入ったデータを参照した．

(6) null 値への四則演算によって，Visual Basic のランタイム・エラー☆が発生した．

2 次障害が発生した原因は，仕様書への反映漏れである．発注プログラムと販売プログラムを修正したときに，共通変数領域を使ってパラメータを渡すことを設計書に反映しなかった．そのため発注設計書の担当者が仕様変更の通知を見たときに，仕様変更が自分の担当する成果物と関係ないと判断した．その結果発注プログラムは修正されず，障害が発生した．

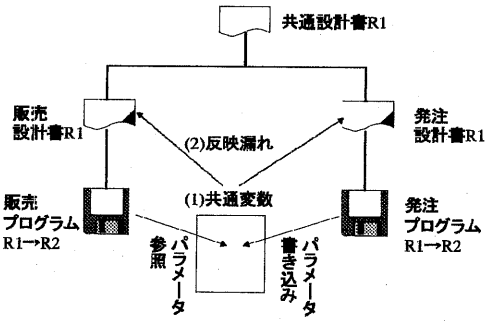

図 7　2 次障害を伴った障害事例（障害の修正）

6.3 本システムの適用事例

この障害に本システムを適用した場合，以下のように障害が回避できる（図 9）．

障害の修正には，次のように対応する．

☆ 当時の Visual Basic では，このエラーが発生するとアプリケーション全体がダウンした．

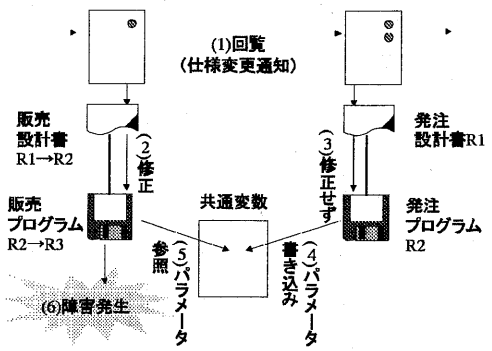

図 8　2 次障害を伴った障害事例（2 次障害の発生）

(1) 販売プログラムと発注プログラムを修正するために，議事録をシステムに入力して販売設計書と発注設計書が変更される．

(2) 販売設計書と依存関係がある販売プログラムと，発注設計書と依存関係がある発注プログラムが修正される．

この結果，売価フィールドに null 値が含まれる仕様変更に対して，以下のように 2 次障害が回避される（図 10）．

(1) 議事録をシステムに入力し，共通設計書を変更する．

(2) 共通設計書と依存関係のある販売設計書と発注設計書が変更される．

(3) 販売設計書と依存関係がある販売プログラムと，発注設計書と依存関係がある発注プログラムが修正される．

(4) 販売プログラムと発注プログラムが null 値に対応したので，障害は発生しない．

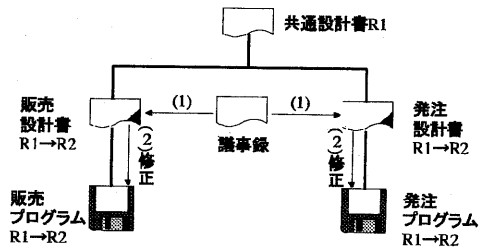

図 9　本システムの適用事例（障害の修正）

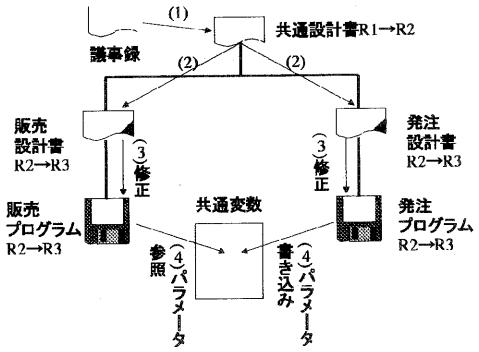

図 10　本システムの適用事例（2 次障害の回避）

6.4　評価の結果

以上のような思考実験を，2 次障害を伴った全ての障害について行った．上記の事例で示したように，本システムを 2 次障害を伴う障害に適用した場合，以下の理由により障害は回避される．

- 成果物を修正するためには，その成果物が参照している成果物を先に修正しなくてはならない．
- 成果物への変更要求が，議事録としてシステムに取り込まれ参照可能になるため，配布された変更通知が自分に関係するものなのか判断が容易になる．

思考実験の結果を表 2 に示す．2 次障害が発生した障害のうち，50%以上が回避可能であった．これらの結果から，担当者間の確実な情報伝達と成果物間の整合性確保に対して，本システムが有効であると考えられる．

表 2　本システムを適用した場合の障害回避率

	小売業者向け店舗システム	メーカ向け生産ライン制御システム	リース会社向け基幹システム
A. 障害票総数	768	1003	464
B. 2 次障害を伴う障害票数	33	54	18
C. 回避可能な障害票数	19	45	13
D. 障害回避率 (C/B)	57.6 %	83.3 %	72.2 %

7. 結　　論

ソフトウェア開発プロジェクトの事例調査から，議事録が重要な役割を果たしていることを確認した．議事録に注目し，ワークフローと構成管理の統合環境のモデルを構築した．本研究におけるモデルは，議事録等の中間成果物に注目したことと，ワークフローによる管理の視点と構成管理の視点の両方があることに特徴がある．思考実験により，本モデルが 2 次障害を伴う障害に有効であることを検証した．この結果，分散環境でのソフトウェア開発において問題となる，担当者間の確実な情報伝達と成果物間の整合性確保に対して，本モデルが有効であると考えられる．

おわりに

本研究開発は，(株) 情報技術コンソーシアムが情報処理振興事業協会の委託を受け実施したものである．筆者らは (株) 情報技術コンソーシアムに 2 年間出向し，本研究開発に従事した．実施にあたって，京都大学，垂水浩幸 助教授のご指導および (株) 三菱総合研究所のご協力を得たことに感謝致します．

参 考 文 献

1) 青山幹雄，吉松朋聖: 分散並行開発支援環境 PRIME の開発と実践, 情報処理学会研究報告, SE-96-109, pp. 41–48 (1996).
2) 佐々木幹郎, 田村直樹，柳生理子：ソフトウェア再利用環境 EvoMan における構成管理機能, 情報処理学会研究報告, SE-122-12, pp. 87–92 (1999a).
3) 佐々木幹郎, 田村直樹: ソフトウェア開発環境 EvoMan における構成管理機能の評価, 情報処理学会研究報告, SE-124-12, pp. 81–86 (1999b).
4) 柳生理子，田村直樹，佐々木幹郎: S/W 部品庫 EvoMan における品質管理機能, 情報処理学会研究報告, SE-120-19, pp.133–140 (1998).
5) 山本里枝子, 吉田裕之: プロジェクト管理ツールとプロセス管理ツールの連携の実現, 情報処理学会研究報告, SE-102-26, pp. 31–36 (1995a).
6) 山本里枝子, 上原忠弘, 森偉作, 吉田裕之: プロダクトベース・プロセス支援の提案と実現, 情報処理学会研究報告, SE-95-84, pp. 33–40 (1995b).
7) 山本里枝子, 上原忠弘, 中山裕子, 吉田裕之: ソフトウェア開発支援システム SOFTPIE の開発, 情報処理学会研究報告, GW-96-20, pp. 31–36 (1996).