エージェントメールを適用したバージョン管理支援システム*

3J－1

服部 真穂　松尾 朗　貫井 春美†
株式会社 東芝　研究開発センター　システム・ソフトウェア生産技術研究所‡

1　はじめに

遠隔に分散した複数拠点でのソフトウェア共同開発において、開発を円滑に進めるためには、開発中の成果物のバージョンを同一にしておくことが必要である。開発途中での仕様変更や、それに伴う修正作業には、変更の履歴登録や関係者への通知といった作業が伴う。これらが同期をとって行なわれない場合、開発成果物のバージョンのずれが拠点間で発生することはよくある。

筆者等は、ソフトウェア共同開発の作業分析を行ない、特に仕様変更に伴う作業に着目し、その流れ(ワークフロー)を抽出した。

本稿では、分析結果から抽出したワークフローと、ワークフローに沿って、成果物の転送とそれに伴う通知を自動化するためのシステムの構想に関して述べる。

2　共同開発の分析

ある制御システムの開発を事例とし、共同開発を分析し、問題点を抽出した。

2.1　共同開発の形態

複数拠点に分散した形態である。図1に示すように、開発管理拠点では、サブシステムの分割や各拠点の進捗管理や総合テストを行ない、仕様書等ドキュメントのマスタを管理する。開発拠点では、分割されたサブシステムの製造や単体テストを行ない、それに関する仕様書等ドキュメントやライブラリ等の複製を持つ。

2.2　現状の問題点

- 仕様変更に関して
  拠点間では多くの成果物を共有している。各拠点の開発者がそれぞれの仕様変更を繰り返している間に共有物のフェーズがずれてしまったり、仕様変更の連絡が行き届かないために、変更を知らずに作業を進めてしまうことによる作業の後戻りといった問題が生じる。

- DR に関して
  拠点が分散しているため、DR や WT の間隔が長くなり、仕様に対するフェーズのずれが大きくなる。

- テストに関して
  いくつかのサブシステムを合わせた組合せテストや実機を用いた総合テストでは、先に述べた仕様のずれが影響し、各成果物のインタフェースがずれている場合が多く、後戻り作業の原因となる。

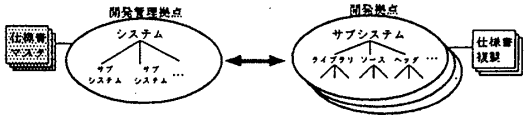

図 1: 共同開発形態

3　仕様変更のワークフロー

前述のように、共同開発の遅延は特に仕様変更時に発生すると考えられる。DR やテストをスムーズに進行させるためにも、開発の途中で頻繁に行なわれる仕様変更時に同期をとることが重要である。

そこで、仕様変更に伴う作業とその流れを整理するために、調査結果から以下の4点を中心に仕様変更のワークフローを抽出した。

1. 仕様変更や修正を行なうトリガとなる要因
   (a) 自発的に仕様変更を行なう場合
   (b) 他からの指示により仕様変更を行なう場合
   (c) 他からの連絡によりドキュメント等の差し替えだけを行なう場合
2. 仕様変更や修正の通知方法
   (a) 仕様変更に対する承認やテストを行なう前にあらかじめ簡単な通知をしておく場合
   (b) 仕様変更に対する承認やテストを行なった後で通知する場合
3. 仕様変更や修正に関連する成果物の有無
   (a) 仕様変更や修正に関連して修正を行なう成果物がある場合
4. 仕様変更や修正後の登録方法
   (a) 仕様変更や修正の承認やテストを行なった後で登録する場合
   (b) 他からの連絡による差し替えだけのため、承認やテストを行なわずに登録する場合

これらから抽出したワークフローは、開発管理拠点において17パターン、開発拠点において7パターンあり、更にこれらはそれぞれ基本的な8パターンの部品の組合せとして表現されることが確認された。基本パターンを表1に示す。

基本パターンの組合せから構成される開発管理拠点でのワークフローの一例を図2に示す。この例では、他からの連絡により開発者がマスタファイルの差し替えを行ない、テストを行なう前にあらかじめ通知する。それと同時に差し替えたソースに関連する別のソースの修正を行ない、それに関する通知はテスト後に行な

*A suppout system for version control using Agent Mail
†Maho Hattori,Akira Matsuo,Harumi Nukui
‡Systems & Software Engineering lab., R & D Center, Toshiba Corp.

表1. 仕様変更ワークフローの基本パターン

開発管理拠点	開発拠点
㋐自発的に仕様変更を行なう(1-a)	㋐自発的に仕様変更を行なう(1-a)
㋑自発的に仕様変更を行ない(1-a)、承認やテストを行なう前に通知する(2-a)	㋑自発的に仕様変更を行ない(1-a)、承認やテストを行なう前に通知する(2-a)
	㋒他からの指示により仕様変更をを行なう(1-b)
	㋓他からの指示により仕様変更を行ない(1-b)、承認やテストを行なう前に通知する(2-a)
㋔他からの連絡により差し替えを行なう(1-c)	㋔他からの連絡により差し替えを行なう(1-c)
㋕他からの連絡により差し替えを行ない(1-c)、承認やテストを行なう前に通知する(2-a)	
㋖仕様変更に関連する成果物の仕様変更を行なう(3-a)	㋖仕様変更に関連する成果物の仕様変更を行なう(3-a)
㋗仕様変更に関連する成果物の仕様変更を行ない(3-a)、承認やテストを行なう前に通知する(2-a)	㋗仕様変更に関連する成果物の仕様変更を行ない(3-a)、承認やテストを行なう前に通知する(2-a)
㋘仕様変更の承認やテストを行なってから登録し(4-a)、通知する(2-b)	㋘仕様変更の承認やテストを行なってから登録し(4-a)、通知する(2-b)
㋙仕様変更の承認やテストを行なわずに登録し(4-b)、通知する(2-b)	

うといった流れである。連絡を受けた他の開発者の後には別のワークフローが続いていく。図中の点線で示す部分が基本パターンにあたる。

以上から基本パターンの組合せからワークフローのカスタマイズが可能となり、開発対象システムや規模による開発方法の違いに対処できる可能性、コンピュータを使った定型的な支援の可能性が確認された。

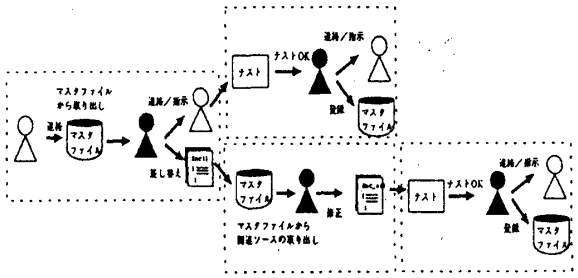

図 2: 仕様変更におけるワークフロー例

4 コンピュータ支援の構想

コンピュータ支援の機能要件を以下に挙げる。

1. 仕様変更と同期をとり通知させる。
2. 仕様変更の範囲により通知の範囲も変える。
3. 成果物を一元管理し、バージョンを統一する。
4. 変更の履歴管理をする。

成果物の管理やバージョン統一のためには、バージョン管理ツールが存在している。これらにメッセージ交換システムをリンクさせれば、通知の自動化が図れ、仕様変更と同期をとった通知が可能になる。その際、メッセージ交換システムにワークフローの流れに沿った動作をさせることが必要となる。

このような理由から、エージェントメールシステムと既存のバージョン管理ツールを組合せた支援システムを開発中である。エージェントメールシステムは、非同期型コミュニケーション支援を行なうシステムで、ユーザのエージェント記述スクリプトに従って、電子メールで交換するメッセージに対する諸操作を自動的にバックグラウンドで行なうものである。

図3に支援システムの構成を示す。エージェントメールシステムが読み込むスクリプトにワークフローに相当する処理を記述する。

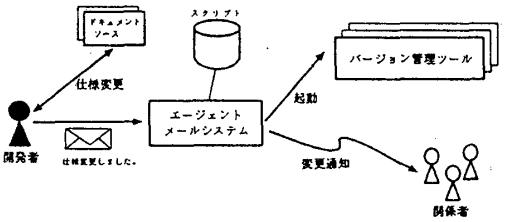

図 3: エージェントメールによるバージョン管理支援システム

5 期待される効果

バージョン管理支援システムにより、仕様変更に伴う複雑な作業の流れを一つのルールとして管理することができる。

- 変更の経路や変更記録が開発者が意識することなく自動的に履歴として登録されるため、データの一元管理が可能になる。それにより必要に応じて変更に関する情報が手に入いる。
- 変更が行なわれると自動的に変更通知が発信されるため、通知漏れや通知不足などを防ぐ。
- エージェントスクリプトを記述することで、誰から誰にどういった手順で作業が流れるかが明確になる。
- 開発段階によってワークフローが異なる場合でも、エージェントスクリプトを書き換えることによって、異なる流れのフローを容易に作成し直すことができるため、幅広く対応できる。

6 おわりに

共同開発上の仕様変更作業をワークフローとして捉え、仕様変更に伴う作業の流れを明確にした。そして、そのワークフローをエージェントスクリプトに記述することにより、開発者個人作業と拠点間でのコミュニケーションを要する作業とを一つの流れとしてルール化し、仕様変更時の同期を確実にとることが可能になった。

今後、更に問題点を抽出しながら柔軟性を持つ支援システムを開発していく予定である。

参考文献

[1] 松尾, 貫井, 中村:電子メールにおけるエージェントシステム, 第 43 回全国大会講演論文集 (5)pp.296-297(1991)