分散協調作業のためのバージョンフロー・モデルについて

4AA－4

岩井原 瑞穂, 井上 創造, 松尾 博基
九州大学 大学院システム情報科学研究科 情報工学専攻

1 はじめに

近年の計算機ネットワークの普及に伴い，遠隔地を高速ネットワークで結び，様々な協調作業を計算機で支援することが可能になりつつある．ビジネス分野におけるワークフローを用いた定型的処理の効率化などは研究が盛んな分野の一つである．

CAD/CASE などの設計分野においてもプロセス管理に計算機支援を取り入れ，プロセス管理を組み込んだCASE ツールにおける並行処理制御のための様々なモデルが提案されている [2]．

一方，これまでに様々なの拡張トランザクションモデルが研究されており，長時間トランザクションやバージョン管理などの問題が検討されてきたが，具体的にこれらのモデルを適用するには，個別の協調作業環境の情報を多く必要とする．設計プロセス管理のような制御構造が与えられるようになっている現在，これらの技術を，トランザクションモデルで議論されてきた一貫性管理とどのように統合するかが重要な課題となると思われる．

本稿では，分散協調作業におけるバージョンフロー・モデルについて述べ，プロセス管理とオブジェクト管理の統合の基礎となるメッセージの定式化について述べる．

2 バージョンフロー・モデル

2.1 バージョンフロー管理

プロセスとは意味的なまとまりのある作業の集合である．プロセス間で受渡される種々の設計オブジェクトについてそのデータの授受関係や生成/消滅過程を依存関係として管理する必要がある．このような設計プロセス間でのオブジェクトの依存関係をバージョンフローと呼ぶことにする [3]．

ビジネスプロセスと異なり設計プロセスでは多くのバージョンを扱う必要があるが，現在までのワークフローモデルにおいてバージョンの問題は，詳細に議論されていない．

2.2 プロセス内/プロセス間の一貫性管理

設計プロセスのなかにはシミュレーションやレイアウトのように数日から数週間といった長期に渡るものもある．これらのプロセスの結果は，論理設計や改良のためのレイアウト設計などの上流プロセスで利用され，デザインループは期待した性能が得られるまで繰り返される．

実際の状況では急な仕様変更やバグレポートなどの予期しえない事象によりしばしばプロセスは中断される．稼働中のプロセスのアボートには多大なコストがかかり，他のプロセスへの一貫性が保てない場合でも古い入力のまま作業を続けることもある．

またプロセスごとに地理的に分散している環境においては，一貫性を保証すべきプロセスの集合が動的に変わることも有り得る．このような環境においては，バージョンフローを中心として一貫性を取るべき範囲を制御することが有効である．

"VersionFlow Model for Distributed Collaborative Work,"
M. Iwaihara, S. Inoue, H. Matsuo, Kyushu University

2.3 コミュニケーションとオブジェクト管理の統合

設計オブジェクトの受け渡しに伴ない，メイルや電話，ファクスなどを用いて多量の通信が行なわれる．その内容は，ツールの使い方などの初歩的な質問から，データ受け渡し方法の確認，設計上の問題点の指摘，仕様変更の解説，設計制約違反の報告など設計オブジェクトに関するものや，さらに署名の伴なう設計移行承認書など，設計責任の所在を明確にするための書類も存在する．このように設計者間のコミュニケーションを図るためのメッセージは，設計の目的であるオブジェクト (バージョン) の受け渡しよりも頻度の面からは，はるかに多い．これらのメッセージを参照するオブジェクトと関連づけ，データベース・オブジェクトとして管理することにより，見通しのよいバージョン管理およびトランザクション管理を行なうことができる．

2.4 バージョンフロー・モデルの構成要素

以上の背景を踏まえ，我々は協調設計環境のためのバージョンフロー・モデルを提案している [3]．バージョンフロー・モデルは以下の構成要素からなる．

- プロセス，メッセージ，ハンドル
- メッセージリンク
- プロセス制御ルール，イベント伝播ルール

メッセージは利用者間のコミュニケーションに利用される，プロセスの状態，オブジェクトのバージョンはメッセージと共に記録され，バージョンフローを表現する．ハンドルはメッセージから設計オブジェクトへのリンクでインターフェイスとして利用される．

プロセス制御ルールはプロセスの状態遷移を制御し．プロセスの実行によりメッセージのネットワークを生成する．新しいメッセージを生成する場合，それは状態の更新を意味する．イベント伝播ルールはイベント伝播関数により更新の影響を解析する．イベント伝播関数とは衝突の重要度を示すイベントランクを返す関数であり，メッセージリンク型ごとに定義される．

我々のアプローチではメッセージを利用することにより，一貫性の保たれているメッセージリンクの範囲を制御する手法をとる．

以下，本稿ではバージョンフロー・モデルの構成要素のうち，メッセージ，プロセス，ハンドルの定式化を述べる．

3 プロセスとメッセージのモデル化

3.1 概要

意味的なまとまりのある作業をプロセスと呼ぶ．すべての作業はメッセージによって記録され，メッセージはそのメッセージ列に加えられる．

プロセスクラスとは同一機能を持つプロセスの集合であり，唯一のプロセスクラス名を持つ．プロセスクラス名は例えば，「提案」(proposal)，「評価結果」(evaluation result)，「設計変更」(designchange)，「バグ追跡」(bug chase)，「改善要求」(improverequest)，といったものからなる．

メッセージは本体 (テキスト/音声/画像) や日付をといった電子メールと同様の情報を持ち，参加者によって生成され

る．各メッセージはハンドル集合を持つ，ハンドル集合とは設計データ，ドキュメントへのリンク集合である．

それぞれのメッセージは状態を持ち，それらはプロセスに属する．各プロセスクラスにはそのプロセスで起こり得るメッセージ状態集合が割り当てられている．メッセージ状態により，参加者およびプロセスの状況を把握できるのが容易になるとともに，プロセス管理およびオブジェクト管理に有用な情報を得ることができる．

メッセージの生成はメッセージ管理ルールで制御される．参加者がメッセージや既存のメッセージに関連づけるリンクを生成する要求を出すと，メッセージ管理ルールは生成できるメッセージ状態を与える．

メッセージ状態には，議論の経過を示すために「提案」，「意見」，「賛成」，「結論」等の構造化議論モデル (gIBIS)[1] と同様の状態，および作業の状況を表わすために「開始」，「終了」，「分担」，「移行」などの状態を用意する．

3.2 プロセス

プロセスは以下の属性からなるオブジェクトである．

$$[pID : p, Pclass : P, Members : M]$$

ここで pID はプロセスのオブジェクト識別子である．各プロセスはそれぞれ一つのプロセスクラス (process class) に属する．プロセスクラスの全体集合を $\mathcal{P}$ で表す．

$Menbers(p)$ は参加者集合の ID でありプロセス p に参加している人を表わす．

3.3 メッセージ

メッセージは以下の属性からなるオブジェクトである．

$$[mID : m, Sender : d, Process : p, State : s, \\ Handles : H, Body : b]$$

ここでメッセージのオブジェクト識別子を特にメッセージ識別子 (mID) と呼ぶ．$Sender(m)$ は m を作成した参加者の名前であり，$Menber(p)$ の要素である．$Body(m)$ はメッセージ本体である．

$Handle(m)$ はメッセージ m が使用するハンドルの集合である．

3.4 メッセージ状態

各メッセージ m はメッセージ状態集合 $\sum$ の要素をメッセージ状態 (m 状態あるいは単に状態と略) $State(m)$ として持つ．また各プロセスクラス P について，それに属するメッセージの状態集合 $\sum^P$ に $\sum$ が定められており，P の各メッセージ m について $\sum^P$ の状態が成り立つ．

3.5 メッセージ・リンク

メッセージを始点または終点とするリンクをメッセージリンク (m リンクと略) と呼ぶ．

メッセージ列 $\mathcal{M}$ は節点のラベルが mID で枝のラベルが m リンク名からなる有向グラフで記述される．これをメッセージネットワークと呼ぶ．

リンクごとに取り得るメッセージ状態を定め m リンク型として導入する．$\mathcal{L} \times \sum \times \sum$ 上の関係 $mlTypes$ を (m リンク型関係) と呼び，これにより組 $[ln, s1, s2]$ が $mlTypes$ に含まれるならば，m 状態 s_1 を始点とし，m 状態 s_2 を終点とする m リンク名 ln の m リンクが存在できることを表わす．

3.6 ハンドル

ハンドルをメッセージとオブジェクトのインターフェイスとして定義する．利用者はオブジェクトを関連づけるためにメッセージ中にハンドルを置く．各ハンドルは一つのプロセスに属しそのプロセス中のメッセージもそれを所有する．ハンドルは次の目的で用いられる．

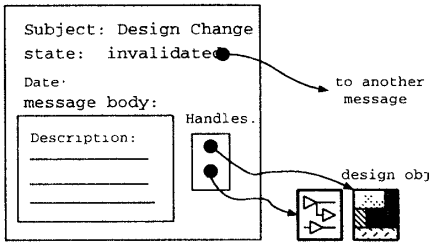

図 1: メッセージ形式

- メッセージが生成されたときに参照していたオブジェクト集合のバージョンを記録する．
- プロセス内からのデータベースへのビュー．
- 合意事項や To-Do リストのような拘束力のあるメッセージを，ハンドルから参照するオブジェクトとし，更新される場合の影響を監視する．
- 構成 (configuration) として，一貫性を保持する単位としてオブジェクト集合をひとつにまとめる．
- プロセス間でハンドルを伝達することによるバージョンフローの表現．

ハンドル型 とは以下のオブジェクトクラス名 $O_i(i = 1; ...; k)$ を要素とする組である．

$$H = [O_1, \ldots, O_k]$$

ハンドルは以下の形式のオブジェクトである．

$$[OID : h, Message : m, Object : [O_1 : o_1, \ldots, O_k : o_k]]$$

ここで h はオブジェクト識別子，m はメッセージ ID で必須である．属性の $Object$ は O_i に対応するオブジェクト識別子 o_i の組である．

4 おわりに

本稿では，バージョンフローの概念を導入した分散協調設計環境モデルについて述べ，その中で用いるメッセージとプロセス，ハンドルの定式化を行なった．

我々の分散設計環境モデルはメッセージネットワークの管理に基づく．メッセージネットワークは現在のプロセスの状態と過去の履歴を表現し，利用者のより良い理解と制御のしやすさを実現する．さらにオブジェクトのバージョンがハンドルによりメッセージから関連づけられ，利用者間で異なったバージョンを見ることにより生じる誤解を避けることができ，またオブジェクト管理に必要な情報を得ることができる．

参考文献

[1]J. Conklin and M. L. Begeman, "gIBIS: A Hypertext Tool for Exploratory Policy Discussion," *ACM Trans. Office Information Systems*, Vol.6, No.4, pp.303-331, Oct 1988.

[2]C. Montangero (ED.), "Software Process Technology, Proc. 5th European Workshop, EWSPT'96," *Lecture Note in Computer Science*, Springer, 1996.

[3]岩井原 瑞穂, "分散協調設計環境におけるバージョンフロー制御に必要な機能," 重点領域研究「高度データベース」, 第 2 回研究集会論文集, Vol.2, pp.441-448, 1996 年 9 月.