4AA－5

分散協調設計におけるメッセージ管理とトランザクション管理の統合

井上創造，松尾博基，岩井原瑞穂
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

これまでCAD／CASEなどの設計分野においては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ツールにおいて並行処理制御のための様々なモデルが提案されてきた．このような環境では，拡張トランザクションモデル[1]が研究されており，長時間トランザクションなどの問題が検討されている．

また，近年の計算機ネットワークの普及にともない，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．

このような分散環境における協調設計作業においては，設計オブジェクトの参照などの機能を付加した電子メイルなどのメッセージを用いて，プロセスの進行状態を管理することが重要となってくる．

これらのメッセージ管理技術を，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主要な課題となるであろう．これまでのトランザクションモデルにおいては，プロセスの意味論を考慮したオブジェクトの並行処理制御，一度コミットされたトランザクションの取り消し／変更，中止されたプロセスによる結果の再利用などに関して問題がある．

本稿では，我々の提案する協調設計作業環境のためのバージョンフローモデル[5]において一貫性を管理するトランザクション管理モデルの検討を行なう．

2　バージョンフローモデル

意味的なまとまりのある作業をプロセスと呼ぶ．

バージョンフローモデルにおいては，すべての作業はメッセージによって記録される．

メッセージは本体や日付といった電子メールと同様の情報のほかに，状態，他のメッセージへのリンク，ハンドル集合といった情報を持ち，参加者によって生成される．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．

メッセージ間のリンクはリンク名を持つ．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．

ハンドル間にもリンクが存在し，リンク名を持つ．

3　プロセス制御ルール

メッセージの生成が可能かどうかを管理するためにプロセス制御ルール$\mathcal{R}_p$を定義する．$\mathcal{R}_p$はメッセージ列$\mathcal{M}$，メッセージ状態集合$\sum$，リンク名の全体集合$\mathcal{L}$を定義域とする制約集合である．

$\mathcal{R}_p$は新しいメッセージmの状態が生成済みのメッセージ列$\mathcal{M}$に対して生成可能であるかどうかを決定し，メッセージ状態の正しい遷移を定義する．

"Integrating Transaction Management and Message Management in Distributed Collaborative Design Process," S. Inoue, H. Matsuo, and M. Iwaihara, Kyushu University

4　既存のトランザクション管理モデル

これまでCAD／CASE分野ではユーザはトランザクションにたよらずに作業対象をロックするなど臨機応変的な対応が主だった．これではデータの一貫性の保証がなされないため，トランザクションモデルで議論されてきた一貫性管理をどう適用するかが課題となる．このときの問題点として以下のようなことがあげられる．

1.一度コミットされたプロセスはその入力が任意の時点で取り消し／変更されることにうまく対処できない．一般にコミット後のデータを取り消したい場合は，あらたに別のトランザクションで補償を行なう手法が用いられる必要がある．しかしあるオブジェクトの変更により波及する影響を受けるプロセスを特定することは困難である．

2.また参照しているオブジェクトがアボートされたプロセス自身をすべてアボートすることは，無駄が大きい．設計作業においては，例えば作業が長時間にわたることも多いため，アボートされるべき作業結果も何らかのかたちで再利用することが試みられるのが現実的である．

このようなことから，あるオブジェクトの変更により波及する影響を受けるプロセスを特定すること，全てのバージョンが同期していない状態で一貫性の取れているプロセスの範囲をシステムで把握することが重要となってくる．

5　メッセージ管理におけるトランザクション管理

以下では，バージョンフローモデルにおいて設計作業をメッセージリンクであらわした例をあげ，トランザクション管理の方法を検討する．図1はソフトウェアプロセスのベンチマーク例題[2]を参考にしたメッセージリンクの例である．ここでは，m_1においてプランが決定した後，m_3でコード生成，m_6でデザイン見直しが開始される．ここでは，プロセス制御ルールによって状態submit_planからreceivedへの遷移が許されているものとする．

次にm_7においてデザイン修正要求が出される．これを受けてm_8においてオブジェクトplanの修正プロセスp4が開始される．ここでオブジェクトplanが更新されるため，プロセスp2の一貫性が保たれなくなる．このことはplanのハンドルリンクを上にたどっていくとm_1，m_2，m_3間のハンドルリンクより，オブジェクトplanをプロセス間で共有していることから容易に調べることができる．ここでp2は以下のいずれかの選択を行なう．

1.(バージョン分離) 古いバージョンのまま作業を続ける．

2.(プロセスの強いアボート) プロセスp2をアボートする．

3.(プロセスの弱いアボート). 古いプロセスを停止し，その途中までの結果を参照しながら新しい入力でのプロセスを再発行する．

これらの選択は人間の指示によるものであるが，メッセージ・リンクの定める意味的制約のもとで振舞うものとする．

1の場合は，何ら特別な作業は必要ない．ただし，一貫性のとれていないプロセスが出現することになる．

2は，$ln\langle m_1, m_2\rangle$ およびそこから有向枝を順方向にたどっていくことのできるメッセージを消去することによって実現される．

3は，$ln\langle m_1, m_2\rangle$ から有向枝を順方向にたどっていくことのできる実行中のメッセージ，つまり実行中を表す状態でそこから枝が出ていないようなメッセージに対して，新たに中止を表す状態名 stopped のメッセージを生成することによって実現する．

図1では，3を選択した例を示す．この選択により，p2 は中止を表す状態 stopped を持つメッセージ m_4 を生成して止まる．p4 が終了して新しいプラン plan' が決定すると，m_{10} で新たなコード生成プロセス p5 が開始される．ただし，ここでは途中まで作成しているコード code も参照していることに注意されたい．

以下ではバージョンフローモデルにおけるトランザクション管理を考察する．バージョンフローモデルにおいては，以下のようなことが可能となる．

1. プロセスの状態を考慮した並行処理制御を行なうことができる．上の例のようにプロセスの状態により他の並行度を変化させることができる．

2. 一度コミットされたプロセスもその入力が任意の時点で取り消し／変更されることによる影響を解析できる．リンクをたどることによって意味的に影響の及ぶ範囲を知ることができる．

3. 入力が取り消されたプロセスの作業結果も再利用することができる．一貫性を保たないプロセスをアボートせず結果を保存しておくことによって実現できる．

また，ハンドルリンクをたどっていくことによって最新のバージョンのオブジェクトがわかり，オブジェクトの一貫性のとれている範囲を検査することができる．

上記を実現するためには，メッセージの生成に際して以下のような性質を満たす必要がある．

1. 各プロセスの状態遷移に関する制約は，メッセージリンクから全て導出できる．

2. オブジェクトの更新／参照の依存関係は，ハンドルリンクから全て導出できる．

3. メッセージおよびリンクが取り消されると，現実の協調作業関係においても影響を残さないと仮定できる．

このような条件のもとで，柔軟なトランザクション管理を行ないながら一貫性を保つことが可能となる．

6 おわりに

本稿ではバージョンフローモデルにおいて一貫性を保ちつつより柔軟なトランザクション管理を行なう方法について考察した．

今後は詳細な定式化，問題の検討を行ないながらバージョンフローモデルとトランザクション管理の統合をめざす予定である．

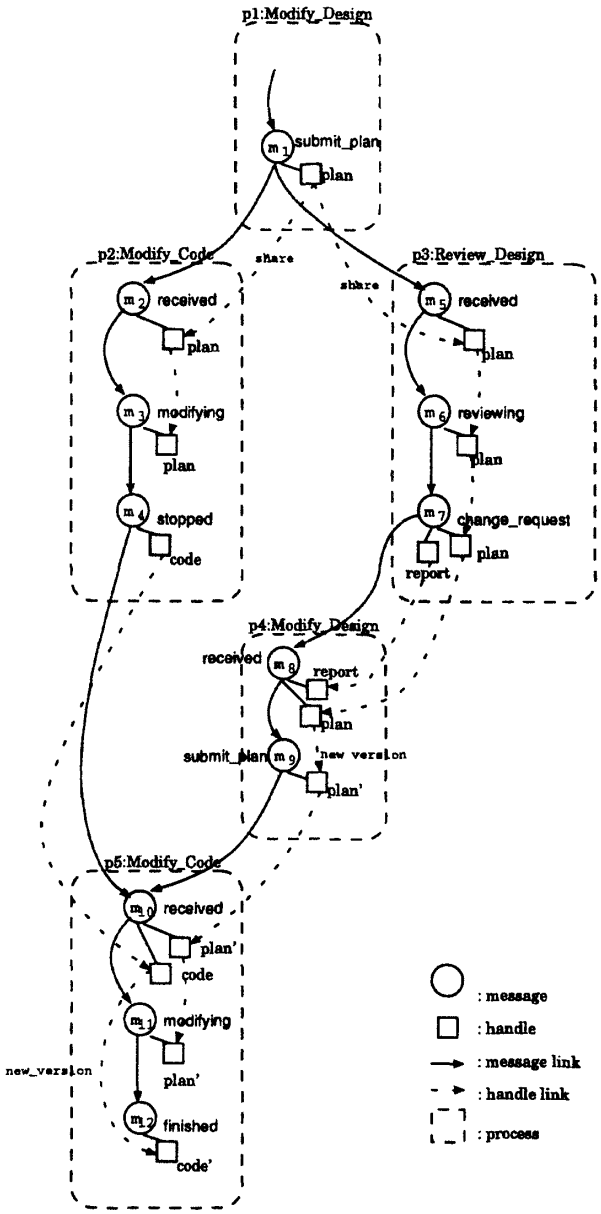

図 1: ソフトウェアプロセスにおけるメッセージリンクの例

参考文献

[1]A. K. Elmagarmid (Ed.), "*Database Transaction Models for Adcanced Applications*," Morgan Kaufmann, 1992.

[2]M. I. Kellner, et al., "Software Process Modeling Example Problem," *Proc. 6th Int. Software Process Workshop, pp.19-29, 1990.*

[3]K. Ramamritham and P. K. Chrysanthis, "*Advances in Concurrency Control and Transaction Processing*," IEEE Computer Society Executive Briefing, 1997.

[4]"特集「グループウェアの実現に向けて」," 情報処理, Vol. 34, No. 8, 1993.

[5]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」第二回研究集会論文集, Vol.2, pp. 441–448, 1996 年 9 月.