

非同期型コミュニケーションにおける トランザクションの動的構築

井上 創造[†] 岩井原 瑞穂[†]

電子メールに代表される蓄積型のメッセージは、発言者と受信者が非同期に通信するコミュニケーションの手段として幅広く用いられている。メッセージは、複数生成されやり取りされることで、まとまりを持った作業を動的に構成していく。本論文では、このようなやりとりに対しトランザクション管理を行なう手法を提案する。本手法により、利用者やシステムがメッセージに表される作業の状態を把握することが容易になり、データベースのトランザクションとの連動も可能になる。メッセージ上のトランザクションは人間の非定型的な協調活動を表すため、動的に構築される複合的な構造を持ち、ワークフロー管理システムに代表される既存のプロセス管理システムでは対応が困難である。提案する手法は、複合的なトランザクションを実行時に動的に構築できる特徴を持つ。またプロトタイプシステムについて述べ、本モデルの実用性を評価した結果を示す。

Dynamic Composition of Transactions on Asynchronous Communication

SOZO INOUE[†] and MIZUHO IWAIHARA[†]

Asynchronous messages, such as E-mails, are widely used in asynchronous human communication. In real situations, exchanges of messages can be regarded as a process of a particular purpose. We propose a model for managing such a process as a transaction. The model facilitates for users, software agents, or the system to grasp the state of a process represented by exchanges of messages, and database transactions can be initiated by message transactions. Transactions on message communications form complex structure, and constructed dynamically in the session. Therefore, it is difficult to support these ad-hoc processes by conventional process management systems, such as workflow management systems. Our method supports dynamic composition of structured transactions. We also describe a prototype system implementing the model, and show results of analyses to ascertain practicality of our model.

1. はじめに

データベース分野においては、並行処理やシステム障害に対しデータの一貫性を保証するために、トランザクション、つまり、ACID 特性(原子性(Atomicity)、一貫性(Consistency)、孤立性(Isolation)、耐久性(Durability))を満たす処理単位の概念が提案された⁵⁾。また、人間が介在する環境ではトランザクションの長時間化が問題となるため、これまでに様々な拡張トランザクションモデルが提案されてきた¹⁶⁾。

一方、電子メールや電子掲示板に代表される蓄積型メッセージは、計算機支援による協調作業における非同期かつ蓄積型のコミュニケーションの手段として、幅

広く用いられている。

現実の協調作業におけるメッセージやメッセージのやり取りは、作業のプロセスの進行やデータベースの更新といった、作業を構成する要素に関する情報を含む。例えば、会議室の予約情報を管理するデータベースを用いて会議室の予約をする場合、参加者の日程調整を完了し確認を促すメッセージは、実際には、参加者への告知をするばかりでなく、参加者の日程調整プロセスの終了やデータベースの予約トランザクションの開始を意味する。

通常のメッセージ管理システムでは、このようなコミュニケーションに関連する作業の状況が明確化されないため、次のような問題がある。

- データベース管理システムやワークフロー管理システムといった他システムのデータとの不整合が生じる。

[†]九州大学 大学院システム情報科学研究科 情報工学専攻
Graduate School of Information Science and Electrical
Engineering, Kyushu University

- メッセージのやり取りに伴い変化する作業の情報に対し、利用者間で認識が統一されない。

これらの問題に対し、我々は以下のアプローチが重要であると考えます。

- (1) メッセージのやり取りをメッセージトランザクションとしてモデル化することにより、他のシステムが協調作業の状況を把握するための情報を明確化する。

メッセージトランザクションを次のように定義する。メッセージトランザクションは、協調作業におけるコミュニケーションの進行に対し、協調作業で定められた一貫性を保証する利用者間のメッセージのやり取りである。上述の例では、参加者への確認、参加者の日程の調整、部屋の予約というメッセージトランザクションが存在する。メッセージにメッセージトランザクション管理機能を導入することにより、以下の利点が得られる。

- 各メッセージの表す、議論や作業といった時間的な長さを持つ要素に対し、現在実行中なのか、成功して終了したか、失敗して終了したかの状態が明確化され、利用者の混乱を避けることができる。また、その状態の遷移が他の要素の状態に及ぼす影響を管理できる。
- ワークフローやデータベースにおいて管理されるトランザクションの情報がコミュニケーションに現れた場合、その状況を把握するための情報をワークフロー管理システムやデータベース管理システムに与えることができる。例えば、ある合意が得られるとデータを公開するトランザクションが発生する等である。

ただし、コミュニケーションは人間の非定型的な協調活動に基づくため、事前には予測困難な性質を持ち、ワークフロー管理システムに代表される既存のプロセス管理システムで効率よく支援することは難しい。よって、我々はさらに以下の課題をあげる。

- (2) 利用者が作業の実行中にメッセージトランザクションを動的に構築していくことにより、各メッセージの表す議論や作業プロセスの状況を柔軟に表現する手法を開発する。

我々は、メッセージ管理システムにおいて、メッセージトランザクション間の従属性表現に基づき、入れ子トランザクション¹⁵⁾や代替トランザクション¹⁶⁾のような構造化されたトランザクションを記述する手法⁷⁾および、メッセージトランザクションを実行時に動的に構築する手法を提案した⁶⁾。また、この手法の有用性を実証するため、提案するモデルに基づくプロトタイプ

システムを開発した。さらに、本モデルが実際の協調作業を十分に表現できるかを電子メール発信記録にモデルを当てはめることと実際にプロトタイプシステムを使用することにより評価した。

本論文では、2節でコミュニケーションにおけるメッセージトランザクションをモデル化し、4節でプロトタイプシステムの開発について述べる。また、5節で本モデルの実用性を評価した結果を示す。

2. メッセージトランザクション

この節では、メッセージトランザクションをモデル化する。メッセージトランザクションは、単一あるいは複数のメッセージから構成されると考え、意味的なメッセージのまとまりを概念化した m-group を 2.1 節で導入する。また一貫性については、やり取りの順序や、やり取りの進行が他のやり取りに与える影響を保証することが重要と考え、2.3 節で m-group 間の順序や発生を表すトランザクション従属性を導入する。メッセージトランザクションは、通常メッセージ管理システムにおけるメッセージや、データベースのトランザクションと比較すると次の特徴を持つ。

- 複数のメッセージから構成されることで、時間的な長さを持ち、作業の進行に従い実行状態を変化させる。通常メッセージ管理システムにおけるメッセージは、発信時刻の情報を持つことはできるが、作業プロセスなど、生成して終了するまでに時間があるような情報を形式的に記述できない。メッセージトランザクションはデータベースにおけるトランザクションと同様に、生成、実行中、commit、abort という実行状態を持ち、時間的な長さを持つことができる。commit と abort は、メッセージトランザクションの定められた意図が有効となって終了したかどうかを表すものである。データベースにおけるトランザクションの commit と abort は、トランザクションによるデータの更新やデータの参照の処理がすべて成功したか、すべて取り消されたかを表すが、協調作業では作業の取消しは無駄が多く現実的ではない。しかし、m-group はしばしば「提案」や「日程調整」といった意図を持つため、「提案が採択された」、「日程調整に成功した」など、m-group クラスの意図が達成されたかどうかを明確にすることは重要であると考えられる。m-group の意図は m-group クラスにより表される。
- メッセージトランザクション間の実行状態の従属性を持つ。参加者間で会議日程を調整する際には、

参加者が主体的に発言し、自由にメッセージがやり取りされるが、会議日程の調整の終了は参加者の合意の後でなければならないというように、やり取りの手順に関する制約が存在する。このような制約を、分散システムに適した制約の記法として文献¹²⁾や文献¹⁾で用いられている時相論理に基づく従属性を用いてトランザクション従属性として記述する。トランザクション従属性を選択的に組み合わせることにより、ACID 特性を必要に応じて導入し、入れ子トランザクションや代替トランザクションなどの拡張トランザクションモデルを記述できる⁷⁾。文献⁷⁾でトランザクション従属性の充足可能性について議論しており、現実の制約を保つために必要な条件として充足可能性を判定する機構を用いる。

- メッセージトランザクションの構造を動的に構築することができる。人間の行動は元々予測困難な要素を持つ。例えば、議論のメッセージトランザクションのある時点では、次にどのような発言が行われるのか予測できない。これまでの提案に対し新たな代案が提案されるかもしれないしされないかもしれない。代案が提案されれば、議論のメッセージトランザクション部分的な構造として、代案に対する議論が開始され、これまでの案か代案のどちらかしか commit できないという制約が与えられる。このような構造は議論の進行に応じて動的に与えられるものであるため、作業の進行に応じて作業の構造を動的に決定する必要がある。あらかじめ代用可能なトランザクションが定義される代替トランザクション¹⁶⁾のように、既存のトランザクションモデルはあらかじめ定義された範囲での構造のみを許すため、このような予測できないトランザクションの動作を適切に管理できない。提案する手法では、メッセージトランザクションの構造を記述する mg テンプレートを導入し、mg テンプレートを任意の時点で組み合わせてメッセージトランザクションを構築する。
- 非巡回有向グラフ (DAG) を許す構造を持つ。人間が介在する応用分野ではトランザクションの長時間化が問題となるため、入れ子トランザクション¹⁵⁾などの階層化された構造を持つトランザクションが提案されたが、その構造はトランザクションはただかだか 1 つのトランザクションの子にしかならないという、木構造を基本としたものである。しかし、メッセージのやり取りにおいては、異なる議論プロセスの間での調整や作業グループ間の交

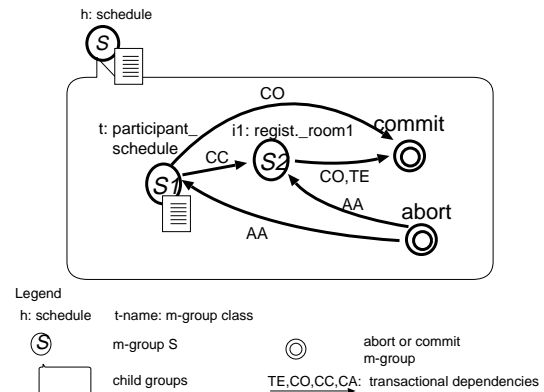


図 1 m-group

渉など、複数のメッセージトランザクションの部分メッセージトランザクションとなるような構造が存在するため、メッセージトランザクションでは複数の親を許す DAG 構造を導入する。

2.1 m-group

m-group S は、タプル

$$[S, Class(S), TName(S), Body(S), CG(S), Dep(S)]$$

で記述される。

$Class(S)$ は、m-group クラスの集合である。m-group クラスは、発言者の意図を提示するために用いられる。m-group クラスの例として、「提案」「意見」「賛成」「結論」といった構造化議論モデル (IBIS)³⁾で用いられているものや、ソフトウェアプロセス¹¹⁾における「バグ報告」「改善要求」「設計変更」「評価結果」といったものがあげられる。提案するモデルではさらに、m-group の構造を記述する mg テンプレート導出するために m-group クラスを用いる (3.2 節)。

メッセージ本体 $Body(S)$ は、電子メールなどの通常のメッセージである。子 m-group 集合 $CG(S)$ は、m-group 識別子 S_i の集合 $\{S_0, S_1, \dots, S_k\}$ である。それぞれの S_i を S の子グループと呼び、 S を S_i の親グループと呼ぶ。 CG による m-group の階層構造は、複数の m-group の CG となるような DAG 構造を許す。

$TName(S)$, $Dep(S)$ はそれぞれ、 t 識別子 (3.2 節) の集合、 S の子グループ間の従属性を記述するトランザクション従属性 (2.3 節) の集合である。

図 1 は、会議の日程調整の m-group S の例である。 S は、 t 識別子 h 、m-group クラス $schedule$ 、メッセージ本体を持ち、子グループとして、参加者の日程を調整する m-group S_1 、部屋の予約を表す m-group S_2 、

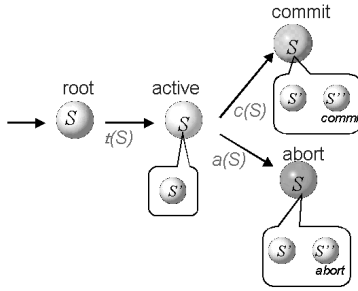


図2 トランザクションイベント

commit, *abort* を持つ。 S_2 はメッセージ本体を持たないが、予定された作業の目的と指針を利用者に示すために、あらかじめメッセージ本体を持たない、空の *m-group* を生成することも可能とする。

2.2 メッセージ上のトランザクション

各 *m-group* S は、メッセージトランザクションの実行状態の遷移に応じて、以下のトランザクションイベントを発生させる (図2)。

- $r(S)$: (*root*) *m-group* S が生成された。
- $t(S)$: (*active*) *m-group* S の子グループ S' が $r(S')$ または $t(S')$, $c(S')$, $a(S')$ のトランザクションイベントを発生している。
- $a(S)$: (*abort*) *m-group* S の子グループに *m-group* クラス *abort* の *m-group* が生成された。
- $c(S)$: (*commit*) *m-group* S の子グループに *m-group* クラス *commit* の *m-group* が生成された。

メッセージトランザクションは次のように実行される。(1)利用者やシステムにより *m-group* S が作られる。(2) S で表されるメッセージトランザクションの進行にともない、新しい *m-group* が S の子グループに加えられる。(3) *m-group* クラス *commit* または *commit* を持つ *m-group* の生成により、実行を終了する。

2.3 トランザクション従属性

メッセージトランザクション間の従属性の表現には、文献¹²⁾, 文献¹⁾ で用いられている記号 $<$, \rightarrow を用いる。

- $e_1 < e_2$: (先行制約) イベント e_1 とイベント e_2 の両方が起きるなら、 e_1 が先に起こる。ただし、 e_1 と e_2 が発生するかどうかに関しては制限しない。
- $e_1 \rightarrow e_2$: (発生制約) イベント e_1 が起きるなら、イベント e_2 も起きる。これは e_1 と e_2 の起きる順序に関しては制限しない。

トランザクションイベント間に上の制約および、論理記号 \wedge (論理積), \vee (論理和), \Rightarrow (含意) を用いて

m-group 間のトランザクション従属性を表す⁶⁾。以下では、トランザクション従属性の代表的なものを示す。 S_i と S_j は、 S または S の子グループであるとする。

- $AA(S_i, S_j)$: (*abort dependency*) $a(S_i) \rightarrow a(S_j)$
 S_i が *abort* なら、 S_j も *abort*。
- $CC(S_i, S_j)$: (*commit dependency*) $c(S_i) < c(S_j)$
 S_i が *commit* かつ S_j が *commit* なら、 S_i が先に起こる。
- $CO(S_i, S_j)$: (*commit occurrence*) $c(S_i) \rightarrow c(S_j)$
 S_i が *commit* なら、 S_j も *commit*。
- $AC(S_i, S_j)$: ($c(S_j) \rightarrow a(S_i) \wedge (a(S_i) < c(S_j))$)
 S_i が *abort* したときのみ、かつ S_i の *abort* 後でのみ S_j が *commit* できる。
- $CA(S_i, S_j)$: ($c(S_i) \rightarrow a(S_j)$)
 S_i が *commit* なら、 S_j が *abort*。
- $TE(S_i, S_j)$: (*terminate*) $(c(S_i) < c(S_j)) \wedge (a(S_i) < c(S_j))$
 S_j が *commit* する前に S_j が終了しなければならない。

これらを組み合わせて用いることで、入れ子トランザクションや代替トランザクションといった種々のトランザクションの記述が可能である⁷⁾。図1の例では、参加者の日程調整が成功しないと部屋の予約ができない ($CC(S_1, S_2)$)、参加者の日程調整に成功し部屋の予約が成功すれば日程調整は成功 ($CO(S_1, S)$, $CO(S_2, S)$, $TE(S_2, S)$)、日程調整が中止されれば子グループは無効になる ($AA(S, S_1)$, $AA(S, S_2)$) ことを示している。 S と S の子グループの間で入れ子トランザクションが実現している。

メッセージグループ集合に対しトランザクション従属性を不用意に与えていくと、トランザクション従属性が充足不能になる可能性がある。この問題については、文献⁷⁾ で議論されている充足可能な状態を保つための十分条件を用いることができる。

3. メッセージトランザクションの動的な構築

3.1 動的に構築されるメッセージトランザクションの分類

この節では、メッセージトランザクションを実行時に動的に構築する手法を述べる。メッセージトランザクションが構築される状況として、次の4つの場合を想定する。

場合1 実行が予定されている場合。例えば、定まった手順で回覧されるメッセージや、定期的な業務

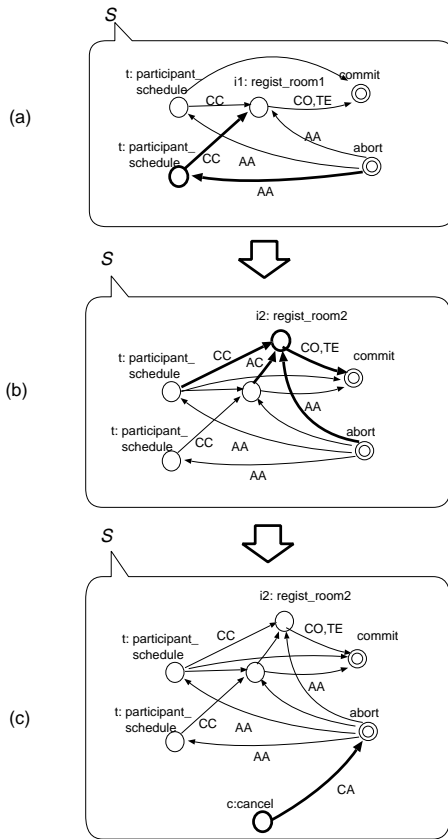


図3 メッセージトランザクションの動的な構築

の報告がある。

場合2 1つ以上生成されることは分かっているが、実際にいくつ生成されるか分からない場合。例えば、問題提起に対する対策案があげられる。

場合3 概念として定義することができるが、実際に生成されるか、またいくつ生成されるか分からない場合、例えば、ソフトウェア開発におけるバグ報告があげられる。

場合4 存在をあらかじめ予測できない場合。例えば、ソフトウェアの予想を上回るバグの発見により、コーディング作業を一旦中断して臨時的議論を開始するという予測しなかった状況になる場合があげられる。

それぞれの場合、メッセージトランザクションの中で混在して現れる。例として、図1に示したメッセージトランザクションが動的に構築される様子を図3に示す。部屋の予約 m-group `regist_room1` は不可欠なので、場合1に該当する。参加者が1人以上必要なので、各参加者の日程調整 `participant_schedule` は1つ以上生成される(場合2)。参加者が加わった時

は、新たな参加者のスケジュール調整の m-group と、関連するトランザクション従属性を追加する必要がある(図3(b))。`regist_room1`に失敗した場合は、代替的に別の部屋の予約 `regist_room2` を実行する(3(b))が、これは `regist_room1` が失敗しないと実行されない(場合3)。さらに、作業の進行中に、会議の中止や延期によりスケジュール調整のメッセージトランザクションの予期しない中断があるかもしれない(3(c))。もしこの中断を場合3として予測していない場合は、場合4に相当する。

我々は、メッセージトランザクションの構造の雛形である `mg` テンプレートを用意することで、メッセージトランザクションを利用者やシステムが任意の時点で適切に把握することを可能にする。

3.2 mg テンプレート

m-group クラス C の `mg` テンプレート $T(C)$ は、タプル

$$[Texp(C), Dep(C)]$$

で表される。 $Texp(C)$ は t 表現である。 t 表現は次のいずれかである。ただし、 C_i は m-group クラス、各 e_i は t 表現である。

- (1) 項 $n:C$.
- (2) 項 $n:\langle e_1, e_2, \dots, e_k \rangle$ (AND-group) .
- (3) 項 $n:[e_1, e_2, \dots, e_k]^+$ (+-group) .
- (4) 項 $n:[e_1, e_2, \dots, e_k]^*$ (*-group) .

上記で、各項の n は、 t 識別子と呼ばれる唯一のラベルである。 t 表現は AND-group, *-group, +-group を階層的に表現できる。

$Dep(C)$ は、 C のトランザクション従属性集合である。 $Dep(C)$ は、2.2, 2.3節のトランザクションイベント、トランザクション従属性中の S_i, S_j を形式的にそれぞれ n_i, n_j に置き換えて定義した際の、トランザクション従属性の集合である。

m-group クラス C を持つ m-group S に対し、 t 表現 $Texp(C)$ と $Child(S)$ の間には、次の制約を設ける。

- (1) t 表現 $Texp(C)$ について、(a)–(d) の制約を満たさなければならない。
 - (a) t 表現 $n_i:C_i$ に対して、 t 識別子 n_i および m-group クラス C_i を持つ S_i が S の子グループに存在しなければならない。また、この $n_i:C_i$ が (e) を満たす。
 - (b) t 表現中に AND-group の項 $n:\langle e_1, e_2, \dots, e_k \rangle$ がある場合、各 e_i が (a)–(d) を

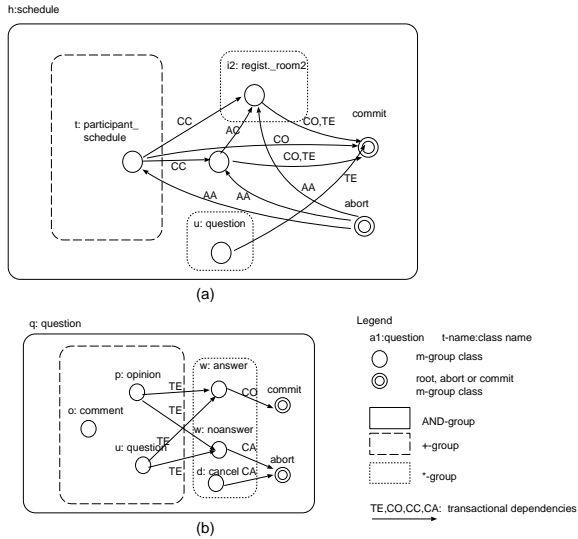


図4 mgテンプレートの例

満たす．また，各 e_i が (e) を満たす．

- (c) $TExp(C)$ 中に $+$ -group の項 $n: [e_1, e_2, \dots, e_k]^+$ がある場合，1つ以上の e_i が (a)–(d) を満たす．また，各 e_i が (e) を満たす．
- (d) t 表現中に $*$ -group の項 $n: [e_1, e_2, \dots, e_k]^*$ がある場合，各 e_i が (e) を満たす．
- (e) t 表現が項 $n_i: C_i$ なら， t 識別子 n_i を持つ m -group S_i は， C_i を m -group クラスに持たなければならない．
- (2) $Dep(C)$ 中のトランザクション従属性 $AA(n_1, n_2)$ について，それぞれ t 識別子 n_1, n_2 をもつ m -group S_1, S_2 が存在する場合， $AA(S_1, S_2)$ でなければならない． CC, CO 等の他のトランザクション従属性についても同様とする．

この制約に従い，メッセージトランザクションの動的な構築に対し次のような支援が可能である．図4(a)は，図3に対する mg テンプレートを示している．場合1にあたる $regist_room1$ は，AND-group の中に現れる． $schedule$ の m -group が生成された時点でシステムが $regist_room1$ を持つ空の m -group を生成し，関連するトランザクション従属性を割り当てる．場合2にあたる $participant_schedule$ は， $+$ -group の中に現れる． $schedule$ を持つ m -group が生成された時点でシステムが $participant_schedule$ を持つ空の m -group を生成する．その後利用者が $participant_schedule$ を持つ別の m -group を生成した場合も，システムが関連するトランザクション従属性を割り当てる．場合3にあたる $regist_room2$ は， $*$ -

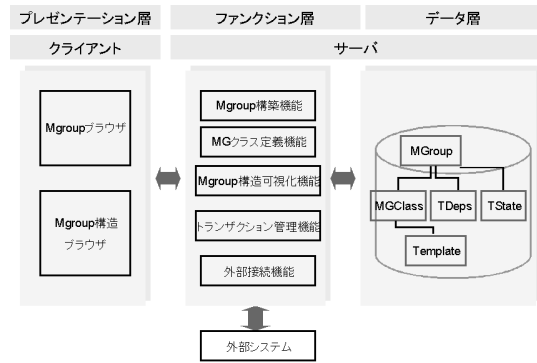


図5 システム構成

group の中に現れる．利用者が $regist_room2$ を持つ m -group を生成した場合，システムが関連するトランザクション従属性を割り当てる．場合4は，メッセージトランザクションの存在を予測できない場合であるため，対応する m -group の構造を t 表現に記述することはできないが，必要に応じて利用者が手動で m -group および関連するトランザクション従属性を定義する．

このように，作業の状況が明らかになるのに伴いメッセージトランザクションを動的に構築していくことが可能になる．

4. プロトタイプシステム

我々は，メッセージトランザクション管理の有用性を実証するため，提案するモデルに基づくプロトタイプシステムを開発している．開発言語として Java 言語を採用し，永続性管理のため ObjectStorePSE を用い，クライアント・サーバ間の通信のために JavaRMI を用いた．

4.1 システム構成

プロトタイプシステムは，図5に示す3層構造を持つ．データ層では， m -group や， m -group クラス， mg テンプレートの内容，利用者の情報を保持する．ファンクション層では，プレゼンテーション層からの要求に応じて m -group を構築し， m -group クラスおよび mg テンプレートの設定， m -group 構造ブラウザのための可視化の管理，メッセージトランザクションの実行管理をする．プレゼンテーション層では，メッセージトランザクションの内容の閲覧，編集をする．

以下の節では，特徴的なユーザインタフェースを示しながら，プロトタイプシステムの機能を述べる．

4.2 m -group クラスと mg テンプレートの定義

ファンクション層では， m -group クラス設定ファイルを読み込んで， m -group クラスおよび mg テンプレ

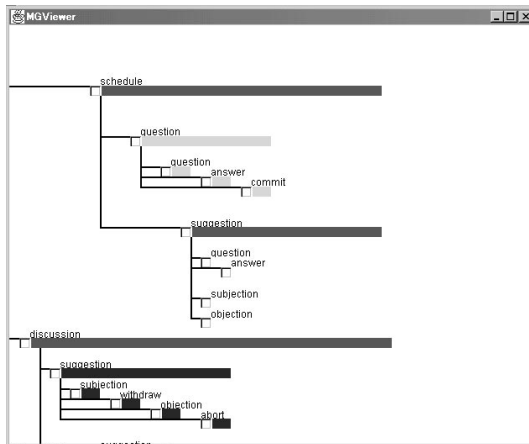


図 6 m-group 構造ブラウザ

トの定義を読み込む．以下では，その記述例を示す．

```
MgClass Question {
  Template = question;
}

Template question {
  Texp = s:<
    p:[c:comment, o:opinion, q:question]+,
    m:[a:answer, n:noanswer, c:cancel]*
  >;

  Dep = {
    TE(o,a),
    TE(o,n),
    TE(q,a),
    TE(q,n),
    CD(a,commit),
    CA(n,abort),
    CA(c,abort)
  };
}
```

この例では，m-group クラス Question および，mg テンプレート question の記述例を示している．mg テンプレート question に関しては， $Texp(Question)$ ， $Dep(Question)$ を定義している．

4.3 m-group 構造ブラウザ

利用者は，図 6 のようなインタフェースで m-group の構造を見ることができる．この画面では，縦方向に m-group の階層構造を表し，横方向に時間軸をとり，m-group の実行の期間を表示している．また，m-group の実行状態を色分けして表示する．また，図の例では，実行中である suggestion の次のメッセージの候補として，question, subjection, objection の m-group を表示している．このように，コミュニケーションに現れる時間的な長さを持つメッセージトランザクションとその状態を可視化する．

図 7 メッセージ編集

4.4 メッセージ編集画面

返信者が特定の m-group を指定すると，図 7 のようなウィンドウが現れる．ここでの操作は，空の m-group にメッセージ本体を入れることにより，m-group 生成時に定められたメッセージトランザクションを進行させることに相当する．mg テンプレートを基に，m-group クラスの候補が計算され，Class: の項目で選択することができる．またこの画面で，終了した m-group に commit または abort を指定することで，m-group の表す内容が成功して終了したか，失敗して終了したかを明確にする．

この画面は，通常の電子メールにおける返信と似ているが，あらかじめ送信者が返信者の発言内容の指針を与えることができ，返信者はその指針に従い返信できる点が異なり，メッセージのやりとりを高度に支援できる．指針は，m-group クラス reply のメッセージのみを許すような，3 節の場合 1 に従う決定的な指針の他，返答の他に質問 question や意見 opinion も許すといったように，場合 2 や場合 3 に従う選択的な指

針を用いることができる。また、指針で想定されない返答に対しても、場合 4 に従い手動で m-group クラスを追加することができる。

4.5 トランザクション管理機能

システムの実行中にトランザクションイベントが発生すると、トランザクション従属性によるメッセージトランザクションの充足可能性が検査される。例えば、日程調整 schedule の m-group クラスをもつ m-group の中で、reply の m-group が commit する前に request の m-group が commit しようとする、充足不能と判断され、エラーメッセージを表示する。

トランザクションイベントは、図 7 に示すウィンドウ上で生成する。このとき、システムによりトランザクション従属性が検査され、その結果 commit または abort になる m-group のトランザクションイベントが自動的に発生する。

5. 実用性の評価

プロトタイプシステムはその全体は完成していないが、m-group を生成することでメッセージトランザクションを動的に構築し、実行の状態を保存する機能は既の実現している。提案するモデルおよびプロトタイプシステムの実用性に関して、2 つの評価を行なった。1 つは、実際の協調作業でやり取りされた電子メールに我々のモデルを後から適用し、提案するモデルが実際の協調作業過程をうまく表現できることを確認するものであり、もう 1 つは、プロトタイプシステムの実装された機能を実際に使用し、その実用性を評価するものである。

5.1 現実の協調作業における実用性の評価

この節では、実際の協調作業でやり取りされた電子メールに我々のモデルを後から適用し、我々のモデルが実際の協調作業過程をうまく表現できるかについて評価した結果を示す。

対象として、ある研究室内のゼミの支援のためにやり取りされた電子メールを採用した。研究室内で、ゼミは年に数十回行なわれる。参加者は約 30 名で、ゼミの準備のためのやりとりはほとんど電子メールで行なわれる。ある特定のプロセス構造が存在するが、学生の発表順の調整のための議論など、あらかじめ構造を定義できない部分が存在する。我々は、約 4ヶ月間の間に発言された 200 通の電子メールでのやりとりの結果を観察し、その際の特徴を分類した。以下にその結果を示す。

(1) 複数のメッセージでまとまった意味を持つ、つまり親グループまたは子グループを持つ m-group を構

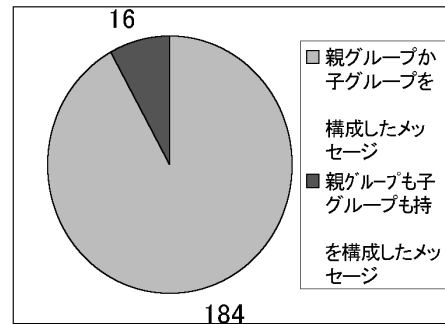


図 8 子グループまたは親グループを持つ m-group を構成したメッセージの数

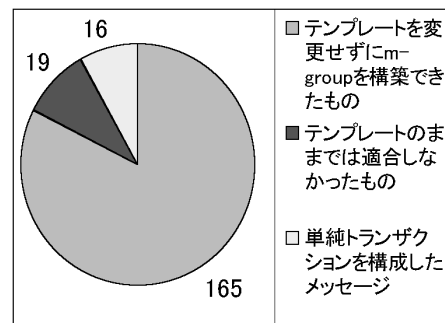


図 9 mg テンプレートを変更せずにできたメッセージの数

成すると考えられたメッセージは 184 通、(2) 単純な告知など、他のメッセージとの間でまとまった意味を持たず、子グループと親グループを持たない m-group を構成したメッセージの数は 16 通だった。

また mg テンプレートは、初めの数十通のメールを読んで作業のプロセスを解析することで、10 個用意した。代表的なものを図 4 に示す。(1) のうち、(1a) この mg テンプレートに従って動的に構築していった m-group と現実の状況が矛盾することなく生成できたメッセージは 165 通、(1b) そのままでは矛盾が生じて m-group に適合できなかったメッセージが 19 通だった。(1b) のメッセージは、(1b-1) やりとりが電子メール以外の手段で行なわれたため、余分な m-group が残ってしまうものが 3 通、(1b-2) 一旦 commit された m-group を abort して作業をやり直しているものが 16 通あった。

(1) のうち、1 つのメールの中に複数の内容を含み、複数の m-group クラスを持つと考えられるものが 29 通あった。

また、生成されたメッセージトランザクションの数は 119 個だった。

生成されたメッセージトランザクションは、メッセー

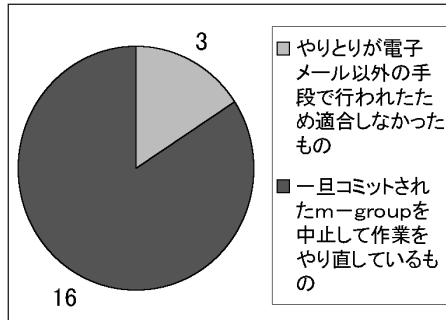


図 10 mg テンプレートのままでは適しなないメッセージの内訳

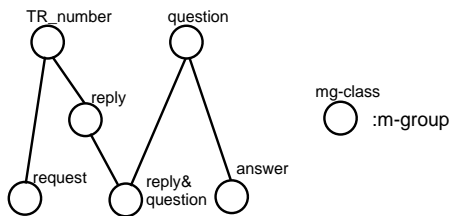


図 11 複数の親を持つ m-group

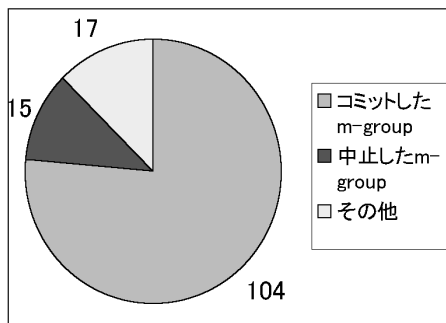


図 12 m-group の動作

ジトランザクションの中にメッセージトランザクションを持つといった、階層化された構造をしていた。ただし、図 11 に示すように、複数のメッセージトランザクションの子になる構造も存在した。

生成されたメッセージトランザクションのうち、(a) commit したと考えられるものは 104 個、(b) abort したと考えられるものは 15 個だった。(c) 残りの 17 個については、(c-1) 実際には終了しているが電子メールからは読みとれないか、(c-2) 実際に実行中のメッセージトランザクションであると考えられる。

以上のことから、以下の結論が得られる。(1)(2) より、かなりのメッセージがメッセージトランザクション

を形成することが観察できる。また (1a),(1b) および、作業の実行時に 10 個の mg テンプレートを組み合わせることで 119 個の m-group を生成することができたことより、mg テンプレートを用いてメッセージトランザクションを動的に構築することの有効性が確認できる。(a)(b) より、m-group の中では、commit, abort するものが存在するため、トランザクションイベントを導入して状態を管理することは重要であることが確認できる。さらに、メッセージトランザクションの構造と動作に関して、以下の特徴をあげることができる。1. 複数の m-group を親に持つ m-group が存在することで、メッセージトランザクションの構造が DAG 構造になることがある。2. 一旦 commit された m-group が abort される可能性がある。3. 電子メール以外で行なわれるやり取りに対してはメッセージトランザクションを整合性を持って構築できないことがある。2,3 に対する対処は、今後の課題として残される。

5.2 プロトタイプシステムの評価

プロトタイプシステムを実際に用いて、5 名の参加者による簡単な議論を行ない、30 個の m-group を生成した。その結果、質問の m-group クラスを持つ m-group が 10 個、回答の m-group クラスを持つ m-group が 6 個、質問の取り下げを意味する m-group クラスを持つ m-group が 1 個、コメントの m-group クラスをもつ m-group クラスが 13 個生成された。また、回答済みの質問 6 個は commit し、取り下げられた質問は abort させられた。この時点で実行中の m-group は 3 個であり、回答されていない質問が 3 つ残っているというように、協調作業の状況を明確にしている。

また、この議論の中では、あるテーマに関する一連の質問といったように、いくつかの m-group の親となる m-group が存在する。このような m-group は、実際に実行されてから認識されることが多い。現時点では実行中の m-group を検索する機能がないが、この機能を実装することでより作業の状況を明確化できると考えられる。

議論の参加者からは、m-group クラスにより発言の意図と議論の流れを明確にできるという点は評価されたが、使用できる m-group クラスを検索する方法や、m-group クラスにより分類した m-group の表示、指定した m-group クラスを持つ m-group の中身と同じキーワードを持つ m-group を検索したいという要望が出された。

6. 考 察

提案するモデルは、コミュニケーションを、プロセ

スや、人間の介在するトランザクションを含めた広い範囲で捕らえている点が特徴である。時相論理に基づく従属性表現により、直列または並列、または選択的に実行されるといったプロセス間に与えられる従属性の他、入れ子トランザクションや代替トランザクションといったトランザクションを記述できる。

構造化議論モデル (IBIS³⁾では、議論の状態を発言者が指定することにより、議論の過程を明確化する。The Coordinator⁴⁾では、協調作業での人間の行動は、参加者間のコミュニケーションにより開始されるという見地に基づき、「要求」「約束」のような属性を持つ発言により起動されるプロセスの状態を観測可能にしている。ConversationBuilder⁹⁾¹⁰⁾²⁾ではさらに、プロセスの構造を利用者が設計し、作業の実行中に動的に詳細化することを可能にしている。これらの研究は、メッセージにその特徴的な属性を付加する点や、メッセージのやり取りがプロセスの進行の情報を含んでいる点では我々と同じであり、ConversationBuilderの、プロセスの構造を動的に構築可能である点は我々のモデルと同じであるが、トランザクションの機能を持たない点がメッセージトランザクションと異なる。現実には作業の目標が失敗することがあり、失敗時の作業を明確にすることが重要である。また種々の情報処理システムで支援される環境では、作業をシステムが受け持つ場合もあり、システムで実行されるトランザクションの動作を明確にする必要がある。我々のモデルは、abortにより例外的な状態をあらかじめmgテンプレートに記述でき、トランザクションイベント間の従属性により、失敗時の代替的な目標を記述したり、失敗時に影響を受ける範囲を特定することができる。また、トランザクション従属性を用いて種々のトランザクションモデルを記述できる。

文献¹⁴⁾では、メッセージ上のトランザクションを、目標とする作業の状態に至るまでの参加者間の一連の作業と定義している。さらに本概念によって、ある研究室内の約1000通の電子メール交信記録を調査したところ、55%の電子メールは質問や要求などのトランザクションを形成していたと報告されている。ただ、この研究では、入れ子構造などの構造化されたトランザクションを構築する機能は検討されていない。メッセージトランザクションは議論の分岐、統合などといった人間の協調した作業を表すものとなるため、このような作業を表現するためには構造化されたトランザクションモデルを導入する必要がある。さらに、メッセージは作業実行時に動的に生成されるため、メッセージ上のトランザクションも実行時に動的に構築していく手

法が必要となる。

文献¹³⁾では、木構造を持つ作業のトランザクションに対し、形式的な分割/結合操作を定義することで動的に構造を変更する手法を提案している。本論文で提案するモデルでは、m-group クラスおよびmgテンプレートにより、現実の作業における意味を取り入れたメッセージトランザクションを構築する点が異なるが、一旦構築したメッセージトランザクションの構造を変更することは考慮していないため、メッセージトランザクションの構造の動的な再構成については今後の課題である。

ビジネス分野で実用化されているワークフロー管理システムは、定型的でない、あるいは一時的な作業を不得手とし、動的ワークフローや適応可能 (adaptive) ワークフローとして盛んに研究されている¹³⁾¹⁷⁾。また、ワークフローの処理単位である各アクティビティにトランザクション管理機能を導入する研究¹⁸⁾もされている。ワークフロー管理システムでは、プロセスの構造は実行される前にあらかじめ定義されることを前提としている。しかし、コミュニケーションにおいては、プロセスの構造を事前に予測できず、作業の実行時または実行後に決定される場合が存在し、既存のワークフロー管理システムで管理することは難しい。

今回のシステムでは、m-group を永続化するためにオブジェクト指向データベースを用い、m-group に対する処理の直列化可能性を保証するためにトランザクション機能を利用したが、メッセージトランザクションの管理は独自に実装する必要がある。

メッセージトランザクションおよびその管理は、今後協調作業への応用システムにおいて必要なデータベースの機能となる可能性が高い。その理由として、上記で評価した実用性の他に、次の点をあげる。

- データベースとコミュニケーションの境界領域に注目した技術である点。データベースがさまざまな分野に応用されるためには、人間の介在を考慮するなど、応用分野の意味論を取り入れたモデルを検討する必要がある。提案するメッセージトランザクションは、メッセージのやり取りの意図とトランザクションの動作を関連づけることで、利用者の視点を取り入れたトランザクションを構築するものである。
- 提案するモデルは、DAG 構造や、あらかじめ定義されていない構造を許すことで、協調作業を一般のプロセス支援システムに比べより一般的にとらえたものである点。不特定多数の作業者が電子メールのみのコミュニケーションによってソフト

ウェアを開発するような状況など、幅広い応用分野が考えられる。

7. おわりに

本論文では、非同同期型コミュニケーションに出現する予測困難なメッセージのやり取りおよびその現実の作業への影響に対し、トランザクションの概念を導入し、作業中に動的に構築することで作業の状況を明確にする手法を提案し、現実の電子メールの蓄積に適用してその有効性を確認した。また、プロトタイプシステムの開発例と実際に使ってみた際の実用性を確認した。

プロトタイプシステムは全体は完成していないが、現在完成している部分だけでも、メッセージのやり取りをトランザクションで表すことにより、完了した作業や、実行中の作業といった状態を区別でき、作業の状況を明確化できる。今回提案した手法では、メッセージトランザクションの進行や、commit, abortの権限に制限は設けなかったため、メッセージトランザクションの実行に混乱が生じる可能性がある。この問題に対する検討は、今後の課題である。

謝辞 有用なご指摘をいただきました査読者および活発に議論していただきました九州大学安浦研究室の諸氏に感謝致します。

参考文献

- 1) Attie, P., Singh, M., Sheth, A. and Rusinkiewicz, M.: Specifying and Enforcing Intertask Dependencies, *Proc. 19th Int'l Conf. Very Large Databases*, pp. 134-145 (1993).
- 2) Bogia, D. P., Tolone, W. J., Kaplan, S. M. and de la Tribouille, E.: Supporting dynamic interdependencies among collaborative activities, *Proc. ACM Conf. Organizational Computing Systems*, pp. 108-118 (1993).
- 3) Conklin, J. and Begeman, M. L.: gIBIS: A Hypertext Tool for Exploratory Policy Discussion, *ACM Trans. Office Information Systems*, Vol. 6, No. 4, pp. 303-331 (1988).
- 4) Flores, F., Graves, M., Hartfield, B. and Wingrad, T.: Computer Systems and the Design of Organizational Interaction, *ACM Trans. Office Information Systems*, Vol. 6, No. 2, pp. 153-172 (1988).
- 5) Gray, J.N.: The Transaction Concept: Virtues and limitations, *Proc. 7th Int'l Conf. Very Large Databases* (1981).
- 6) Inoue, S. and Iwaihara, M.: Structured Message Management for Group Interaction, *Proc. Int'l Workshop. New Database Technologies for CSCW and Spatio-Temporal Data Management (NewDB'98)*, Singapore (1998).
- 7) Iwaihara, M., Inoue, S. and Matsuo, H.: On Unifying Message and Transaction Management for Collaborative Design Work, *Proc. Int'l Symp. Digital Media Information Base (DMIB97)*, Nara, pp. 300-304 (1997).
- 8) Jajodia, S. and Kerschberg, L.(eds.): *Advanced Transaction Models and Architectures*, Vol. 2, Kluwer Academic (1997).
- 9) Kaplan, S. M., Carrol, A. M. and MacGregor, K. J.: Supporting Collaborative Process with ConversationBuilder, *Proc. ACM Conf. Organizational Computing Systems*, pp. 69-79 (1991).
- 10) Kaplan, S. M., Tolone, W. J., Bogia, D. P. and Bignoli, C.: Flexible, Active Support for Collaborative Work with ConversationBuilder, *Proc. ACM Conf. Computer-supported Cooperative Work*, pp. 378-385 (1992).
- 11) Kellner, M. I. et al.: Software Process Modeling Example Problem, *Proc. 6th Int'l Software Process Workshop*, pp. 19-29 (1990).
- 12) Klein, J.: Advanced Rule Driven Transaction Management, *Proc. COMPCON* (1991).
- 13) Liu, L. and Pu, C.: Methodological Restructuring of Complex Workflow Activities, *Proc. 14th Int'l Conf. Data Engineering*, California, pp. 342-350 (1998).
- 14) Milewski, A. and Smith, T.: An Experimental System for Transaction Messaging, *In Proc. of the Int'l ACM SIGGROUP Conf. on Supporting Group Work(GROUP97)*, pp. 325-330 (1997).
- 15) Moss, J. E. B.: *Nested Transactions: An approach to reliable distributed computing*, PhD Thesis, Massachusetts Institute of Technology, Cambridge, MA (1981).
- 16) Ramamritham, K. and Chrysanthis, P.K.: Advances in Concurrency Control and Transaction Processing, *IEEE Computer Society Executive Briefing* (1997).
- 17) Reichert, M. and Dadam, P.: A Framework for Dynamic Changes in Workflow Management Systems, *Proc. Int'l Conf. Database and Expert Systems Applications* (1997).
- 18) Worah, D. and Sheth, A.: Transactions in Transactional Workflows (1997). in⁸⁾.

(平成 1999 年 3 月 20 日受付)

(平成 1999 年 10 月 5 日採録)

(担当編集委員 牧之内 顕文)



井上 創造 (学生会員)

九州大学工学部情報工学科平成 9 年卒, 同大学院システム情報科学研究科情報工学専攻修士課程平成 11 年修了. 同年 4 月より同専攻博士後期課程に在籍. データベースの協調作業支援への応用の研究に従事.



岩井原瑞穂 (正会員)

九州大学工学部情報工学科昭和 63 年卒, 同大学院工学研究科情報工学専攻修士課程平成 2 年修了, 同博士後期課程平成 5 年修了, 博士 (工学). 平成 5 年より九州大学大学院総合理工学研究科情報システム学専攻助手, 平成 8 年より現在まで同大学院システム情報科学研究科情報工学専攻助教授. データベース質問言語, 質問処理, 協調作業支援システムの研究に従事. 電子情報通信学会, ACM, IEEE 会員.