柔軟なプロセス管理のためのプロセス設計履歴の管理

井上, 創造
九州大学大学院システム情報科学研究科情報工学専攻

岩井原, 瑞穂
九州大学大学院システム情報科学研究院情報工学部門

柔軟なプロセス管理のためのプロセス設計履歴の管理

井上 創造* 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
† 九州大学 大学院システム情報科学研究院 情報工学部門

〒 816-8580 福岡県春日市春日公園 6-1

E-mail : {sozo,iwaihara}@c.csce.kyushu-u.ac.jp

本論文では，コミュニケーションプロセス，つまり，プロセスの構造があらかじめ明確には定義されず，作業の参加者間の合意によって定義され実行されるプロセスを取り上げる．またコミュニケーションプロセスを管理するシステムとして，参加者の議論を記録し，その記録に沿ってコミュニケーションプロセスを生成し実行することができる M-Trans システムを述べる．コミュニケーションプロセスは設計と並行して実行される可能性があり，設計と実行の間にやりとりが行なわれることも考えられる．コミュニケーションプロセスの管理は，ワークフローの例外処理や動的再構築を含めた，予測困難な状況への適応を実現するという意味で重要である．本システムはプロセスとコミュニケーションを統合的に記述するモデルに基づく．

1 はじめに

現実の世界においては，ワークフロー管理システム (Workflow Management System, **WfMS**) において扱うことが 難しい協調作業が存在する．本稿ではそのような作業のためのモデルとしてコミュニケーションプロセスを導入する．コミュニケーションプロセスは，プロセスの構造があらかじめ明確には定義されず，作業の参加者どうしの合意によって定義され実行されるプロセスである．コミュニケーションプロセスは設計と並行して実行される可能性があり，設計と実行の間にやり取りが行われることも考えられる．

コミュニケーションプロセスの管理は，ワークフローの例外処理や動的再構築を含めた，予測困難な状況への適応を実現するという意味で重要である．現実の世界においては，頻繁に変わりうる環境や予測困難な人間の行動にさらされるため，柔軟なプロセス管理が WfMS において重要な要求である．

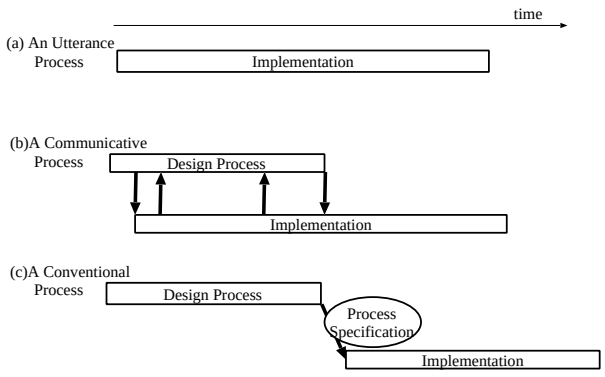

図 1: プロセス設計の分類

本論文では， コミュニケーションプロセスの設計履歴を記録し，その設計履歴に沿ってコミュニケーションプロセスを生成，実行することができる M-Trans システムを提案する． コミュニケーションプロセスは，プロセスと利用者間のコミュニケーションを統合的に記述するメッセージトランザクションモデル [4, 10, 5] に基づく．

2 コミュニケーションプロセス

この章では，コミュニケーションプロセスを導入する．

2.1 プロセスの設計の分類

協調作業の例として， 学術会議のプログラムを作成する場合を考える．プログラムを作成する担当の人々が，定められた手順に従い発表者にタイトルを請求する．その請求のためのプロセスでは，誤りの訂正のための例外的な処理が行なわれる可能性がある．誤りの訂正は，発表者へ訂正を要求することで行う．タイトルがそろった時点で，プログラム作成の担当者は発表をセッションに分け，セッションの座長を割り当てる．このための議論は，提案や質問といったプログラム作成担当者の発言により進行する．このようなプロセスはその設計のされかたにより次のように分類できる：

- **発言プロセス** (Utterance Process)：発言自体により実行されるプロセス．上記の例で座長を打診する発言は，打診した時点で返答を期待しているため，要求に引き続く返答というプロセスを実行していると考えることができる．発言プロセスは，構造化議論モデル [2, 7] で実用化されている．
- **コミュニケーションプロセス** (Communicative Process)：作業者間の合意に基づき実行されるプロセス．プロセスの構造は合意の中に示唆されるが，ワークフローのような形で明確には定義されない．上記の例で誤りの修正は，場合により対応が異なるという例外的な対応であり，早急な修正が必要なため，修正方法が意思決定の中で決定され，プロセスを明示的に記述しないまま実行される．

- **ワークフロープロセス** (Conventional Process)：議論の結果を元に，あるいは設計者がワークフローを明記する場合．例えば，タイトルを要求する方法は，プログラムを作成する担当の人々によってその手順をワークフローで記述することが可能である．

以下では，他のプロセスを設計するために行なわれる意思決定のプロセスを，**設計プロセス**と呼ぶ．

図 1(a) は発言プロセスを示している．発言プロセスは設計プロセスを独立して持たず，参加者の発言によりその場で生成され実行される．

図 1(b) はコミュニケーションプロセスを示している．設計プロセスはコミュニケーションプロセスをシステムが理解できる形では出力しない．また，コミュニケーションプロセスの実行は，設計プロセスの最中に行われる可能性もあり，コミュニケーションプロセスと設計プロセスがやり取りを行うことも考えられる．

図 1(c) はワークフロープロセスを示している．設計プロセスはワークフロープロセスの仕様を出力し，ワークフロープロセスの実行は設計プロセスが終了した後に行われる．

コミュニケーションプロセスの管理は，ワークフローの例外処理や動的再構築を含めた，予測困難な状況への適応を実現するという意味で重要である．現実の世界は，頻繁に変わりうる環境や予測困難な人間の行動にさらされる．柔軟なプロセス管理が WfMS において重要な要求である．予測困難な状況においては，その場で適応が検討され実行されるため，適応自身がコミュニケーションプロセスの一部と言える．

適応の設計プロセスにおいて参照される情報として，その時点の作業の状況と，現在のプロセスの設計履歴をあげることができる．前者は文献 [3] において検討されているが，後者は検討されてこなかった．プロセスの設計履歴を記述することは，プロセスの目的や構造，それらに対する議論を取り出すことができる点で有用である．

コミュニケーションプロセスのための設計プロセスについては，2.3 節で取りあげる．

2.2 M-Trans システム

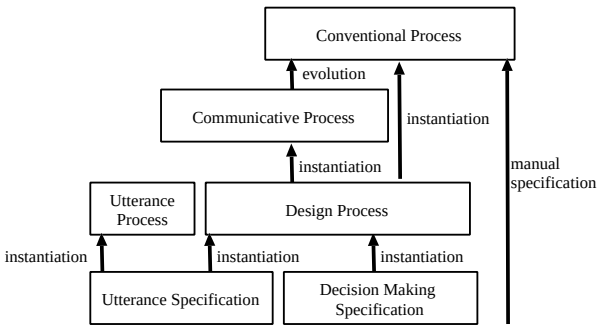

図 2: M-Trans システムにおけるプロセスの構成

コミュニケーションプロセスやワークフロープロセスの実行，それらの設計プロセスは主に，参加者間のコミュニケーションにより行われる．我々はワークフローと非同期型コミュニケーションを統合する M-Trans システムを導入する．M-Trans システムの特徴は次のとおりである．

- 発言プロセスの実行とコミュニケーションプロセスの構築が可能である．
- プロセスの仕様を全て記述することなく実行することが可能である．
- 設計プロセスに記録された設計履歴に従ってプロセスを構築することが可能である．

発言プロセスの実行は，人間の発言のモデルを用意し，そのモデルに基づいてプロセスを実行することによって行なう．コミュニケーションプロセスの構築は，意思決定のプロセスをモデル化し，そのプロセスを実行することにより行なう．意思決定プロセスのモデルは，SIBYL[7] といった構造化議論モデルに基づく．どちらのモデルも 3.2 節で述べる mg テンプレートで記述される．さらに，意思決定の記録をテンプレートとしてコミュニケーションプロセスを生成，実行する方法を示す．mg テンプレートはまた，ワークフロープロセスを記述することが可能であるため，コミュニケーションプロセスをその設計履歴を保存したままワークフロープロセスへ進化させることが容易である．図 2 は M-Trans システムにおけるプロセスの構成を示す．

M-Trans システムは，発言プロセスのモデルであるメッセージトランザクションモデル [4, 10, 5] に基づく．本稿では，メッセージトランザクションモデルをコミュニケーションプロセスを記述できるように拡張する．これについての詳細は 4 節で述べる．

2.3 コミュニケーションプロセス

SIBYL[7] によると，意思決定のためのプロセスの特徴は次のとおりである．

- まず，解決されるべき問題が記述される．問題は議論がすすむにつれていくつかの部分問題に分割される．上記の例では，会議のプログラムを出版することが問題であり，それはタイトルを集めることとプログラムを編集するという部分問題に分割される．
- 問題や部分問題に対する解決策が提案され，検討される．例えば，タイトルの収集を郵送で行なうか，電子メールで行うかといったいくつかの解決策が提案される．
- 解決策に対し，賛成，反対といった議論がなされる．これらは後でそれぞれの解決策を評価する際に有用である．

M-Trans システムにおいては，問題と解決策をプロセスの原始的な仕様ととらえる．意思決定において生成された問題はプロセスになり，その部分問題は子プロセスとなる．例えば，タイトルを収集するプロセスとプログラムを編集するプロセスは，プログラムを出版するプロセスの子プロセスとなる．

コミュニケーションプロセスのための設計プロセスにおいては，全ての子プロセスが記述されるわけではない．いくつかは実行時に動的に生成される発言プロセスとなる．このような発言プロセスに対して，設計プロセスに

おける意思決定の記録は，参加者がどのように発言プロセスを生成すべきかという間接的な基準になりうる．例えば，電子メールとFAXのいずれを使うかを発言プロセスにより動的に選択する場合には，設計プロセスにおいて賛成意見が多い方が選択されるべきである．M-Transシステムにおいては，問題や，解決策，意見は発言プロセスの実行時に参加者に表示される．

構造化議論モデルから受け継ぐ特徴に加え，我々はプロセスを設計するための語彙を用意する．つまり，precede, follow，role assignment，そして resource assignment である．これらについては 4.2 節で述べる．.

3 メッセージトランザクションモデル

この節では，メッセージトランザクションモデルを述べる．

3.1 m-group

メッセージトランザクションモデルでは，メッセージやプロセスは **m-group** として統一的に扱う．図 3 は m-group の例である．m-group の中では，発言者の意図や，プロセスの目的を **mg-class** という属性により表す．mg-class の例として,「Issue」「Argument」「Position」といった構造化議論モデル（IBIS）[2] で用いられているものや，ソフトウェアプロセス [6] における「バグ発見」「改善要求」「設計変更」「評価結果」があげられる． m-group は，複数の m-group を**子グループ**として持つことで，一連のメッセージのやりとりを表すことができる．子グループを持つ m-group を**複合 m-group** と呼び，子グループを持たない m-group を**単純 m-group** と呼ぶ．m-group S では，生成される事象を root，子グループが生成される事象を「実行」，意図を達成して終了する事象を commit，達成せずに終了する事象を abort と呼ぶ．m-group は全ての子グループが「終了」，つまり commit または abort した後に commit することが許される． また，m-group は，m-group への参加者を**役割**，使用するデータの情報を，**リソース** として複数個持つことができる．

図 3 で，m-group 間に与えられる有向枝は**トランザクション従属性 (t-dep)** である． t-dep は，m-group の実行に関する m-group 間の従属性である．t-dep がいずれかが終了していない m-group の間に割り当てられている場合，t-dep は WfMS におけるアクティビティ間のコントロールフローと同じ意味を持ち，始点が commit した後でなければ終点は実行されない．一方，両端の m-group が終了している場合は，t-dep は終点が始点の内容を参照していることを意味する．t-dep はあらかじめ決められた文字列の集合の要素をラベルとして持つ．4 章ではこのラベルを拡張し，設計プロセスとコミュニケーションプロセスの意味を持たせる．

図中で root, commit および, abort イベントは二重円で示されている．入力値は m-group の左側に，出力値は右側に示されている．

本システムにおいては，mg テンプレートの定義に従い，t-dep を動的に割り当てることが可能である．次節でこの方法を述べる．

3.2 mg テンプレート

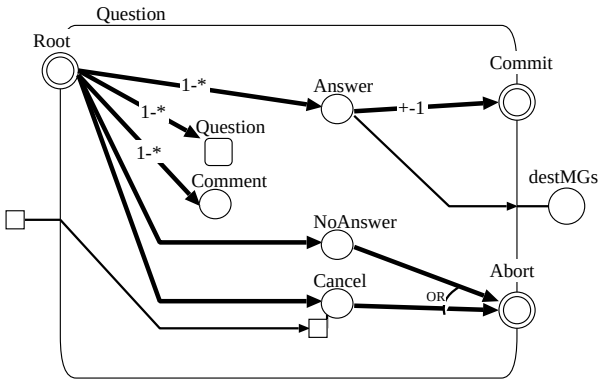

図 5: 発言プロセスのための mg テンプレート

mg テンプレート は，m-group の仕様を記述したものであり，1 つの mg-class に対し 1 つが割り当てられる．mg テンプレートの定義に従い m-group が生成されることを**インスタンス化**と呼ぶ．m-group のインスタンス化については 3.3 節で述べる．m-group の インスタンス化はある m-group S の参加者により行なわれ，インスタンス化された m-group は S の子グループとなる．このインスタンス化は同時に S の実行を意味している．

mg-class `Question` と `Decision` に対する mg テンプレートの例を，それぞれ図 4 および図 5 に示す．

mg テンプレートは，1 つの mg-class に対して 1 つ定義される．mg テンプレートの左端に Root , 右端には Commit と記された 2 重丸が書かれる．

図 4 の有向枝は mg テンプレートがインスタンス化されたときに与えられる t-dep を表し，**tdep テンプレート**と呼ぶ． tdep テンプレートの両端には，m-group，リソース，役割， 両側が m-group の場合には，AND-split, OR-join といった分岐と結合を与えることができる．AND-split は，始点が commit した後に全ての終点がインスタンス化されることを意味し，OR-join は，始点のうち 1 つでも commit すれば終点がインスタンス化可能であることを意味する．

tdep テンプレートの両端には，インスタンス化された m-group の多重度に関する次の制約を与える．

- 1: 反対側に接する要素 1 つに対し，同じ側の要素が 1 つ存在する．
- *: 反対側に接する要素 1 つに対し，同じ側の要素が任意個存在する．
- +: 反対側に接する要素 1 つに対し，同じ側の要素が 1 つ以上存在する．
- ALL: 反対側に接する要素 1 つに対し，同じ側の要素が 1 つ以上存在し，その全てが反対側の 1 つの要素と接する．

tdep テンプレートの両端に用いるこれらの属性により，tdep テンプレートを 1–ALL dependency，+–* dependency のように名づける． それぞれの tdep テンプレートの意味を次に述べる．

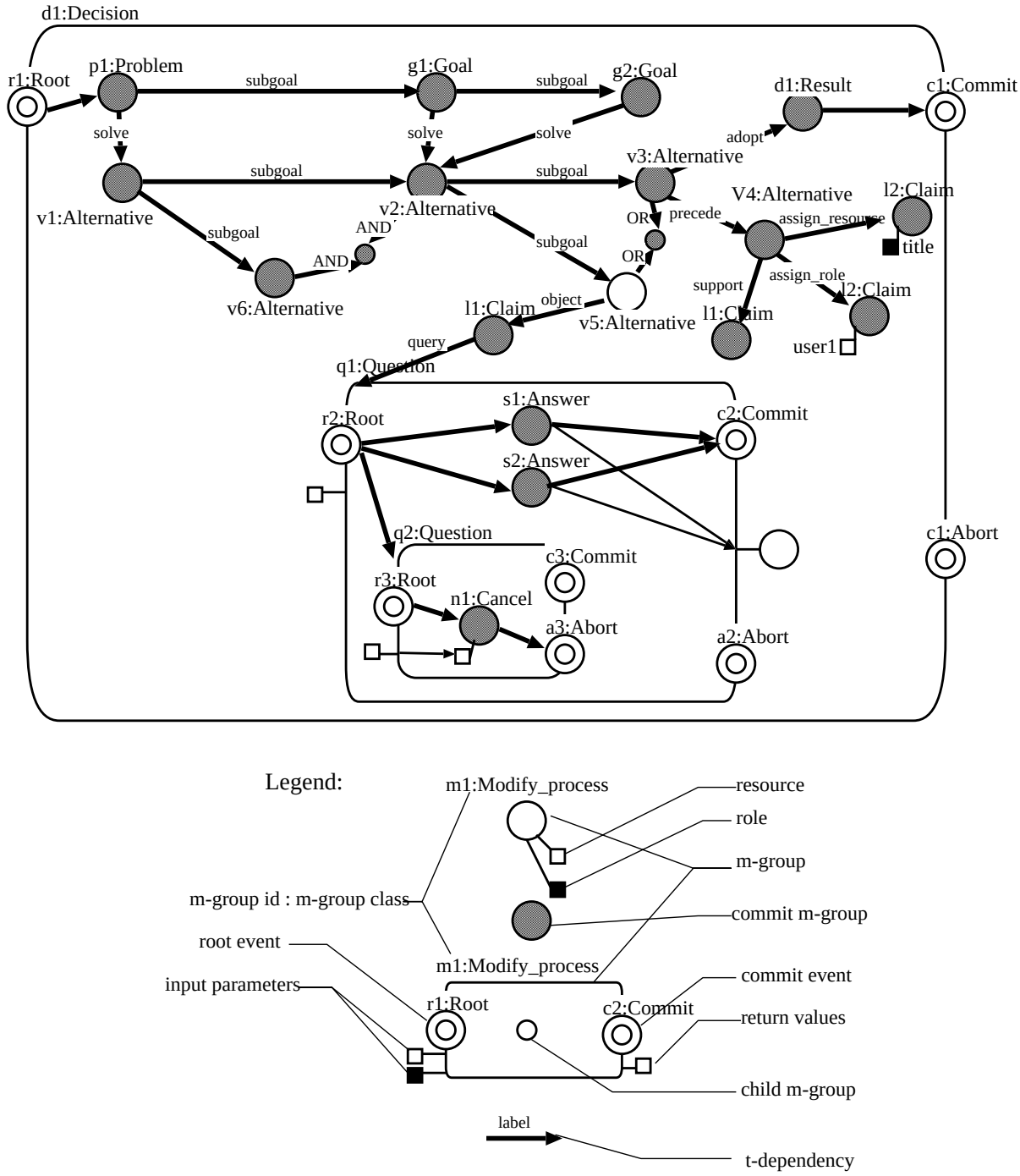

図 3: m-group の例

- mg テンプレートから mg テンプレートへの tdep テンプレート：終点がインスタンス化する際に t-dep がインスタンス化される．さらに，終点のインスタンス化は始点が commit した後に行なわれる．
- リソースからリソースへの tdep テンプレート：終点のリソースが属する m-group がインスタンス化される際に，始点のリソースの値が終点のリソースに割り当てられる．
- 役割から役割への tdep テンプレート: m-group における終点の 役割の値は始点の 役割の値となることを表す．
- 入力値からの tdep テンプレート: 入力値として渡されるそれぞれの要素がインスタンスにおける終点の値となる．
- 出力値への tdep テンプレート: インスタンスにおける始点の値が，出力値の要素に加わる．

3.3 発言プロセスの実行

発言プロセスは人間の発言の分類に基づくため，発言プロセスの分類が協調作業の種類によって大きく変わることはない．M-Trans システムでは発言プロセスのためのあらかじめ定められた mg テンプレートを持つ．図 5 はその

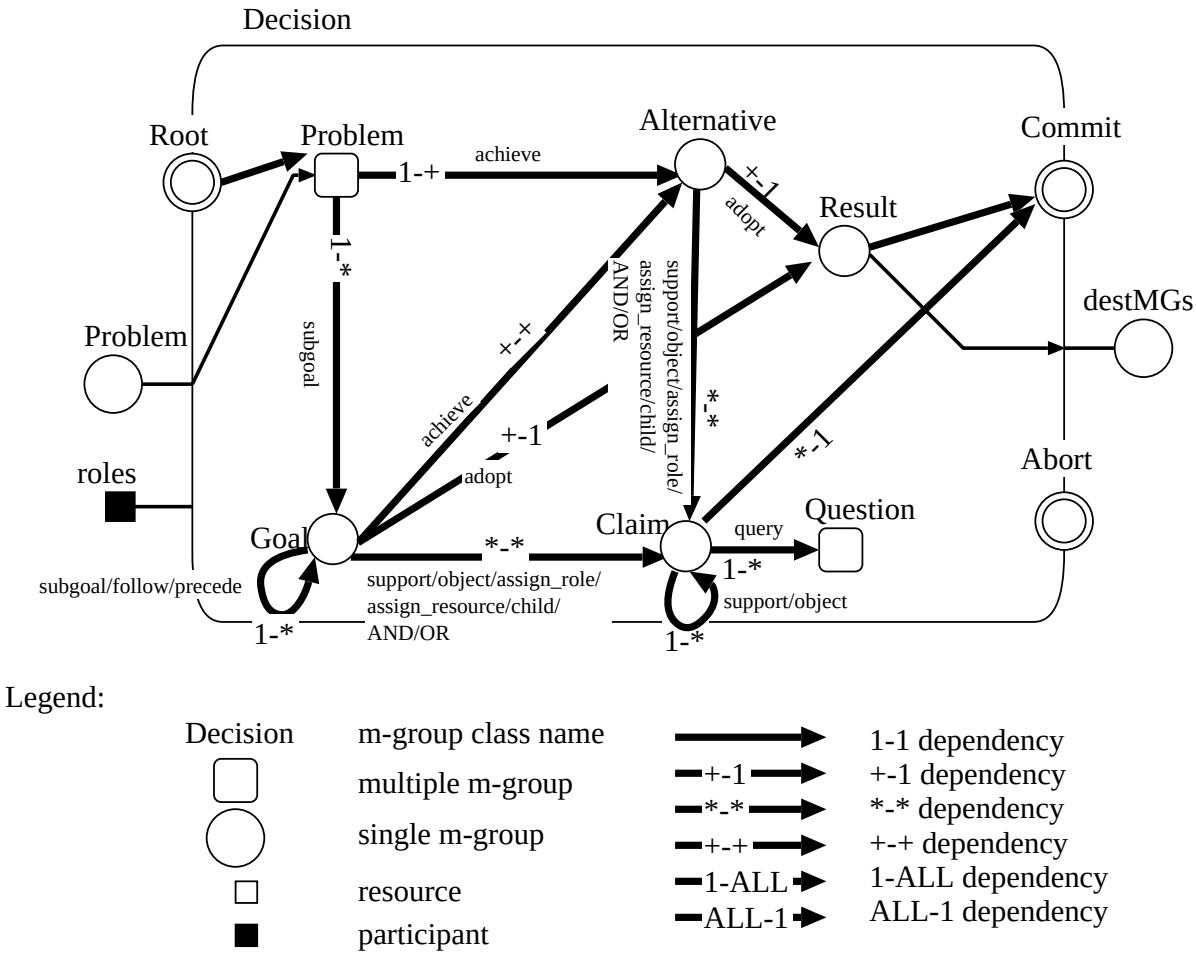

図 4: mg テンプレートの例

例である．図 3 中の `q1:question` は，図 5 の mg テンプレートを元に発言プロセスを実行する例である．図 3 において，m-group `d1` の参加者は m-group `l1` に対する質問のメッセージを生成し，mg-class `Question` を割り当てている．この操作により `Question` がインスタンス化され，`l1` からの t-dep を持つ m-group `q1` になる．`Question` のインスタンス化に続いて，mg-class `Answer` を持つ返答の単純 m-group を生成したり，mg-class `Question` をもつ複合 m-group によりさらに詳細な質問を行なうことが可能である．単純 m-group は生成されると同時に commit する．一方複合 m-group は，`q2:Question` のように再帰的にインスタンス化することにより実行に時間的な長さを持つことが可能である．`q2` は最初の参加者によりキャンセルされるが，mg テンプレートには，入力値からの tdep テンプレートによりキャンセルを行なうことのできる参加者は，最初の参加者のみであることが記述されている．

4 M-Trans システム

この節では，コミュニケーションプロセスの生成と実行を提供する M-Trans システムを述べる．

4.1 コミュニケーションプロセスの設計プロセス

コミュニケーションプロセスの設計プロセスのために，特定の mg テンプレートを用意する． 設計プロセスの参加者はその mg テンプレートをインスタンス化しながらコミュニケーションプロセスを設計する．図 4 はそのための mg テンプレートである．さらに，t-dep のラベルのうち特定のものに，設計プロセスのための意味を与える．これらの mg テンプレートと t-dep の一部は，SIBYL[7] から導入する．図中の mg-class と t-dep を次に説明する．

`Problem` は解決すべき問題，`Alternative` は問題を解決するための解決策の選択肢，`Goal` は解決策の結果予想される結果を，`Claim` は発言者の主張，`Question` は，発言者の質問，`Result` はどの解決策が採用されたかを示す結論を表す．

`subgoal` の t-dep は，終点が `Goal` または `Problem` を持つ始点の一部分であることを示す．`achieve` の t-dep は，`Alternative` を持つ終点が `Goal` を持つ始点を達成することを示す．`support` または `object` の t-dep はそれぞれ，終点が始点に対する賛成，または反対の主張を意味する．`query` の t-dep は，終点が始点に対する質問であることを意味する．

上に述べたラベルは構造化議論モデルから導入したも

のである．これらに加え本手法では，follow，precede，child，OR，AND，assign_role，
そして assign_resource という，プロセスを構築するための次のラベルを用意する．これらは，コミュニケーションプロセスがインスタンス化されるときに意味を持つ．follow は，コミュニケーションプロセスの中で終点に対応する m-group は始点に対応する m-group より後に実行されることを意味する．precede は，コミュニケーションプロセスの中で始点に対応する m-group は終点に対応する m-group より後に実行されることを意味する．OR を持ち終点を共有する複数の t-dep は，始点のうち 1 つだけがインスタンス化可能であることを意味する．AND を持ち終点を共有する複数の t-dep は，始点のうち全てがインスタンス化しなければならないことを意味する．assign_role は，始点に対応する m-group は終点のリソースに書かれた役割を持つことを意味する．assign_resource は，始点に対応する m-group は終点のリソースを持つことを意味する．adopt は commit した始点を持つ t-dep に与えられ，始点が終点により採用されたことを意味する．ラベルを持たない t-dep は follow を意味するものとする．

4.2 コミュニケーションプロセスの実行

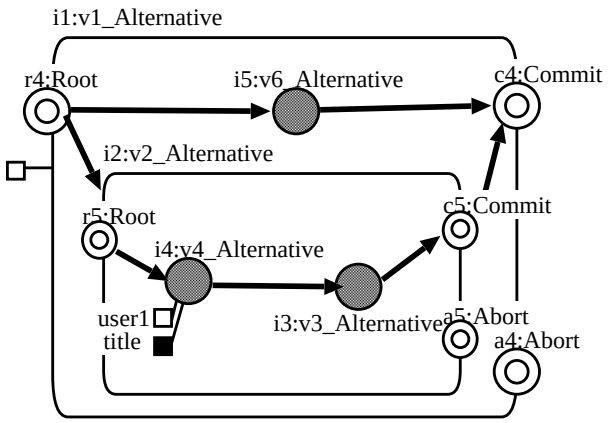

図 6: コミュニケーションプロセスの実行

ここでは，コミュニケーションプロセスの実行方法を述べる．コミュニケーションプロセスの実行中には，設計プロセスで記録された m-group と mg テンプレートはそれぞれ mg テンプレートおよび tdep テンプレートと見なされる．コミュニケーションプロセスの実行は，コミュニケーションプロセスを利用者がインスタンス化することで開始し，インスタンス化された m-group を利用者が commit することで終了する．図 6 はコミュニケーションプロセスの実行の例である．コミュニケーションプロセスのインスタンス化の際のシステムの処理を以下に述べる． インスタンス化の際には，tdep テンプレートの種類により以下の処理が発生する．

コミュニケーションプロセスの中の m-group は，設計プロセス中の対応する子グループが commit した後に commit 可能である．これは採用された解決策のみが commit 可能であることを意味する．

subgoal に対し，終点のインスタンスは始点のインスタンスの子グループになる．また，始点が終点より先にインスタンス化する．precede について，終点が始点より先にインスタンス化する．また，follow について，始点が終点より先にインスタンス化する．OR については，始点のうちいずれかのみがインスタンス化可能であり，AND については，始点全てがインスタンス化しないとコミュニケーションプロセスが commit できない．

achieve および，support，object, query に対しては，終点をインスタンス化する際に，インスタンス化しようとする利用者に対してシステムが始点を表示する．これにより利用者が m-group の目的と意見を知ることができ，設計プロセスと mg テンプレートの整合性を保つことが容易になる．assign_role に対しては，終点のリソースに指定された役割をインスタンス化された m-group の役割に指定する．assign_resource に対しては，終点のリソースをインスタンス化された m-group の リソースに指定する．

5 おわりに

協調作業支援における要求の一つとして，予期しない状況への適応があげられる．現実の世界は頻繁に変化する環境や予測困難な人間の行動にさらされる．プロセス管理技術においては，例外的な状況に対応するための動的ワークフロー [9] や適応可能ワークフローが提案されている．文献 [8] 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．文献 [3] では，プロセスの仕様を部分的に記述したものを蓄積しておき，例外時にその記述を詳細化することによって適応する方法が導入されている．さらにこの文献では，多数の作業者にコピー可能なアクティビティや，定められた期間に何度でも実行可能なアクティビティという拡張を行なっている．このように適応の形式化は議論されているが，議論の結果や組織間の合意といった，プロセスの設計の背景にある人間の意思を利用することは考慮されていない．文献 [1] では過去のプロセスを拡張したり，ECA ルールを用いて例外処理を記述することにより再利用性を促進する方法を提案しているが，コミュニケーションプロセスにおいては，あらかじめ定義される必要のある ECA ルールは用いることができない．

これまでのいくつかの構造化議論モデルが提案されている [2, 7]．これらのモデルは設計の根拠を記録するために参加者の発言を分類したものであり，設計の目標や，解決策，解決策間の比較，そして決定といった記述をもつ．これらのモデルは有効かつ一般的に適用できるものであるが，実行の順序や役割，リソース，コミュニケーションプロセス間の従属性といった，コミュニケーションプロセスを実現するために必要な情報は持っていない．

本論文では，コミュニケーションプロセスの設計履歴を示すために解決策の概念を拡張し，発言プロセス，コミュニケーションプロセス, ワークフロープロセスを統合的に扱うことを可能とした．さらにその設計履歴に沿ってコミュニケーションプロセスを生成，実行する M-Trans

システムについて述べた．現在我々は会議の運営に本システムを適応している．このシステムの有用性を評価することが今後の課題である．

参考文献

[1] D. K. W. Chiu, K. Karlapalem, and Q. Li. “E-ADOME: A framework for enacting e-services”. In *Proc. 1st Workshop on Technologies for E-Services*, Egypt, Sep 2000.

[2] J. Conklin and M. L. Begeman. “gIBIS: A hypertext tool for exploratory policy discussion”. In *ACM Trans. Office Information Systems*, Vol. 6 of *4*, pp. 303–331, October 1988.

[3] D. Georgakopoulos, H. Schuster, D. Baker, and A. Cichocki. “Managing escalation of collaboration processes in crisis mitigation situations”. In *Proc. 16th Int’l Conf. Data Engineering*, pp. 45–56, 2000.

[4] S. Inoue and M. Iwaihara. “Structured message management for group interaction”. In *Proc. Int’l Workshop. New Database Technologies for CSCW and Spatio-Temporal Data Management (NewDB’98)*, Singapore, November 1998.

[5] S. Inoue and M. Iwaihara. “Adapting transactions to exceptional situations using structured messages”. In *Proc. Int’l Symp.Database Applications in Non-Traditional Environments(DANTE’99)*, pp. 264–271, Kyoto, November 1999. IEEE Press.

[6] M. I. Kellner, et al. “Software process modeling example problem”. In *Proc. 6th Int’l Software Process Workshop*, pp. 19–29, 1990.

[7] Jintae Lee. “SIBYL: A tool for managing group decision rationale”. In *Proc. Conf. Computer-supported collaborative work*, pp. 79–92, Oct 1990.

[8] L. Liu and C. Pu. “Methodological restructuring of complex workflow activities”. In *Proc. 14th Int’l Conf. Data Engineering*, pp. 342–350, California, 1998.

[9] M. Reichert and P. Dadam. “ADEPT$_{flex}$ – supporting dynamic changes of workflows without loosing control”. *Journal of Intelligent Information Systems*, 1997.

[10] 井上創造, 岩井原瑞穂. “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”. 情報処理学会論文誌： データベース, No. SIG 8 (TOD 4) in 40, pp. 1–12, November 1999.