協調作業プロセスの構造を動的に取得することができるコミュニケーション支援システム

井上, 創造
九州大学大学院システム情報科学研究科

岩井原, 瑞穂
京都大学大学院情報学研究科

出版情報：情報処理学会研究報告. 125 (3), pp.17-24, 2001-07. 情報処理学会DBS研究会
バージョン：

協調作業プロセスの構造を動的に取得することができるコミュニケーション支援システム

井上 創造* 岩井原 瑞穂†

概要

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ないまま実行され，実行と並行して人間同士の議論によって定義が詳細化されるプロセスを管理する M-Trans システムを述べ，その方式の有効性の評価結果を示す．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いており，プロセスの例外的な，あるいは未定義の動作に対してその動作を局所的に許す方法と，過去の局所的な動作の記録に基づいてプロセスの仕様を変更する方法をコミュニケーション支援に提供することによって作業プロセスを参加者が動的に獲得できる．

Dynamic Process Specification using Communication based Processes

Sozo INOUE* Mizuho IWAIHARA†

Abstract

In this paper, we introduce a communicative process, which is a process which is executed without complete specification and designed in a discussion in parallel with the process. The execution of the communicative process may be performed on parallel with the design process, and the design process and the implementation may have interactions for coordination. Managing communicative processes is important since it realizes adaptation to unexpected situations, including exception handling and dynamic re-composition of a process in WfMS. We introduce a M-Trans System, which manages communicative processes on the model that provides integrated specification of a process and communication. The system provides communication support for localizing unexpected or undefined behavior of a process, and for dynamically capturing process specifications by generalizing the record of once localized processes.

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ないまま実行され，実行と並行して人間同士の議論によって定義が詳細化されるプロセスである．以下では，プロセスを定義するための議論を，**設計プロセス**と呼び，定義されたプロセスの仕様を**プロセス仕様**，プロセスの実体を**プロセスインスタンス**と呼ぶ．コミュニケーションプロセスは，図 1(a) のようにプロセスインスタンスと設計プロセスの間にやり取りが行われることもある．

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 [2, 5] や動的再構築 [13, 14] 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．なぜなら，予測困難な状況においては，その場で適応方法が検討され実行されるため，適応方法の検討が設計プロセスであり，適応自身がプロセスインスタンスと言えるためである．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ため，柔軟なプロセス管理が WfMS における重要な要求である．

M-Trans システムは，プロセスとその設計プロセスを統一的に扱い，参加者間のコミュニケーションの記録を元にコ

*九州大学大学院システム情報科学研究科，〒 816-8580 福岡県春日市春日公園 6-1, Graduate School of Information Science and Electrical Engineering, Kyushu University. E-mail: sozo@c.csce.kyushu-u.ac.jp

†京都大学大学院情報学研究科, 〒 606-8501 京都市左京区吉田本町, Graduate School of Informatics, Kyoto University. E-mail: iwaihara@i.kyoto-u.ac.jp

ミュニケーションプロセスを支援するシステムである． 本論文では，設計プロセスとプロセスインスタンスの間のやり取りに注目し，設計プロセスと 1 つ以上のプロセスインスタンスの間に整合性がとれなくなる場合に設計プロセスとプロセスインスタンスが協調して整合性を確保する方法を提案する．設計プロセスとプロセスインスタンスの不整合は，プロセスに例外が発生した場合や，プロセスインスタンスの実行が設計プロセスより先行する場合に起きる．提案する手法では，次の方法を用いる．

- **局所化**: プロセスインスタンスが設計プロセスに反した，あるいは未定義の動作をする場合に，その動作を局所的に許す方法である．
- **一般化**: 一旦局所化されたプロセスインスタンスをプロセス仕様に取り込み，他のプロセスインスタンスからも利用できるようにする方法である．

提案する手法では，設計プロセスとプロセスインスタンスが不整合の場合に，局所化を行なう方法と，プロセス仕様を変更して整合性を確保する方法を用意する．プロセス仕様を変更する方法は，過去に局所化されたプロセスインスタンスを設計プロセスの参加者に提示し選択させ，選択されたプロセスインスタンスをシステムが一般化することにより行なう．つまり，過去のプロセスインスタンスの記録を利用してプロセス仕様を詳細化，あるいは修正する方法を設計プロセスに実現するものである．また一般化をプロセス仕様の動的な獲得ととらえると，プロセスインスタンスの動作が 2 度以上繰り返されるまではプロセス仕様にたよらずに局所化して実行できるので，実際に繰り返される動作のみをプロセス仕様に採用することのできる効率の良い方法である．

M-Trans システムは，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モデル [6, 15, 7] に基づく．我々は文献 [6, 15] で，参加者間の議論をトランザクションと考え，参加者の発言の分類に基づきトランザクションを動的に構築する手法を提案しており，文献 [7] では，議論のトランザクションとプロセスを統合したシステムにおいて，プロセスの例外を議論のトランザクションを用いて解決する方法を示した．これらの手法における議論のトランザクションとプロセスは，それぞれ本論文における設計プロセスおよびコミュニケーションプロセスで実現できるが，本論文ではさらにコミュニケーションプロセスが設計プロセスにが先行する場合に有効な方法を提案する．

本論文の構成は次のとおりである．2章ではコミュニケーションプロセスとその設計プロセスを議論し，3章で M-Trans システムのモデルとその実行について述べる．4章ではプロセスの実行記録を用いてプロセスを動的に獲得する手法を述べる．5章ではその手法を実際の電子メールの蓄積に適用した結果を述べ，6章でまとめを述べる．

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の性質を明らかにし，技術的な課題を明らかにする．

2.1 プロセスの設計の分類

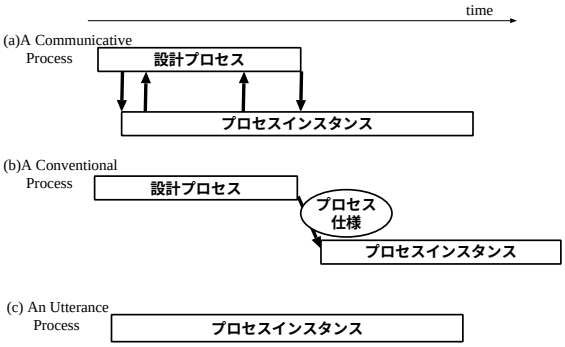


図 1: プロセスの設計の分類

一つのプロセス仕様に対しプロセスインスタンスは複数個存在してよいものとする．コミュニケーションプロセスの例を図 1(a) に示す．

ワークフロープロセス (Conventional Process) は，設計者によってあらかじめプロセス仕様が決定されて実行されるプロセスである．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がその手順をワークフローで記述することが可能である．通常の WfMS は，あらかじめ設計者がプロセスを定義していることを想定している．図 1(c) はワークフロープロセスを示している．設計プロセスはプロセスインスタンスのためのプロセス仕様を出力し，プロセスインスタンスの実行は設計プロセスが終了した後に行われる．

文献 [4, 8, 9, 1] では，発言自体により実行されるプロセスが議論されている．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．このように参加者の発言により生成され実行されるプロセスを，**発言プロセス** (Utterance Process) と呼ぶ．質問，提案といった参加者の発言の分類がシステムの支援の対象であ

る．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．発言プロセスのプロセス仕様は，システムに静的に記述されている．

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論文情報を請求し収集する．収集のためのプロセスは，それぞれの著者に応じて異なることが考えられる．ある著者は電子メールで提出するかもしれないし，別の著者は FAX で提出するかもしれない．また著者によってはタイトルと著者情報を別々に送るかもしれないし，著者によっては誤りの修正のための例外的な処理が行なわれる可能性がある．このための議論は，著者とプログラム委員の間の提案や質問といった交渉により決定される．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．

コミュニケーションプロセスの特徴は次のとおりである．

- プロセスは議論によって設計され，その中で採用されたプロセス仕様が 実行される．上記の例では，論文情報を提出する方法はプログラム委員と各著者との間の交渉により決定される．

- プロセスインスタンスの実行が設計プロセスよりも先に行なわれる可能性がある． 設計プロセスは，議論がその大半を占めるため，プロセス仕様や対象領域の知識を完全には記述する時間が無いことが考えられる．例えば，タイトルの間違いは後続の作業の遅延を防ぐために早急に修正されなければならないため，プロセス仕様が完全に定義されないまま実行される可能性がある．

- 設計プロセスは，単独で実行できずに，むしろそのプロセスインスタンスに依存する場合がある．プロセスインスタンスが設計プロセスより先行してしまう場合は，設計プロセスにおける主な議論はプロセスインスタンスの正当性や，他のプロセスインスタンスへの影響の検討になるはずである．例えば，プログラム委員が例外的な処理として，著者情報を FAX で再送してもらうように著者に要求した後には，その行為が設計プロセスで了承され，例外を通常処理に組み込むかどうかが決定される必要がある．もし組み込まれるなら，FAX で再送するという処理がプロセス仕様に組み込まれる．

- コミュニケーションプロセスは発言プロセスやワークフロープロセスと混在する．上記の例では，プログラム委員と各著者の交渉は発言プロセスにより行なわれる．またプログラムを作成する全体のプロセスは，著者情報収集後，プログラムを編集し座長を割り当てるというように，WfMS により管理されることも考えられる．

2.2 設計プロセスと実行の一貫した管理

コミュニケーションプロセスの前節に示した特徴から，設計プロセスの結果のプロセス仕様とプロセスインスタンスの間の整合性を保ちつつ，柔軟な管理を行なうことが重要である．しかし伝統的な WfMS では，プロセスは設計プロセスが終了した後に実行されるというトップダウンの方法しか保証しない．コミュニケーションプロセスを支援するためには，プロセスインスタンスから得られる情報を活用することが重要である．

本論文では，設計プロセスの中ではプロセスインスタンスから得られる情報が次のように利用されると考える．

- 局所化 (Localization): プロセス仕様とプロセスインスタンスの不整合を防ぐための単純で，強力な方法は，プロセス仕様とプロセスインスタンスを独立させ，プロセスインスタンスをその場限りの実行と見なしてしまう方法である．例えば，論文情報を再送することは，特定の著者に限った臨機応変の対応であると見なす方法である．図 2(b) は，図 2(a) のような状態から実行された実行が，プロセス仕様に対応する部分がないような場合に，局所化により整合を取る例である．

- 一般化 (Generalization): 局所化は，しばしば危険な状況をもたらす．なぜなら， 他のプロセスインスタンスで著者情報の再送がプログラム委員の承認があれば認められるというような場合に，局所化されたプロセスインスタンスが他のプロセスインスタンスの進行から孤立してしまうためである．そのため，本論文ではプロセスインスタンスの一般化を導入する．一般化は，局所化されたプロセスインスタンスをプロセス仕様に取り込み，他のプロセスインスタンスからも利用できるようにする方法である．一般化は，次のように利用される．すなわち，設計プロセスとプロセスインスタンスの 不整合が発生した時に，過去に局所化されたプロセスインスタンスのうち不整合を解決できるものを一般化し，プロセス仕様を変更して整合性を確保する．例えば，著

者情報の再送が繰り返されたときに，1 度目のプロセスインスタンスを一般化することによりプロセス仕様の変更からの孤立を避けることができる．図 2(c) は局所化されたプロセスを一般化し，他のプロセスインスタンスに適用する例である．

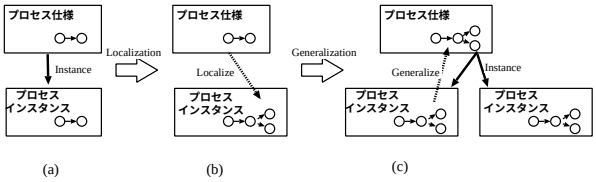


図 2: コミュニケーションプロセスの局所化と一般化

3 M-Trans システム

本節では，M-Trans システムを説明する．

現在の M-Trans システムは Java 言語で実装されており，Java applet と Java servlet が協調して動作する．利用者は Web browser 上で利用可能である．図 3は, M-Trans システムのユーザインタフェースである．システムで扱うデータは XML(eXtensible Markup Language) で記述される．XML を用いることにより，普及している XML のツールを用いてシステムのデータを流通させることが可能である．

以下では，プロセスインスタンスのモデルであるメッセージグループと，プロセス仕様のモデルである MG-テンプレートを簡単に述べる．文献 [15] に詳細を述べている．

3.1 メッセージグループ

本システムでは, メッセージやプロセスは**メッセージグループ (MG)** として統一的に扱う．図 4(b) は MG の例である．**MG** は，タプル：

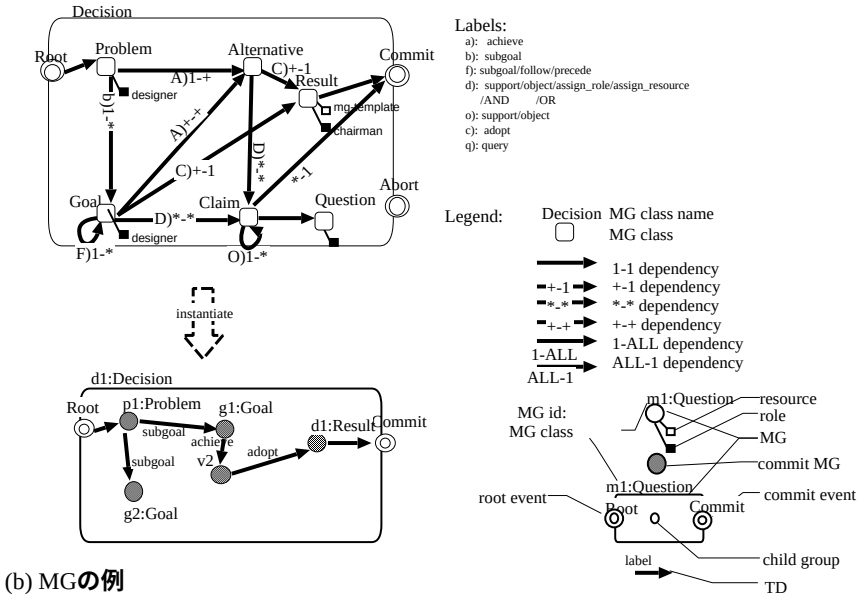
$$[m\_id, m\_class, Children, Tdeps, body, Rscs, Roles].$$

である．m_id は MG の識別子である．$body$は文章，音声，動画といった任意の媒体による参加者のメッセージである．m_class は **MG-クラス** と呼ばれ，発言者の意図や，プロセスの目的を表す．MG-クラスの例として,「Issue」「Argument」「Position」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 [10] 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．MG は, 複数の MG を $Children$ に**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．MG m が生成される事象を root, m の子グループが生成される事象を m の「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．MG 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，MG は，MG 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: M-Trans システムのユーザインタフェース

$Tdeps$ は**トランザクション 従属性 (TD)** である．図 4 では，MG 間に与えられる有向枝で示されている．TD は，MG の実行に関する MG 間の従属性で ある．MG m_1から MG m_2への TD を $label(m_1 \rightarrow m_2)$ のように表す． TD がいずれかが終了していない MG の間に割り当てられている場合，TD 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 MG が終了している場合は，TD は終点が始点の内容を参照していることを意味する．TD はあらかじめ決められた文字列の集合の要素を $label$として持つ．図中で root, commit および, abort は二重円で示されている．

(a)MG-**テンプレートの例**



(b) MG**の例**

図 4: MG と MG-テンプレートの例

3.2 MG-テンプレート

この節では，MG-テンプレートを定義する．**MG-テンプレート** は，MG の仕様を記述したものであり，次のタプルで表される．

$$[t\_id, t\_class, TChild, TDTmpls, TRscs, TRoles].$$

t_id は MG-テンプレートの識別子，t_class は MG-クラス，$TChild$ は MG-クラスの集合，$TDTmpls$ は TD-テンプレートの集合，$TRscs$ はリソースの集合，$TRoles$ は役割の集合である．これらは本節内で説明する．MG-テンプレートは，1 つの MG-クラスに対し 1 つが対応付けられる．以下では MG-クラス c を t_class にもつ MG-テンプレートを $mgTmplate(c)$ と表す．図 4(a) は MG-クラス `Decision` に対する $mgTmplate(c)$ である．この構造は SIBYL[12] から導入した．

MG-テンプレートは MG の定義と似ているが，以下の点で異なる：MG-テンプレートは子グループの代わりに MG-クラスの集合を持ち，MG-テンプレートの集合代わりに TD-テンプレートの集合を持つ．

MG-テンプレート T の定義に従い MG が生成されることを**インスタンス化**と呼び，インスタンス化された MG m を Tのインスタンスと呼ぶ．逆に Tは m の $mgTmplate$ であると呼ぶ．ある MG m のインスタンス化は，m の $mgTmplate$ 中の $TChild$ に属する MG-クラスをもつ MG を利用者あるいはシステムが生成し，m の子グループに加えることにより行なわれる． 図 4(b) 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 TD を表し，**TD-テンプレート**と呼ぶ． TD-テンプレートは

$$label(m_1[multi_1 \rightarrow multi_2]m_2)$$

で表される．$label$ はあらかじめ決められた文字列集合の要素であり，m_1と m_2は両端のオブジェクトを表す．TD-テンプレートの両端には，MG-クラス，リソース，役割が許される．$multi_1$ と $multi_2$ は，TD-テンプレートの両端に与えられる，後述する多重度制約である． 両側が MG-クラスの場合には，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．

TD-テンプレートの両端には，TD-テンプレートの属する MG-テンプレートからインスタンス化された MG の多重

度に関する制約として，1，*，+を与える．

MG m が，MG-テンプレート Tの与える制約を満たすとき，m はTに対して**整合**であると呼ぶ．本システムは，m が T に対してが整合であるかを検査し，整合でない場合はその原因となる部分を得る手続き **match**(m,T) を持つ

4 MG-テンプレートの動的な獲得

この節では，2.2節で述べた局所化と一般化の機構を用いて MG と MG-テンプレートの間の整合性を柔軟に管理する手法を述べる．本システムでは，局所化と一般化の機構を，MG-テンプレートと MG の間に実現する．

利用者が発する MG の生成や，commit，abort の要求は，システムに *MG*-操作列として伝えられる．*MG*-操作列は，MG に対する操作の列である．MG-操作列に対し，システムはその時点で MG-操作列が安全，つまり MG-操作列を適用しても全ての MG が整合であるかを検査する．もし安全なら，MG-操作列は即時に適用される．そうでないなら，システムは不整合となる原因を，MG-操作列を要求した利用者に示し，次の選択肢を与える．

- (キャンセル)MG-操作列をキャンセルし，破棄する．
- (局所化)MG-操作列を局所化として適用し，対象の MG のテンプレートへの影響を避ける．
- (一般化) 対象の MG のテンプレートを共有するすでに局所化されたインスタンスのうち，一般化することによって対象の MG が整合になるものを利用者に選択させ，一般化をする．

この方法をプロセス仕様の動的な獲得ととらえると，プロセスインスタンスの動作が 2 度以上繰り返されるまではプロセス仕様にたよらずに局所化して実行できるので，実際に繰り返される動作のみをプロセス仕様に採用することのできる効率の良い方法である．

以下では，MG-操作列の安全性の検査，不整合の原因を利用者に提示する方法，MG-操作列の局所化，局所化された MG のテンプレートへの一般化を説明する．

安全性の検査

システムは，MG-操作列の各列について，順に **match** を用いて安全性の検査を行なう．MG-操作列の各列 i の操作対象を x_iとする．MG-操作列の各列は例えば「MG m を生成する」という内容であり，このときの捜査対象は m となる．x_iの型は MG, リソース, 役割, TD のいずれかである．i 列までを適用した時点での x_iを含む MG を m_{x_i}とする．またこのときの m_{x_i}のテンプレートを $T_{m_{x_i}}$とする． MG-操作列の各列を適用した結果について，**match**(m_{x_i},$T_{m_{x_i}}$) を適用する．MG-操作列内の全ての列について **match** が空集合であれば，MG-操作列は安全である．そうでなければ，**match** の出力の和集合が不整合の原因である．

不整合の可視化

MG-操作列が安全なら，MG-操作列は即時に適用されるが，そうでなければ，システムはその原因を利用者に視覚的に表示する．図 3はその原因を表示する例である．左のフレームで，MG-操作列によって変更される部分，この場合は論文を再送するという追加処理が強調された色で表示される．

この画面は m_xの参加者が閲覧することが可能である．この画面に続いて，利用者は「キャンセル」,「局所化」，「一般化」のいずれかを選択することができる.「キャンセル」を選ぶと，システムは MG-操作列を消去して MG-操作列の適用を破棄する．残りの 2 つについては次の 2 つの節でべる．

局所化

利用者が「局所化」を選ぶと，システムは MG-操作列を m_x内での局所的な操作として適用する．x を *Local* という m_xの特別な MG-クラスからインスタンス化されたものとする．

一般化

MG の構成要素 x を MG-テンプレート Tに一般化する方法を説明する．x を MG，リソース，役割，TD のいずれかとする．一般化の手続きは次のとおりである．

1. x がリソースまたは役割なら，x をそれぞれTの$TRscs$, $TRoles$ に追加する．
2. x が MG なら，
   (a) x に対応する新しい MG-クラス c を作り，Tの $TChild$ に追加する．
   (b) Cに対応する新しい MG-テンプレート T_cを作る．
   (c) x の各子グループと T_cにこの手続きを再帰的に適用して一般化する．
3. x が TD なら，TD-テンプレート “$c_s(1 \rightarrow 1)c_d$” を T に加える．ただし c_s, c_dはそれぞれ x の始点と終点の MG-クラスである．

一般化された後の MG-テンプレート T'について，x および既存のインスタンスが整合なら，TはT'と置き換えることができる．

システムは MG-テンプレートと置き換え可能な MG-テンプレートの候補を探索し，利用者にもとの MG-テンプレートからの変化を表示する．図 3は視覚的に表示された候補の例である．図では，一般化可能な MG-テンプレートを示している．

利用者が MG の候補を一つ選択すると，システムは MG を一般化し Tと置き換える．最後に，MG-操作列が適用される．図 5は一般化の例を示している．

5　電子メール蓄積への適用結果

我々は，提案する手法をソフトウェア開発プロジェクトで流れた電子メールの蓄積に適用し，その効果を評価した．約 2 週間に流れた 130 通の電子メールを解析した．

各電子メールに対し，その内容を考慮し MG-クラス を割り当て，各電子メールを MG と想定した．今回は局所化と一般化の効果を計るため，はじめは空の MG-テンプレートのみを用意し，局所化と一般化の手法を用いて MG-テンプレートを構築していった．また，電子メールのヘッダから参照関係を導き，それを MG-テンプレートと想定した．そして MG のインスタンス化，局所化，一般化の契機となった電子メールの個数を調べた．

図 6に解析の結果を示す．図のグラフは，単位期間にやり取りされた電子メールのうちインスタンス化，局所化，一般化それぞれの契機となった電子メールが割合を示すものである．単位期間は 2 日とした．図から，MG-テンプレートなしでインスタンス化された MG が多くの割合で存在することから，局所化の必要性が分かる．一般化の割合は大部分が全体の 10%以内であり，一般化の結果生成された MG-テンプレートからのインスタンス化の割合は一般化よりも大きく，大部分が全体の 20%以上である．このことから，一般化を用いることにより一般化の約 2 倍の MG が通常のワークフロープロセスと同様の支援を受けることができることが観察できる．また，時間の進行とともに，一般化の割合はわずかながら減少する傾向にあり，インスタンス化の割合は概して増加する傾向にある．このことから，長期的なプロセスになればさらに一般化の有効性が増すと推測できる．図 5は，解析で見つかった一般化の一例である．

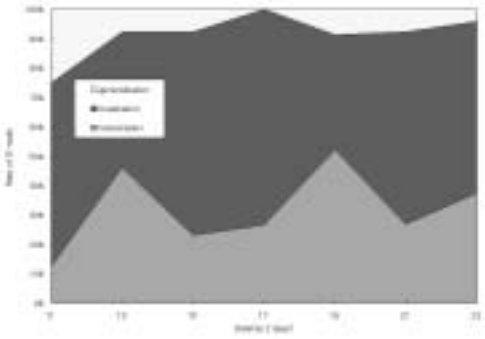


図 6: インスタンス化，局所化, 一般化の占める割合の推移

6　おわりに

M-Trans システムは，MG と MG-テンプレートを柔軟に管理することにより，コミュニケーションプロセスにおける設計プロセスとプロセスインスタンスを整合性を保ったまま協調作業の予測困難な状況に対応できるプロセス管理システムである．本論文では，局所化された MG の記録と一般化を組み合わせて MG-テンプレートと MG の整合性を保つ方法を示し，電子メールの蓄積に適用して有効性を調べた結果，プロセスインスタンスの局所化が頻繁に発生し，プロセスインスタンスの一般化用いてプロセス仕様を獲得する方法が有効に利用されることを示した．M-Trans システムは，ワークフロープロセス, 発言プロセスも扱うことが可能であるため，広い応用領域を持つことが考えられる．

参考文献

[1]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).

[2]Casati, F., Ceri, S., Parabosci, S. and Pozzi, G.: Specification and Implementation of Exceptions in Workflow Management Systems, *ACM Trans. Database Systems*, 3, Vol. 24, pp. 401–451 (1999).

[3]Conklin, J. and Begeman, M. L.: gIBIS: A Hypertext Tool for Exploratory Policy Discussion, *ACM Trans. Office Information Systems*, 4, Vol. 6, pp. 303–331 (1988).

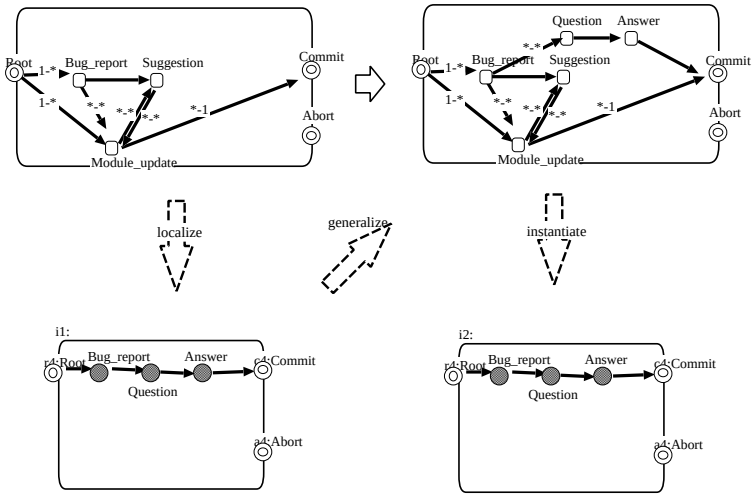


図 5: プロセスインスタンスの一般化の例

[4]Flores, F., Graves, M., Hartfield, B. and Wingrad, T.: Computer Systems and the Design of Organizational Interaction, *ACM Trans. Office Information Systems*, 2, Vol. 6, pp. 153–172 (1988).

[5]in chief, E. E., Number, V., Hagen, P. and Alonso, G.: Flexible Exception Handling in the OPERA Process Support System (1998).

[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]Inoue, S. and Iwaihara, M.: Adapting Transactions to Exceptional Situations Using Structured Messages, *Proc. Int'l Symp.Database Applications in Non-Traditional Environments(DANTE'99)*, Kyoto, IEEE Press, pp. 264–271 (1999).

[8]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).

[9]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).

[10]Kellner, M. I. et al.: Software Process Modeling Example Problem, *Proc. 6th Int'l Software Process Workshop*, pp. 19–29 (1990).

[11]Klein, M.: iDCSS: Integrating Workflow, Conflict and Rationale-based Concurrent Engineering Coordination Technologies, *CERAs* (1995).

[12]Lee, J.: SIBYL: A tool for Managing Group Decision Rationale, *Proc. Conf. Computer-supported collaborative work*, pp. 79–92 (1990).

[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]Reichert, M. and Dadam, P.: ADEPT$_{flex}$ – Supporting Dynamic Changes of Workflows Without Loosing Control, *Journal of Intelligent Information Systems* (1997).

[15]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, 40, No. SIG 8 (TOD 4), pp. 1–12 (1999).