インターワークフロー支援：組織間連携ワークフロープロセスの構築と分散型運用管理の支援機構

森田昌宏[†] 向垣内岳弥[††]
山下武史[†] 速水治夫[†]

ワークフロー管理システムは業務の流れ（ワークフロー）に従って業務の進捗や情報の管理を自動化するシステムであり，業務の効率化をもたらす企業の新たな情報基盤として幅広く利用されている．近年では，組織や企業どうしがワークフローを相互に連携させ，組織内部の業務だけでなく，企業間の取引や業務の提携などの管理のシステム化が求められている．これに対し，本研究では，このようなワークフローの相互連携を実現するため，業務の連携関係の構築という上流過程から実際の運用という下流過程までを総合的に支援する機構の提供を検討してきた．これらの支援機構を総称してインターワークフロー支援と呼んでおり，業務の相互の連携関係のモデリングの支援，連携関係のモデリング結果に基づく各組織のワークフロープロセス構築の支援，構築されたワークフローの分散環境下での運用の支援の3つからなる．本稿では，これら支援の最も重要な基盤であるモデリング言語（インターワークフロー記述言語），およびそれによるワークフロープロセスの構築の支援について述べる．

Interworkflow Management: Interorganization-cooperated Workflow Process Modeling and Management over Decentralized Heterogeneous Systems

Masahiro Morita,[†] Takeya Mukaigaito,[††] Takeshi Yamashita[†] and Haruo Hayami[†]

The workflow management system is a system that manages the enactment of business processes and related information; it is widely used to increase efficiency. In addition to managing the business within an organization, interoperability of different systems is required to integrate workflows among organizations to manage large business transactions or support cooperation among enterprises. To cope with this, we are investigating integrated management, we call it *interworkflow management*, that offers full support from the early phase of building the business relations to the working phase of the business cooperation, including the support of: modeling the relations among organizations, building workflows for each organization based on the model, and enacting the workflows in a decentralized manner. This paper proposes the modeling language *interworkflow specification language*, and describes its support of workflow realization.

1. はじめに

近年，ワークフロー管理システムと呼ばれる，業務の効率化と自動化のための支援システムが企業の新しい情報基盤として注目されている．このシステムの特徴は業務の流れ，すなわちワークフロープロセスを自動化することにある．このシステムでは，定型的な業務の手順や作業の担当者，作業に関連する業務アプリケーション等をあらかじめワークフロープロセスとして定義しておくことができる．こうすることで，システムは業務案件の処理を自動的に各作業者に割り当て，必要な情報を作業者に配送し，適切な業務アプリケーションを起動するといった制御や進捗の管理を自動的に行う．このシステムを用いることで，作業の誤りや情報の滞留等を排除し，確実かつ効率のよい業務の遂行の支援が可能になる．このようなワークフロー管理システムは現在さまざまな分野で利用されつつあり，出張伝票や購買伝票など社内の各種伝票処理など比較的小規模のものから，受注から納品までの作業の管理，

† 日本電信電話株式会社情報通信研究所
Information and Communication Systems Labs., Nippon Telegraph and Telephone Corp.
†† NTT ソフトウェア株式会社ニュービジネス事業本部
New Business Sector, NTT Software Corp.

資材の調達の管理，あるいは金融機関等の査定業務の管理など大規模な業務にも用いられている．

ワークフロー管理システムが浸透するに従い，その次の利用形態としてワークフロープロセスの相互連携が必要になるものと考えられる．現在のワークフロー管理システムの導入は，ある部門あるいは企業の内部の業務への適用に重点が置かれている．しかしその次のステップでは，それらワークフロープロセスを部門や企業の壁を越えて相互に接続し，グローバルな業務の連携や作業の分担と情報の共有への対応が求められるものと考えられる．これにより，組織の間の作業や情報の流れをも円滑にし，さらなる業務の効率化をもたらすことができる．これには，資材調達のワークフロープロセスを取引先の受注納品管理のプロセスと相互に接続することで，資材の調達をより効率化するといった応用が考えられる．

本研究では，ワークフロープロセスの相互連携を図るための一連の支援機構の実現に向けた検討を行ってきた．この一連の機構を我々はインターワークフロー支援機構と呼んでいる．この支援機構には次の3つが必要であると考えている．第1は，各部門あるいは企業間の業務の連携を分析し，それに基づいて相互のプロセスの連携関係をモデリングするための支援である．ワークフロープロセスの相互連携の関係のことを我々はインターワークフローと呼んでおり，これには各プロセスがどのような作業を分担し，どのような情報を共有するのか等が含まれる．第2は，インターワークフローのモデリングに基づいて，各部門等の業務に対応するワークフロープロセスの構築を行うための支援である．これによりプロセスの迅速かつ容易な構築を可能にする．第3は，構築されたワークフロープロセスを分散環境下で運用するための支援である．ワークフロープロセスの相互連携の運用では各プロセスは各部門等に分散したシステム下で管理されると考えられるため，ここでは各システム間で相互に制御を行うことを考える．これら3つは，業務の連携関係の構築という上流過程から実際の業務の運用という下流過程までの一連のサイクルに対応しており，本研究ではこれらを総合的に実現することを目指している．

これまでに筆者らは，インターワークフロー記述言語と呼んでいる，インターワークフローのモデリングの基盤となるモデリング言語を開発した．この言語はワークフロープロセスの相互連携の関係を記述するものであり，アクターモデルをベースとした記述体系を採用することで複雑なワークフロープロセスの相互連携を表現することができるようになっている．また，この言語によって記述した連携関係は，容易にさまざまな種類のワークフロープロセス定義言語上へ変換することが可能であると考えられ，これによりワークフロープロセスの容易で迅速な構築を支援することができると期待できる．

本稿の構成は以下のとおりである．まず2章では，先に述べたインターワークフロー支援機構として提供すべき機能についてより詳細に議論する．3章では，インターワークフロー記述言語の設計方針，概要，例題，およびワークフロープロセス定義言語への変換について述べる．4章では本研究に関連する研究について概要を述べ，最後に5章で本稿のまとめを述べる．

2. インターワークフロー支援機構

2.1 支援機構の要件

ワークフロープロセスの相互連携を図るためのインターワークフロー支援機構への要件として，以下で述べる大きく3つの支援を提供することが必要である．

インターワークフローのモデリングの支援 インターワークフロー支援機構の第1の要素として，ワークフロープロセスの相互の連携関係，すなわちインターワークフローのモデリングを行うための支援が必要である．これは，全体のワークフロープロセスの相互連携の中で各プロセスがどのような作業を分担し，どのような手順に従って作業を行うのか，あるいはプロセスが相互にどのような情報を共有もしくはやりとりするか，といった事柄をあらかじめ明確に定義するための支援である．ここでは，各組織での作業の内容や情報の流れといった業務モデルの全体を見渡し，たとえばシミュレーションなどの手法を用いてより望ましい連携形態を検討するという過程を経て，最終的にインターワークフロー定義としてこれら相互の連携関係が明確化される．

各組織におけるプロセスの構築の支援 インターワークフロー支援機構の第2の要素として，インターワークフローの定義に基づいた，各組織でのワークフロープロセスの構築を迅速化あるいは容易化するための支援が必要である．すなわち，インターワークフローに定義された各プロセスの具体的な内容，たとえばその中で処理されるべきアクティビティやアクティビティを担当する作業者の割当て，プロセスに関連するデータの定義などを決定し，それらをワークフロープロセスとしてシステム上に構築するための支援である．

分散環境におけるプロセスの運用管理の支援 インターワークフロー支援機構の第3の要素として，イン

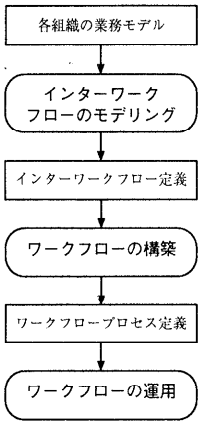


図 1 インターワークフロー支援機構の構成要素
Fig. 1 The components of the interworkflow support.

ターワークフローの定義に従って構築されたワークフロープロセスを，異種ワークフロー管理システムの相互運用によって，分散環境下で運用管理するための支援が必要である．インターワークフローの定義に従って構築されたプロセスは，各組織で独自にそれぞれのシステム上に分散して構築されている．そこで，このようなプロセスの運用時には，システムをまたがって分散して構築されたプロセスが相互にその運用管理を制御したり，情報のやりとりを行う必要がある．ここでの支援は，このような相互のプロセスの制御や情報のやりとりを実現するためのものである．

これら3つの支援は**図1**に示したように，業務の連携関係の構築という上流過程から実際の業務の運用という下流過程までの一連のサイクルに対応している．

2.2 本研究のアプローチ

本研究では，インターワークフローのモデリングおよび各組織におけるワークフロープロセスの構築の支援に対し，**図2**に示すようなアプローチを採用する．まず，インターワークフローのモデリング結果を記述するための共通のモデリング言語を用意する（これをインターワークフロー記述言語と呼んでいる）．この言語は，ワークフロープロセスの相互の連携関係を簡潔に記述できるだけでなく，計算機により容易かつ厳密に解釈が可能で，記述された内容をシミュレーションやプロセス代数的な方法によって検証を行うことが可能であるものを検討する．

次に，この言語によって記述された内容をワークフロープロセス定義のスケルトンにマッピングするための変換機能を用意する．これによって，インターワークフローに定義された各ワークフロープロセスのスケルトンが，それぞれのワークフロー管理システム上に定義される．このスケルトンは，プロセス間の作業分担の制御や情報のやりとりを実現するための定義等が記述されているものである．実際のワークフロープロセスの構築は，このスケルトンに具体的なワークフローアクティビティや作業担当者の配置などを定義することで行うことができる．このようにすることで，ワークフロープロセス定義を行う際に混入する誤りを減らすことができ，さらにWPDL[17]をはじめとする多種多様なワークフロープロセス定義言語への対応が容易になると考えられる．

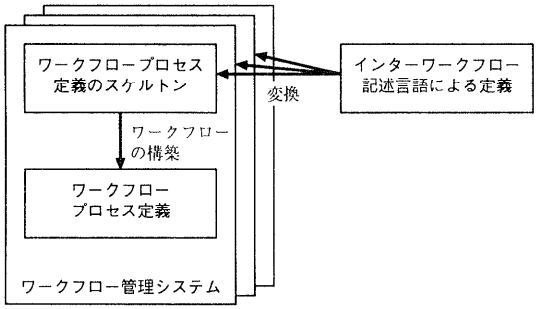


図 2 本研究のアプローチ
Fig. 2 Our approach.

3. インターワークフロー記述言語

3.1 ワークフロープロセスの連携モデル

ワークフロープロセスの相互連携の実現形態には，文献15)，16)で議論されているように，以下に示す4つのモデルが考えられる☆（**図3**参照）．

(1) 引継ぎ型
これは，あるワークフロープロセスから別のプロセスへ作業を引き継ぐことによって，あるプロセスとそれに関連する他の複数のプロセスとが連携するというモデルである．

(2) 請負い型
これは，あるワークフロープロセスが別のプロセスを呼び出し，作業の一部を代行させ，その結果を受け取ることによって，あるプロセスとその作業の一部を代行する他の複数のプロセスとが連携するというモデルである．

(3) 相互分担型
これは，あるワークフロープロセス全体のうち，各

☆ 原典ではそれぞれ，(1) connected discrete, (2) nested sub process, (3) connected indiscrete, (4) parallel synchronized と呼ばれているが，ここではあえてこのような意訳を与えた．

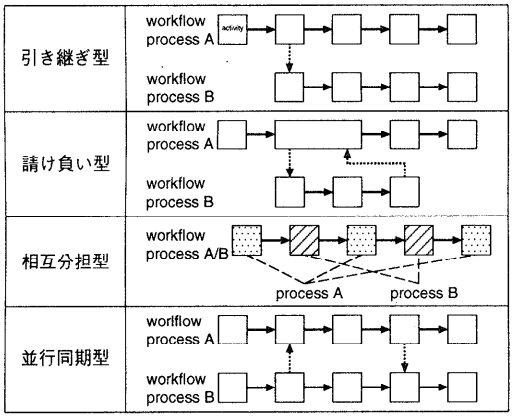


図 3 ワークフロープロセスの相互連携の形態
Fig. 3 The models of workflow process integration.

部分を別々のワークフロープロセスが相互に分担し合うことによって，複数のプロセスが対等に連携するというモデルである．

(4) 並行同期型

これは，独立なワークフロープロセスが相互に同期をとりながら，情報を交換し合うことによって，複数のプロセスが対等に連携するというモデルである．

このうち，引継ぎ型および請負い型の連携モデルは，リモートプロシージャコールの一種と考えられることから，その実現は比較的容易である．反面，モデルが単純であるために，たとえばプロセスの間で密接に情報を交換し合う必要がある場合などへの適用は困難であると考えられる．

相互分担型の連携モデルは，引継ぎ型および請負い型では困難だった連携形態にも対応できる．しかしその反面，文献 15) が指摘するように，ワークフロー管理システム間の高度な互換性の実現を必要とする．多種多様なワークフロー管理システムが混在する環境の下でこのような連携モデルに基づくワークフロープロセスの相互連携を実現するのは，非常に困難だと考えられる．

一方，並行同期型の連携モデルは，相互分担型と同様，引継ぎ型および請負い型では困難だった連携形態にも対応できる．しかも，相互分担型にくらべて実現が容易である．

このような理由により，本研究では，これら 4 つのうち並行同期型を基本としたワークフロープロセスの相互連携の実現を検討する．

3.2 インターワークフロー記述言語の設計

インターワークフローを記述する言語の記述モデルとして，ここでは次のようなモデルを採用する．このモデルでは，それぞれのワークフロープロセスは相互に独立した存在で，それぞれの運用は他から切り離されて行われると考える．さらにこれらワークフロープロセスは他のプロセスに対して，メッセージを交換することでインタラクションすると考える．このモデルは先ほど説明した並行同期型のプロセスの相互連携を表現するうえで非常に好都合である．

さらにこのモデルを採用する利点として，組織の独立性[13]への対応が容易である点があげられる．組織は本来他の組織から必要以上に干渉を受けることなく，独立に物事を決定する能力あるいは権限を持っていると考えられる．これには，作業人員の配置や担当作業の割当て，さらにはワークフロープロセスの決定などが含まれる．このような独立して物事を決定する権限を持っていることを，組織の独立性と呼んでいる．

このような独立性の一方で，インターワークフローの一部を担う各組織は，他のプロセスとの相互連携の兼ね合いからそれらとのインタラクションを厳格に規定されなければならない．このことは，組織の独立性と矛盾する要求であり，インタワークフローを構築するうえでの板挟みをもたらす．このような板挟みを解消するには，各組織のプロセスの内部を隠蔽しブラックボックスとして扱い，他との相互連携のみに着目することが必要である．同様の指摘は文献 11), 12) でもなされている．

組織の独立性はさらに，連携関係の多面性をもたらす．組織はさまざまな組織と連携関係を持っている．このとき，ある組織がどの組織と連携しているかは，組織によって見え方が異なる場合が考えられる．このことをここでは連携関係の多面性と呼んでいる．たとえば次のような資材調達の例を考えてみる．すなわち，ある企業がある資材をある商社から調達するという例で，さらにその商社が実際にはある製造元企業の代理店である場合である．この場合，調達側では，資材の購入先が代理店であるかどうかにかかわらず，資材をそこから購入するという連携を行っていることだけ意識していればよい．通常，その商社が製造元との間でどのような連携を行っているかを，調達側は知る必要はない．しかし，商社側では調達側との連携だけでなく，製造元企業との連携関係をも意識する必要がある．このことは，同じ 1 つのインターワークフローの中でありながら，当事者によって連携関係が異なって見えることになる．このような多面性の解消にも，先ほどと同様の理由によりプロセス内部の隠蔽が有効である．

そこで，インターワークフロー記述言語の記述モデルでは，プロセスの内部の処理を他のプロセスとの間のインタラクションと，内部に閉じた作業とに大きく

分けている．このうち，内部に閉じた作業は完全にブラックボックスとして扱われており，実際にその作業がどう処理されるかという定義とインタラクションの定義とを切り離している．組織の独立性とそれにともなう板挟みは，このような記述モデルの特徴により回避されている．

3.3 インターワークフロー記述言語の概要

今回我々が設計したインターワークフロー記述言語は，以下で述べるプロセス，スレッド，アクション，メッセージの4つの言語要素から構成されている．

アクション プロセスの最小構成要素である．アクションには大きく2種類ある．一方は，他のプロセスとの間で行われるべきインタラクションを表すものであり，外部アクションと呼ぶ．外部アクションには，他の特定のプロセスもしくはスレッドのインスタンスの生成，特定のインスタンスへのメッセージの送信，および特定のインスタンスからのメッセージの受信の3つがある．

- インスタンス生成アクションは，ある指定されたプロセスあるいはスレッドのインスタンスを生成させる．これにより，そのプロセス（スレッド）に対応するワークフロープロセスのインスタンスが生成され，そのプロセスを起動し処理が開始される．
- メッセージ送信アクションは，ある指定されたインスタンスに指定されたメッセージを送り出すものである．
- メッセージ受信アクションは，ある指定されたインスタンスから指定されたメッセージの着信を待ち合わせ，その内容を受け取るものである．

もう一方のアクションはプロセスの内部処理を表すものであり，内部アクションと呼ぶ．このアクションは単にある名称を持った作業を実施する必要があることだけを表現するもので，他のプロセスに対して何ら作用を与えない．このアクションは組織内部の処理をブラックボックス化するためのものである．

メッセージ 前述のメッセージ送信あるいはメッセージ受信アクションによって送受信されるデータを表すものである．メッセージにはそれぞれユニークな識別子が付与されており，これらのアクションはこの識別子を用いて送受信するべきメッセージを特定する．

メッセージは1つ以上のフィールドから構成されている．フィールドはそれぞれ識別子が付与されており，各フィールドにはメッセージ送信アクションにより対応するワークフロープロセスの内部データの値がそこへ代入されたうえで送信される．また受信アクションでは，受信したメッセージの各フィールドの値は対応するプロセスの内部データへ格納されると解釈する．

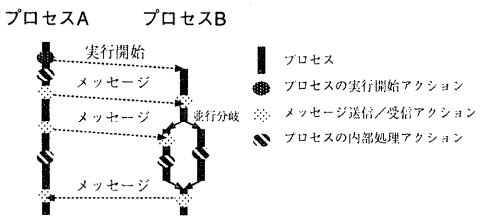


図4 プロセスの例
Fig. 4 An example of a process.

プロセス 対応するワークフロープロセスの挙動を表すものであり，実行するべき順序に従って並べられたアクションの列によって表現される．また，プロセスのインスタンスとはプロセスに定義された挙動に従って実際のアクションを実行する実体であり，実際にはワークフロープロセスのインスタンスに対応する．実行するべき順序の指定には単純なシーケンシャルな実行の他，2つ以上のブロックを並行に処理する，2つ以上のブロックのうち指定した条件によりいずれか1つのみを処理する，あるブロックを一定の条件が成立するまで繰り返す，といった制御構造を持つことができる．

たとえば**図4**に示したようなプロセスを考えてみよう．図中，縦の太線は1つのプロセスを表しており，上から下へ向かってプロセスは進行することを表している．この図で，プロセスAはBのインスタンスを生成させた後にメッセージをBとの間でやりとりし，この間に2つの内部処理を行う．プロセスBは開始後Aからメッセージを受信後，2つの処理ブロックに分岐しそれぞれを並行に処理する．一方のブロックではプロセスAからメッセージを受信し，もう一方では別の内部処理を行っている．このように表すことで，各プロセスが他のプロセスとの間で行われるインタラクションの種別，およびそれらインタラクションの間の順序規則を簡潔に表現できる．

スレッド プロセスに内包された特殊なサブプロセスであり，他のスレッドを内包することができない点を除き，プロセスと同様にして表現される．それぞれのスレッドのインスタンスは並行に処理され，プロセスとは独立に処理が行われる．

スレッドは，あるプロセスのインスタンスが他の

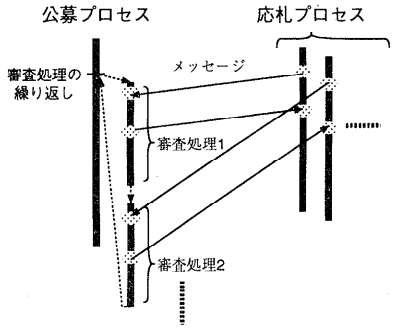


図 5 繰返しによる公募入札の審査処理
Fig. 5 A public bidding process using *iteration*.

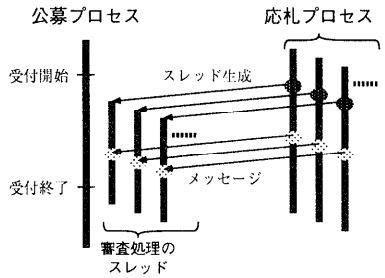


図 6 スレッドによる公募入札の審査処理
Fig. 6 A public bidding process using *thread*.

プロセスの不特定多数個のインスタンスとの間でメッセージの交換などを行う必要がある場合に用いられる．たとえば，公募入札のインターワークフローを考えてみよう．このインターワークフローでは公募を行う側のプロセスは，ある期間を区切ってその間に多数の応札側プロセスがそこへ応募を行い，公募側ではそれぞれの応札に対して審査を行う．このとき，実際に応札を行うプロセスの数は実際に公募が完了するまで不明である．また，公募側ではそれぞれの応札を並行して審査する必要がある．このような公募の扱いを，たとえば繰返し処理などの制御構造を用いて表現した場合，**図 5** のように，それぞれの審査をシーケンシャルに処理することになってしまい，1つの審査が終わるまで他の審査を処理できないなど不便である．また，それぞれの審査の途中結果を見比べ比較するといった処理を表現するのも困難である．

これをスレッドを用いて表現した場合は次のようになる．まず，入札の審査処理を公募側プロセスのスレッドとして定義する．そのうえで，公募側プロセスでは応札の受付け開始時にスレッドの受付けを宣言する．この後，応札側は公募側に対し審査処理スレッドの開始を要求する．公募側はスレッドの開始を受け付けるたびに新たな審査処理スレッドのインスタンスを生成する．公募側は応札の終了時にスレッドの受付け完了を宣言することでそれ以上の応札受付けとスレッドインスタンスの生成を停止させる．この様子を**図 6** に示す．審査処理をこのように扱うことで，それぞれの審査を並行に処理することを表現できる．また，公募側プロセスとその各スレッドは並行して処理を行うことから，審査処理の途中結果を公募側プロセスから逐一把握するという表現も可能である．

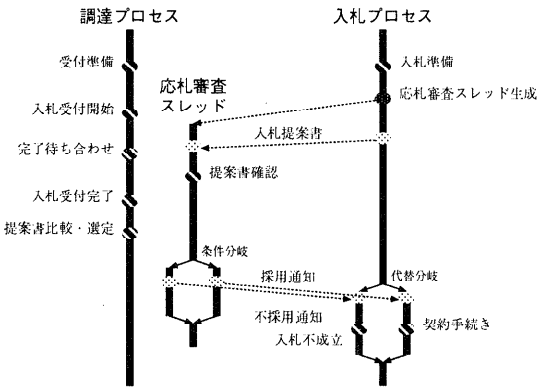


図 7 公募入札による資材調達のインターワークフローの例
Fig. 7 An example interworkflow of procurement based on public bidding.

3.4 記述形式と記述例

図 4 などではプロセスを説明のため模式的に図示したが，実際にはインターワークフロー記述言語ではプロセスをテキスト形式により表現する．プロセスの記述は大きく2つの部分に分かれており，一方をヘッダ部，もう一方をボディ部と呼ぶ．ヘッダ部にはメッセージの定義およびプロセスの宣言を，ボディ部にはプロセスの実際の挙動定義を記述する．

インターワークフロー記述言語によるプロセスの記述を例題を用いて説明する．ここで用いる例題は公募入札による資材調達のインターワークフローであり，次のようなものである．このインターワークフローには調達側と供給側プロセスの2つが含まれており，調達プロセスには応札審査スレッドが含まれている．このインターワークフローは**図 7** に示したような手順で処理を行うものとする．ここで，図中の表記は先ほどと同様である．

このインターワークフローの記述は**図 8**～**10** のようになる．このうち図 8 はこのインターワークフロー

```
1  message 入札提案書 {
2    string 入札者;
3    string 連絡先;
4    file   提案書;
5  }
6
7  message 採用通知 {
8    string 採用理由;
9    file   契約条件;
10 }
11
12 message 不採用通知 {
13   string 否決理由;
14 }
15
16 process 入札プロセス def {
17   interact 調達プロセス : INSTANCE;
18   interact 調達プロセス.応札審査 : INSTANCE;
19 }
20
21 process 調達プロセス def {
22   thread 応札審査 def {
23     interact 入札プロセス : CREATOR;
24   }
25 }
```

図 8 資材調達のインターワークフロー記述（ヘッダ部）
Fig. 8 An interworkflow description of the procurement (header part).

```
1  // 入札プロセスの挙動定義
2  process 入札プロセス body {
3    act [入札準備] retrieve 調達プロセス;
4    // プロセスの内部処理
5
6    new 調達プロセス.応札審査;
7    // 応札審査スレッドを生成
8    入札提案書 >> 調達プロセス.応札審査;
9    // 入札提案書をスレッドに送信
10
11   alt << 調達プロセス.応札審査 { // 代替分岐
12    採用通知   { // 採用通知を受け取った場合
13      act [契約手続き];
14    }
15    不採用通知 { // 不採用通知を受け取った場合
16      act [入札不成立];
17    }
18   }
19 }
```

図 9 資材調達のインターワークフロー記述（入札プロセスのボディ部）
Fig. 9 An interworkflow description of the procurement (body part of the bidding process).

のヘッダ部であり，この中で用いられるメッセージの定義とプロセスの宣言を記述している．また，図9は入札側プロセス，図10は調達側プロセスのボディ部

```
1  // 調達プロセスの挙動定義
2  process 調達プロセス body {
3    // 応札審査スレッドの挙動定義
4    thread 応札審査 body {
5      入札提案書 << 入札プロセス;
6      // 入札提案書を入札プロセスから受信
7      act [提案書確認];
8
9      sync 選定完了;
10     // 選定作業の完了を待ち合わせる
11
12     switch { // 条件分岐
13       [採用] {   // 採用の場合
14         採用通知 >> 入札プロセス;
15         // 採用通知を送信
16       }
17       [不採用] { // 不採用の場合
18         不採用通知 >> 入札プロセス;
19         // 不採用通知を送信
20       }
21     }
22   }
23
24   act [受付準備];          // プロセスの内部処理
25
26   open 応札審査;           // 入札受付開始
27   act [完了待ち合わせ];    // 受付締め切りまで待つ
28   close 応札審査;          // 入札受付完了
29
30   act [提案書比較・選定] wait 応札審査.選定完了;
31   // 応札審査スレッドと同期して選定作業を実施
32 }
```

図 10 資材調達のインターワークフロー記述（調達プロセスのボディ部）
Fig. 10 An interworkflow description of the procurement (body part of the procuring process).

を記述したものである☆．

図8のヘッダ部のメッセージの定義では，入札提案書（1〜5行），採用通知（7〜10行），不採用通知（12〜14行）の3つのメッセージが定義されている．このうち，たとえば入札提案書メッセージは文字列を値として持つフィールドが2つ（入札者と連絡先）とファイルの内容を値として持つフィールドが1つ（提案書）から構成されていることを定義している．他のメッセージについても同様である．

プロセスの宣言では，そのプロセスの名前の他，そのプロセスが内包しているスレッドの宣言，そのプロセスが他のどのプロセスとインタラクトするのかを表す連携プロセスの宣言，およびそれら連携プロセスとの間にどのような連携関係が存在しているかを宣言す

☆ なお，これらの記述中で//以降行末までの部分はコメントである．

る．プロセスとの連携関係にはCREATORとINSTANCEの2種類がある．CREATORはその連携プロセスによってこのプロセスのインスタンスが生成されることを表す．一方，INSTANCEはそれ以外の場合を表し，その連携プロセスのインスタンスをプロセスの中で新たに生成し実行を開始させる場合か，もしくはまったく別のところで生成されたプロセスのインスタンスとランデブーする場合のいずれかを意味する．たとえば，図8の調達プロセス（21～25行）では，入札プロセスが調達プロセスの連携プロセスであること，および調達プロセスには応札審査というスレッドが含まれることが宣言されている．

スレッドの宣言はプロセスの宣言とほぼ同様であり，そのスレッドの名前とそのスレッドの連携プロセスの宣言が行われる．先の図で応札審査のスレッドの例（22～24行）では，このスレッドが入札プロセスによって生成されることを表している．

各プロセスのボディ部では，プロセスの実際の挙動を定義する．たとえば**図9**では入札プロセスの挙動が定義されている．このプロセスはまずはじめに，入札準備という内部処理を行う（3行目）．この処理の中では，**retrieve**節により，すでに起動されている調達プロセスのインスタンスを検索し，この入札プロセスが入札を行うべきインスタンスを決定する．これにより独立に起動されている調達プロセスに対してランデブーが行われる．そのうえで，そのインスタンスに対して応札審査のスレッドのインスタンスを生成させる（6行目）．その後，入札提案書メッセージを送信する（8行目）．次に，代替分岐（11行目）により調達プロセスから受信したメッセージの種類別に応じて後続の作業を切り替える．ここでは採用通知メッセージを受け取った場合には契約手続きを，不採用通知メッセージの場合には入札不成立の処理を実行することになっている．

一方，**図10**では，次のような定義がなされている．この中では，4～22行で応札審査スレッドの挙動が定義されており，それ以降24～31行でこの調達プロセス自身の挙動が定義されている．このプロセスは内部処理として受付準備（24行）を行った後，入札の受付を開始するため応札審査スレッドの生成開始を宣言する（26行）．その後，応札の完了待ち合わせを行った後（27行），応札審査スレッドの生成停止を宣言する（28行）．一方，この間に入札プロセスのインスタンスにより生成された応札審査スレッドのインスタンスは，そのスレッドを生成したインスタンスから入札提案書メッセージを受信し（5行），提案書の確認を行う（7行）．調達プロセスは，生成されたすべてのスレッドインスタンスに送信された提案書を比較し，落札者を決定する（30行）．このとき，“**wait** 応札審査．選定完了”節により，すべてのスレッドのインスタンスが“**sync** 選定完了”の実行に到達したことを待ち合わせる．選定が完了すると，スレッドのインスタンスは実行を再開し，条件分岐（12行）が行われる．この条件判断は，先の30行目で行われた調達プロセスでの落札者の決定が何らかの方法でスレッドの内部状態へ反映されることで行われていると，ここでは仮定している．その後，採用の決定が行われた入札プロセスのインスタンスに対応するスレッドのみが入札通知メッセージ（14行），それ以外は不採用通知メッセージ（18行）をそれぞれ入札プロセスのインスタンスに送信する．

3.5 ワークフロープロセス定義への変換

インターワークフロー記述言語による記述は非常に抽象度が高く，この記述に基づいてそのまま実際の業務を実行することはできない．この記述には，プロセス間のインタラクションなどのプロセスの外部から観測可能な挙動しか含まれておらず，プロセスが実際に処理する内容などの具体的な記述が含まれていないためである．そこで何らかの方法により各プロセスを，それとまったく等価な挙動を示すワークフロープロセスとしてワークフロー管理システム上に構築する必要がある．

このようなワークフロープロセスの構築では，3.2節で述べた理由により，同じプロセスの記述に基づいて構築されたワークフロープロセスでも，組織ごとのさまざまな制約や規則により，その具体的な定義内容は状況に応じてさまざまなものとなりうる．そこで，インターワークフローの記述からのワークフロープロセスの構築では，ワークフロープロセス定義のスケルトンを生成させ，それをもとにワークフロープロセスの構築を行うことをここでは検討する．これにより，他のプロセスとの間でインタラクションを行うために最小限必要な定義を含んだスケルトンを自動的に生成し，それ以外の部分をワークフロープロセスの構築者が自由に定義変更できることになる．

インターワークフロー記述言語による記述からのスケルトンの生成は，次のようにして行うことができると考えられる．まず，あるインターワークフロー記述の中から生成の対象となるプロセスを指定し，そのプロセスの挙動定義をとりだす．次にその定義に含まれる全アクションのそれぞれに対応するワークフロープロセスアクティビティを生成し，ワークフロープロセ

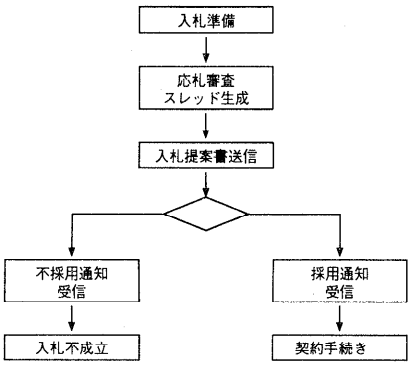


図 11 インターワークフロー記述からのワークフロープロセスへの変換例（資材調達のインターワークフローにおける入札プロセス）

Fig. 11 An example of a translated workflow definition from the interworkflow description (the procuring process in the procurement interworkflow).

ス定義に追加する．ワークフロープロセスアクティビティをプロセスの挙動定義にある制御構造に従って順次結合する．たとえば，先ほどの図 9 のプロセスは，このような置換えによって**図 11** に示したようなワークフロープロセス定義のスケルトンを生成することができる．

しかし，このようなスケルトンの生成やその運用支援の実現については，現状ではさまざまな課題が残されている．現段階ではワークフロー管理システムの互換性がきわめて低いため，さまざまなアドホックな手法をとらざるをえない．たとえば，インターワークフロー記述言語ではスレッドというフロー制御機構を導入したが，スレッドを含むプロセスをワークフロープロセスのスケルトンに変換する際に，あるシステムではプロセスとスレッドが別々のワークフロープロセスに表現され，あるシステムでは 1 つのワークフロープロセスとして表現される，という場合が考えられる．また，それぞれのワークフロー管理システムのプロセスモデルの表現能力によっては，たとえば処理の繰返しの表現が困難な場合が存在する．また，WPDL[17] などワークフロープロセスの記述言語の標準化が試みられているが，それぞれのシステムの持つプロセスモデルまでを含めて統一することはきわめて困難であり，記述内容の解釈がシステムによって異なる可能性がある．このようなシステム依存部分をスケルトン生成時に吸収し，システム固有の方法でワークフロープロセスのスケルトンを生成する効率の良い方法を検討する必要がある．

さらにこのスケルトンから定義されたワークフロープロセスを実行するには，さまざまな運用支援のためのランタイムサービスが提供されていなければならない．これにはたとえば，ワークフロープロセスインスタンス間でメッセージの送受信を行うための通信サービスや，インターワークフロー定義中のプロセスに対応するワークフロープロセスのインスタンスの生成やその際のアクセス権の管理を行うプロセス管理サービス，プロセスのスレッドインスタンスの実行を制御するためのスレッド管理サービス，独立に起動されているプロセスへのランデブーを行う際のプロセスインスタンスの検索（たとえば図 9 の 3 行目）を行うためのディレクトリサービスなどが含まれる．もちろん，これらのサービスをワークフロープロセスから利用するため，生成されたスケルトンにはこれらのサービスをアクセスするためのプログラムコードなどが適切に定義されていなければならない．これらに関して多くの部分が未検討である．

4. 関連研究

ワークフロー関連の業界によって結成された非営利団体である Workflow Management Coalition（WfMC）では，ワークフロー関連技術の標準化を行っている．この標準化活動の中で規定された Workflow Interoperability Specification[18),20)] は，複数のワークフロー管理システム上に構築されたワークフロープロセスの相互連携を実現するために必要となる，ワークフロー管理システム間のインタフェースと通信プロトコルを規定している．このインタフェースによって実現される相互連携は，3.1 節で述べた 4 つのワークフロープロセスの相互連携の実現形態のうち，引継ぎ型および請負い型のものである．この仕様には，別のシステム上のワークフロープロセスの起動，起動したプロセスの実行の制御，起動したプロセスとの間でのデータのやりとりなどのためのインタフェースが含まれている．

文献 7)~9) では，先の分類の相互分担型に基づくワークフロープロセスの相互連携の実現について研究が述べられている．この研究では G. Volker らが開発した Leu と呼ばれるペトリネットベースのワークフロー管理システム[5)]を拡張することでワークフロープロセスの相互連携を実現している．この Leu は，ペトリネットのトランジションをプロセスのアクティビティに対応付けることで，ワークフロープロセスを表現し，その実行を管理しようとするものである．このシステムの下でのワークフロープロセスの相互連携は次のようにして実現されている．まず，各プロセスの連携の全体の流れを表すグローバルビューを導入し，これによって各プロセスがどのようなアクティビティ

をどのように分担するかを記述できるよう，プロセスの表現モデルを拡張している．そのうえで，ビジネスプロセスブローカと呼ばれる機構をシステムに組み込み，これによってプロセスの連携の実行を制御させている．ただし，この研究ではホモジニアスな環境を想定しており，異種システム上に構築されたワークフロープロセスの相互連携については考慮されていない．

文献 2), 3) では，先の請負い型と相互分担型の中間的なモデルに基づいたワークフロープロセスの相互連携の実現についての研究が述べられている．この研究では，“international alliance” メタファに基づいたサミットおよび条約プロトコルと呼ばれる分散プロセスの連携モデルを提唱し，これを G.E. Kaiser らが開発した Oz と呼ばれるルールベースのプロセス実行支援システム[1]上に実装している．この international alliance とは，国家のような独立した組織どうしが協定や条約に基づいて同盟関係を結ぶことを指している．サミットプロトコルは，このような同盟関係にある組織どうしが協調して何らかの作業を実施する際に行われる手続きを模倣したものであり，独立した複数のプロセスの間で協調作業を行うためのプロトコルである．一方，条約プロトコルは，独立したプロセスの間で行われる協調作業を定義するためのプロトコルである．このような連携モデルを採用することで，自律性の高いプロセス間の連携を可能にしている．ただし，この研究でもホモジニアスな環境を想定しており，異種システム上に構築されたワークフロープロセスの相互連携については考慮されていない．

5. まとめ

本稿では，インターワークフロー支援機構すなわち部門や企業の間でワークフロープロセス間の相互連携を実現するために必要とされる一連の支援機構の検討について述べた．この中では，まずワークフローの相互連携の実現に必要とされるインターワークフロー支援機構の概要および本研究のアプローチについて述べた．そのうえで，それら支援機構のうち，インターワークフローのモデリングとワークフロープロセスの構築のための基盤として必要となるインターワークフロー記述言語について，現在までの検討結果を述べた．

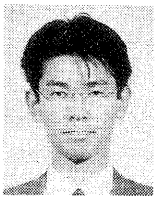
今後の課題としては次の点があげられる．まず，インターワークフロー記述言語の処理系を開発し，この言語による記述から実際にワークフロープロセス定義へ変換するコンパイラを開発することである．また，インターワークフロー記述言語で採用した記述モデルはプロセス代数などとの親和性が高いと考えられることから，それらを基盤としたインターワークフローのシミュレーションや定性的分析ツールの構築を検討する必要がある．さらに，インターワークフロー支援機構の中で指摘した分散環境下でのワークフロープロセスの運用管理の支援の実現については，いまだ検討が十分でない．WfMC の動向をふまえ，その実現方法を検討しなければならない．

参考文献

1) Ben-Shaul, I.Z.: Oz: A Decentralized Process Centered Environments, Technical Report CUCS-011-93, Department of Computer Science, Columbia University (1993).
2) Ben-Shaul, I.Z. and Kaiser, G.E.: A Paradigm for Decentralized Process Modeling and its Realization in the Oz Environment, *Proc. 16th International Conference on Software Engneering*, pp.179–188, IEEE Press (1994).
3) Ben-Shaul, I.Z. and Kaiser, G.E.: An Interoperability Model for Process-Centered Software Engineering Environments and its Implementation in Oz, Technical Report CUCS-034-95, Department of Computer Science, Columbia University (1995).
4) Department of Defense, United States of America: *Contractor Integrated Technical Information Service*, Military Standard, MIL-STD-974 (1993).
5) Dinkhoff, G., Gruhn, V., Saalmann, A. and Zielonka, M.: Business Process Modeling in the Workflow Management Environment Leu, *Proc. 13th International Conference on the Entity-Relationship Approach*, Loucopoulos, P. (Ed.), Lecture Notes in Computer Science, Vol.881, pp.46–63, Springer-Verlag (1994).
6) Georgakopoulos, D., Hornick, M. and Sheth, A.: An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure, *Distributed and Parallel Databases*, Vol.3, pp.119–153 (1995).
7) Graw, G.: Process Management In-the-Many, *Proc. 4th European Workshop on Software Process Technology*, Wilhalm, S. (Ed.), Lecture Notes in Computer Science, Vol.913, pp.163–178, Springer-Verlag (1995).
8) Graw, G. and Gruhn, V.: Distributed Modeling and Distributed Enaction of Business Processes, *Proc. 5th European Conference on Software Engineering*, Lecture Notes in Computer Science, Vol.989, pp.8–27, Springer-Verlag (1995).
9) Graw, G., Gruhn, V. and Krumm, H.: Sup-

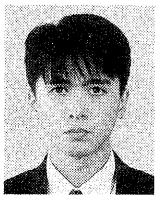
port of Cooperating and Distributed Business Processes, *Proc. 1996 International Conference on Parallel and Distributed Systems*, pp.22–31, IEEE Press (1996).
10) 石塚圭樹：オブジェクト指向プログラミング，アスキー出版局 (1988).
11) 垂水浩幸，金政ふじ，小笠原章夫：ワークフロー技術とその応用，計測と制御，Vol.34, No.12, pp.932–936 (1995).
12) 垂水浩幸，田渕 篤，吉府研次：ルールベースの電子メールによるワークフローの実現，情報処理学会論文誌，Vol.36, No.6, pp.1322–1331 (1995).
13) Veijalainen, J.: Issues in Open EDI, *Proc. Systems Integration'92*, pp.401–412 (1992).
14) White, T.E. and Fischer, L. (Eds.): *New Tools for New Times: The Workflow Paradigm*, Future Strategy (1994).
15) Workflow Management Coalition: *Workflow Product Interoperability White Paper*, Document No.WfMC-TC-1006 (1994).
16) Workflow Management Coalition: *The Workflow Reference Model*, Document No.WfMC-TC-1003 (1994).
17) Workflow Management Coalition: *Interface 1: Process Definition Interchange*, Document No.WfMC-TC-0020 (1996).
18) Workflow Management Coalition: *Interoperability Specification*, Document No.WFMC-TC-1012 (1996).
19) Workflow Management Coalition: *Terminology & Glossary*, Document No.WfMC-TC-1011 (1996).
20) Workflow Management Coalition: *Workflow Interoperability – Abstract Specification*, Document No.WFMC-TC-1012 (1996).

(平成9年3月24日受付)
(平成9年9月10日採録)



森田 昌宏（正会員）

1970年生．1994年北陸先端科学技術大学院大学情報科学研究科情報システム学専攻修士課程修了．1994年日本電信電話株式会社入社．現在，同社情報通信研究所第7プロジェクトに所属．以来，情報検索，インターネット，グループウェア，ワークフロー管理システムなどの研究に従事．共訳「GNU AWK リファレンスマニュアル」（アジソンウェスレイ，1993年）．日本ソフトウェア科学会，ACM，IEEE各会員．



向垣内岳弥（正会員）

1989年筑波大学第3学群基礎工学類卒業．1989年日本電信電話株式会社入社．共通OS，オペレーションシステム，ワークフロー管理システムなどの研究開発に従事．現在，NTTソフトウェア株式会社ニュービジネス事業本部課長代理．

山下 武史

1969年生．1991年ECCコンピュータ学院卒業．1991年日本電信電話株式会社入社．同社関西情報システムセンターを経て，現在情報通信研究所分散環境アーキテクチャ研究部に所属．電子番号案内システムの開発，ワークフロー管理システムおよびコンピュータウィルス対策の研究に従事．

速水 治夫（正会員）

1947年生．1970年名古屋大学工学部応用物理学科卒業．1972年同大学院工学研究科応用物理学専攻修士課程修了．1993年工学博士．1972年日本電信電話公社入社．現在，同社情報通信研究所分散環境アーキテクチャ研究部主幹研究員．主に，DIPSハードウェアシステム，リレーショナルデータベース処理の高速化に関する研究実用化に従事．現在，グループウェア，ワークフローシステムの研究に従事．1992～1996年本会学会誌編集委員会委員・幹事・主査．1996年より本会学会論文誌編集委員．1994年より電気通信大学大学院情報システム学研究科客員教授兼務．電子情報通信学会会員．