ソフトウェア工学 120－10
（1998．7．24）

ObjectFlowによる 分散ソフトウェア設計の図式仕様

臼井義美

奈良先端科学技術大学院大学　情報科学研究科

インターネットの発展によって分散ソフトウェアが広く利用されるようになり，オブジェクト指向技術も多くの人々に用いられるようになった．しかしながら，多くの仕様記述法が提案されているが，分散ソフトウェアの記述に十分有効な方法がないのが現状である．本報告では，分散ソフトウェアの設計を行うための，図式仕様記述について提案する．この方法によると，ネットワーク上に分散したプロセス間の動きを容易に理解することができる．

A graphical notation for distributed software design "ObjectFlow"

Yoshimi Usui

Nara Institute of Science and Technology

By the spread of the Internet,distributed software have been widely used by the user, and object technology used in the aspect of a lot of application. A lot of specification description methods are proposed to the software, in the present situation, there seems not to be a way which is still effective sufficiently in the description of distributed software.

In this paper, it proposes the effective chart specification description to the design of the software. When using this way, the movement during the process which is distributed on the network can be easily understood.

1．はじめに

複数のコンピュータ上に実現され，ネットワークを介して協調的に動作するサブシステムの集まりとして構成された情報処理システムを一般に分散システム，また，それを実現するための複数のコンピュータにまたがって実行されるソフトウェアを分散ソフトウェアと呼ぶ[1]．

今日，インターネットの爆発的な普及によって，ビジネス分野においても分散ソフトウェアの利用が期待されている．特に，ネットワーク上に分散して接続されたクライアントとサーバの関係が，ネットワーク上で動的に形成されながら１つの業務を遂行できるような分散オブジェクト指向によるシステム設計が注目されている．

分散オブジェクトは，WWWと連動してインターネット上で電子決済や電子商取引などシステムが実現し始めたことや，JavaやActiveXなどのネットワークを前提としたオブジェクト指向言語が提供されたことによって，ビジネスアプリケーションの分野でもにわかに脚光を浴びるようになった．また，オブジェクト間の通信インターフェース仕様であるCOBRA (Common Object Request Broker Architecture)を介在して，異機種間で実行されるオブジェクト同士が，相互に通信しあって処理を進めるネットワークシステムをマルチベンダー環境で構築できるようになってきた．

このように，分散オブジェクト指向によるソフトウェアの開発には，複数のプロセスが協調して1つの処理を実行するため，複雑な仕様となりがちである．また，多くの場合，システムの開発も異なる部署で同時並行的に進められる可能性がある．しかしながら，現在提案されているソフトウェア設計におけるほとんどの表記法は，分散オブジェクト開発についてはあまり考慮されているとは言えない[2]．

本報告では，ObjectFlowと呼ぶ独自の記述法を用いた分散ブジェクト指向の図式仕様を提案する．ObjectFlowは，オブジェクト指向設計に，構造化設計技法の特徴であるトップダウンアプローチや機能の部品化，プロセスの表現などを取り込んだため，ビジネス・アプリケーションにも利用しやすい[3]．ここで提案する図式仕様は，この記述法を分散ソフトウェアの設計に拡張したものである．これによると，従来多くの図式を組み合わせて表現していた内容を簡潔な少数の図式で表現できるとともに，仕様に用いる図式が広い範囲に拡散せず，紙面や画面の有効利用が可能である．また，複数の計算機に分散配置されたソフトウェア・プロセス間のメッセージ交換について，それらのトレースだけでなく，各プロセスで実行する制御フローやオブジェクトの状態遷移との関係をも的確に表現できる[4] [5] [6]．

2．では，分散ソフトウェア設計の特徴と設計の現状について考察し，3．で，提案する図式仕様の考え方を他の記述法と比較しながら説明する．4．では，この記述法を分散ソフトウェアに適用した例を示し，5．で今後の課題について述べる．

2．分散ソフトウェア設計

2.1．特徴

分散ソフトウェアは，システム資源の最適な配分が可能であり，システムのスケーラビリティに柔軟に対応できる上，部門や顧客などの組織を越えて相互接続することができるなどの特徴がある．

しかしながら，分散ソフトウェアの開発にあたっては，大規模なシステムを最適に分散化するための設計や，システムの拡張や変更に対応しやすい分散コンポーネントの設計を行うなどの難しさがある．

分散ソフトウェアを設計する場合，ソフトウェア・モジュールのコンポーネントが，現実世界のモノと対応したオブジェクトとしてモデル化することができれば，問題領域と実装すべきソフトウェア仕様との対応が非常に分かりやすくなる．このため，オブジェクトモデルは，分散ソフトウェアの設計に適しており，オブジェクト分析を行うことによって，業務上の問題を自然にモデル化できるという利点がある．

また，分散ソフトウェアの開発も同時並行的に進められることも多く，設計すべき機能を複数の開発グループに分担させる場合もある．この際，他人の行った設計が理解しやすいことが重要となる．これは，相互のプロセスが協調して処理を実行するためのシステムにおいて，開発チーム間で設計の内容を把握し，相互に矛盾がないことを確認する上で重要なポイントとなる．

このように，分散ソフトウェアの開発では，単体ソフトウェアに比較して関連要素が多く，システム全体の非同期性により再現性が得にくいことから，テストによる動作確認や不具合の原因追及が難しい．こうした困難を回避し，分散ソフトウェアの開発を容易にするための設計ツールや検証ツールあるいは簡易記述言語の整備が進められている[7]．このことから，分散ソフトウェアの設計には，分散オブジェクト指向が適しており，分散したプロセスが協調して処理を実行する様子が分かりやすい仕様記述が重要であり，ツールによる支援が不可欠であると言える．

2.2．分散ソフトウェア仕様記述の現状

現在，多くのオブジェクト指向分析，設計において，そのモデル化の手段として，さまざまな図式仕様が利用されている．このような図式仕様は，ユーザによって直感的に理解でき記述しやすいので，特に分析段階で非常に有用である[8]．

オブジェクトモデル化技法では，ユースケースがよく利用されており，それはユーザと情報システムがどのように相互作用しあうかを表現したものである[9]．ユーザは，入力イベントを発生させるオブジェクトであり，それに対応するシステムは出力イベントを発生させるオブジェクトであると捉えている．

オブジェクト指向設計の記述法としては，Shlaer&Mellor法，Coad&Yourdon法，Rumbaughの

OMT(Object Modeling Technique)，Booch法などが有名である[10] [11] [12] [13]．

また，最近注目されているものにUML（Unified Modeling Language）がある．これは，上記のBooch法，OMT法にOOSE法を統合したもので，オブジェクト指向設計の標準仕様とすべく活動している[14]．

しかし，オブジェクトモデリングツールのほとんどの表記法は，分散オブジェクト開発についてはあまり考慮されていない．例えば，Booch法の表記を用いると，プロセッサやデバイスなどの物理的なオブジェクトを表現することができ，それによって分散システムを記述することが可能である．しかしながら，並行性，並列性，および分散に対するオブジェクトモデル化技法の使い方はまだ一般に標準化されていない[2]．

一方，通信系のシステムの仕様記述には，LOTOS,Esttele,メッセージシーケンス，SDLの4つが国際標準として認められているが，SDLは1976年にCCITTにおける世界標準として勧告されて以来，数々の機能改良が行われ，広く利用されている[1]．しかし，SDLは複数のプロセス間における通信プロトコルの仕様記述には適しているが，分散オブジェクト指向などのアプリケーションの仕様記述には十分とはいえない．

3．提案する設計方法

3.1．基本的な考え方

まず，ObjectFlowの基本となっている考え方について説明する．図1に示すように本表記法では，オブジェクト間の関係を示すオブジェクト図と，各オブジェクトのプロセス間の時間的な活動を示すプロセスフロー図から構成される．

オブジェクト間の関係を示すためのオブジェクト構造図には，クラス間の概念的な包括関係を示す「is-a関係」，オブジェクト間の構造的な集約関係を表す「has-a関係」，オブジェクト間の参照，利用関係を表す「関連」を簡潔な図式で記述する．

一方，目的の処理に必要なオブジェクトの時間的な推移によるメッセージ交換の様子や状態遷移を表したのがプロセスフロー図である．この図式では，オブジェクト間のメッセージ送受信のタイミングを示すとともに，それをトリガーとしたオブジェクトの生成，消滅，活性化などの状態遷移を表現する．

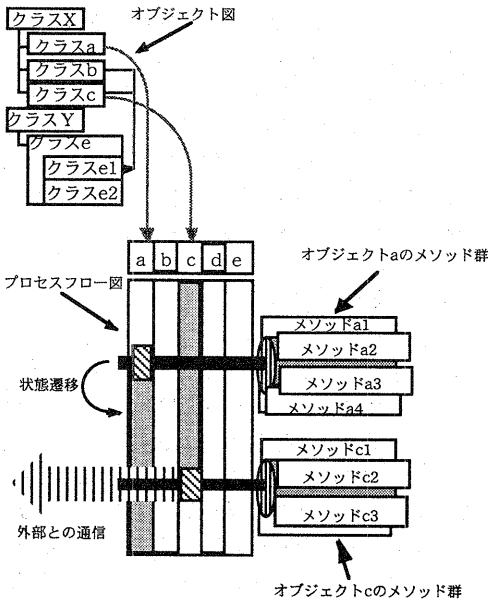

図1　ObjectFlowの表現モデル

前述のように，分散ソフトウェアの仕様を記述するためには，ネットワーク上に分散したソフトウェア・プロセスが互いにメッセージを送受信しながら処理を実行する様子を，視覚的に表現できることが望ましい．そこで，ObjectFlowでは，分散配置された複数のオブジェクト間を，ネットワークを介したメッセージが行き来する様子を図2に示すようなモデルで表現している．

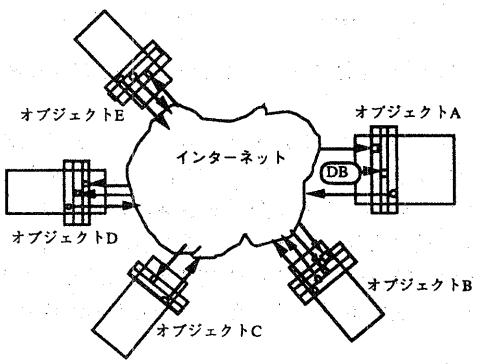

図2　ObjectFlowによる分散ソフトウェアの表現

図に示されている，オブジェクトA～Eは，各

オブジェクトのプロセスフロー図である．

以下に，ObjectFlowの図式仕様について，他のオブジェクト指向設計技法で使われる図式仕様と比較しながら説明する．はじめにオブジェクト指向設計におけるオブジェクトの関係を示す静的モデルとオブジェクトの振る舞いを表す動的モデルについて述べ，その後分散ソフトウェアに対する記述について説明する．なお，図15にUMLとObjectFlowの図式表現の比較を示す．

3.2. 静的モデル

Coadのオブジェクトモデル表記法では，オブジェクト同士の関係をオブジェクトモデルとして，また，それらのオブジェクトの関係するプロセスをシナリオビューと呼ぶ図式で表現している[15]．

この表記法では，図3に示すようにクラスを長方形で表し，クラスに属する1つまたは複数のオブジェクトを外側の長方形で表す．クラスのシンボルの中には，クラス名，属性，メソッドが記述され，オブジェクト同士の関係はそれぞれを特徴のある線で結ぶことで表す．

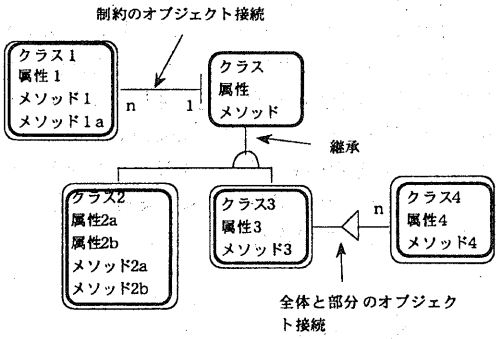

図3　Coad法によるオブジェクトモデルの表現

図3と同じ内容をObjectFlowの表記法で記述したのが図4である．クラスの継承関係を左側に，それらの制約を表す関係を右側に線を記述して表す．両者を比較すると，記述内容はほぼ同じであるが，ObjectFlowは図が単純で整理し易く，エディタで作図する場合は，画面の利用効率が良い．

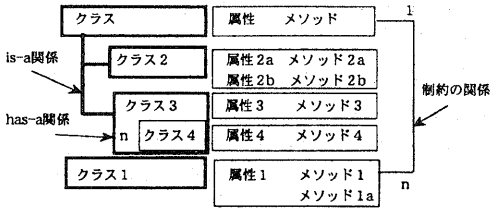

図4　ObjectFlowによるオブジェクト図

3.3. 動的モデル

Coadの方法では，それぞれのオブジェクト同士の間でメッセージ交換を行う様子をシナリオビューと呼び，図5に示すような図式で表している．これによって時間の経過によるオブジェクトの相互作用の進み工合を表している．

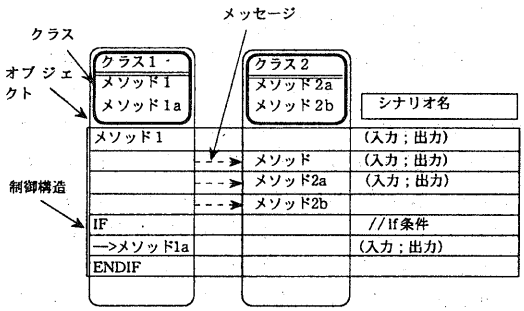

図5　Coad法によるシナリオビューの表現

Code法によるシナリオビューでは，クラスとオブジェクトを表す長方形を下方に延長し，適当なタイミングで実行すべきメソッドを記入しているので，どのタイミングでメソッドが実行されるのかが分かる．

図5の表現をObjectFlowで表したものが図6である．ここでは，あるタイミングで実行されるオブジェクトのメソッドが，図式の右側に集約され制御フローと共に記述される．従って，シナリオビューでは文字で書かれていた制御構造がプロセスフロー図では構造化チャート形式の図式表現で示される．また，図式の左側には，メッセージの送受信やメソッドの実行と関連づけてオブジェクトの状態遷移を表すことができる．ここで，横方向の有向枝はメッセージの送られる向きを示し，○印や□印は，メッセージを送受信するオブジェクトを示す．

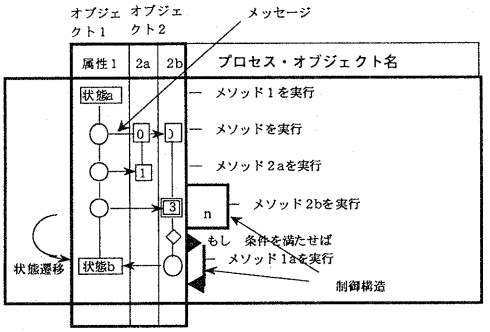

図6　ObjectFlowによるプロセスフロー図

一方，最近注目されているUMLでは，オブジェクトの振る舞いのための図式として，シーケンス図とコラボレーション図を用意している．また，オブジェクトの状態遷移を表すためにステートチャート図を，ワークフローを表現するためにアクティビティ図を用いる．コラボレーション図は，1つのユースケースやメソッド等を実現するために必要な一連のインタラクションの固まりを表現しており，オブジェクト間で受送信されるメッセージを表すが，この仕様による表現は，接続関係を詳細に表現できるものの，時間順のトレースが分かりづらい[4]．そのため，コラボレーションに参加するオブジェクトの間でやりとりされる個々のメッセージ送受信の順序を示すためにシーケンス図を併用することが多い．

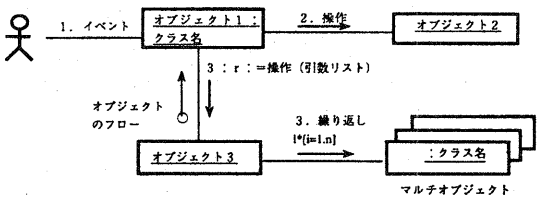

図7　UMLによるコラボレーション図

ObjectFlowでは，オブジェクト間のメッセージの送受信とメソッドの実行手順を表すシーケンス図，一連の処理の固まりを表すコラボレーション図，状態遷移を表すステートチャート図，ある業務の流れを表すアクティビティ図にあたる内容を，統合してプロセスフロー図として記述できるのが特徴である．さらに，各オブジェクトの実行に関する制御フローや，利用者や分散したオブジェクトと当該オブジェクトの内部を含めたシームレスなデータフローを記述することができる．

3.4．ライフサイクルの表現

オブジェクトにはライフサイクルがあり，何らかのプロセスによって生成された後，削除されるまでに様々な活動を行う．設計者は，これらのライフサイクルを，それぞれのオブジェクトについて定義しなければならない．

ObjectFlowでは，図8のようにそれぞれのオブジェクトに関して，内部状態の種類とその状態の変化を促すイベントを定義し，すべての変化に対して漏れや矛盾がないかを検証するのに役立つ．ここに示すオブジェクトの状態エリアは，そのオブジェクトが取り得るすべての状態を取り出し，あるメソッドの実行によって変化するすべての状態遷移を関連づけて示すことができる．

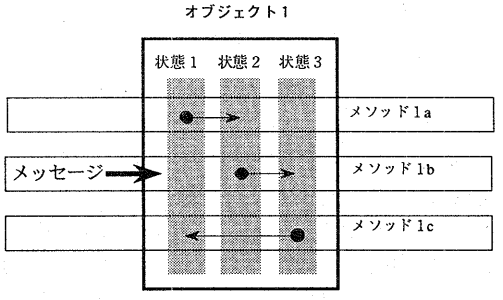

図8　ObjectFlowによるライフサイクルの表現

プロセスフローを記述する場合は，先に定義した状態遷移に基づいて，イベントによってそのオブジェクトの内部状態が変化することを示す．そのため，，あるオブジェクトの状態遷移とメッセージの交信，メソッドの実行との関係をより明確に表示することができるという特徴がある．

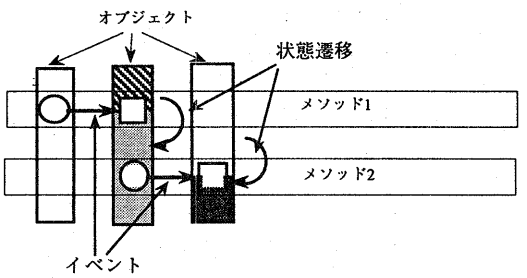

図9　ObjectFlowによる状態遷移の表現

3.5. オブジェクトの協調動作

UMLでは，分散ソフトウェアにおける複数のオブジェクトの協調動作を記述するために，アクティビティにレーンを設定してその実行責任主体を明示する．アクティビティ間での成果物のやりとりをオブジェクトフローと呼び，図10のように表現しており，これはDFDの代用にもなる[16]．

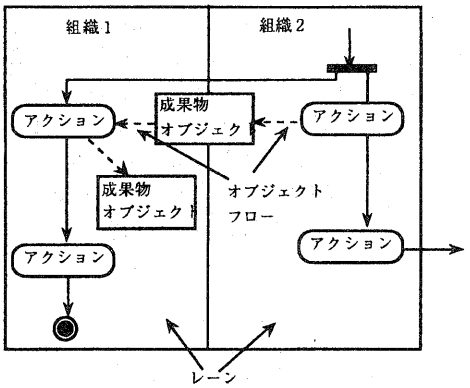

図10　UMLによるレーンの記述例

ObjectFlowでは，複数のプロセスが同時に実行される場合は，図11のように複数のプロセスフロー図を組み合わせて，相互のメッセージ交換を示す．従って，複数の物理的な実行責任主体である組織や計算機を，時間軸を共有する複数のプロセスフロー図に割付け，それぞれの図の間でメッセージの交換やオブジェクトのやりとりを記述する．このように，複数の組織間を送受信するメッセージやデータの移動関係を記述することにより，分散したプロセス間のデータフローを，それぞれのプロセスの制御フローと関連づけながら表現できるのも特徴の1つである．

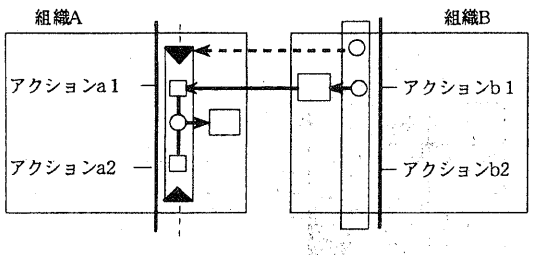

図11　ObjectFlowによる分散ソフトウェアの記述モデル

3.6. ビジネスオブジェクト

ビジネス・アプリケーションにおいて，顧客，注文，お金，支払いなどのアプリケーションに依存しない概念を記述するための中立的な方法を提供するものに，ビジネスオブジェクトがある．オブジェクト技術およびコンポーネントの目指す究極にあるのが，このような中規模で，より実世界のものに近い振る舞いをするコンポーネントを提供することである．[17]

ビジネスオブジェクトは，オブジェクトをある機能や振る舞いを表すものととらえ，外に対しては必要最小限のインターフェースで，オブジェクトの機能を規定する．こうすると，振る舞いの方法は変化しても，影響の範囲が限定されオブジェクトの独立性を保つことができる．

ObjectFlowは，構造化設計における機能分割や段階的詳細化の扱いが容易であるため，特定の機能を表すプロセスフローとそれを実現する複数のオブジェクトを1括りにしてコンポーネント化するのに適している．従って，ビジネスオブジェクトに対応したオブジェクト群を，コンポーネントとして扱うのに適当な大きさの粒度で構築することができる．

4. ObjectFlow の適用事例

図12にクライアントのWWWブラウザからHTTP(Hypertext Transfer Protcol)を介して，WWWサーバのデータベースをアクセスする簡単なアプリケーションの例を示す．この例では，サーバはCGI(Common Gateway Interface)を起動してデータベースのアクセスを行っている．

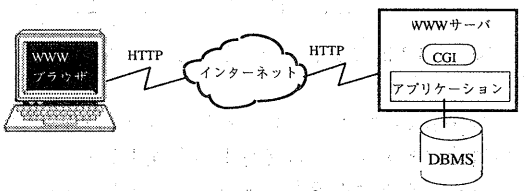

図12　WWWベースのアプリケーション例

図13にこのアプリケーションのプロセスフロー図を示す．この図から，CGIプログラムがWWWサーバより起動され，環境変数を読み込んだ後，標準入力パイプ経由でメッセージを受け取り，データベースをアクセスし，標準出力パイプ経由で

WWWサーバに結果を返すという処理の流れが容易に読みとれる．また，サーバ／クライアント間，アプリケーション間，データベースにまたがるデータフローがシームレスに表現される．

図13では，データベースをアクセスするCGIの部分を１つの機能を表すコンポーネントとして扱っていることを示している．

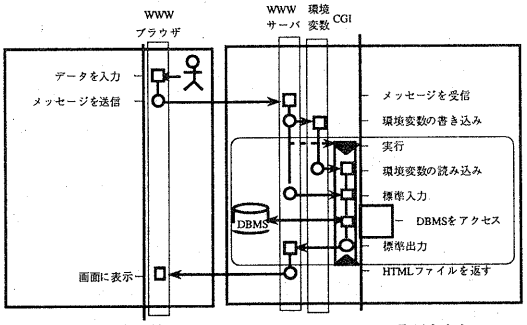

図13　簡単なアプリケーションの記述例

図14に，やや複雑なアプリケーションの記述例を示す．この例では，利用者が場エージェントと呼ばれるアプリケーションに対してコミュニケーションを行い，その指示によってネットワークで結ばれた別々の計算機に配置されたデータ・エージェントおよびプロセス・エージェントと呼ばれるアプリケーションとメッセージ交換を行っている様子を示している．

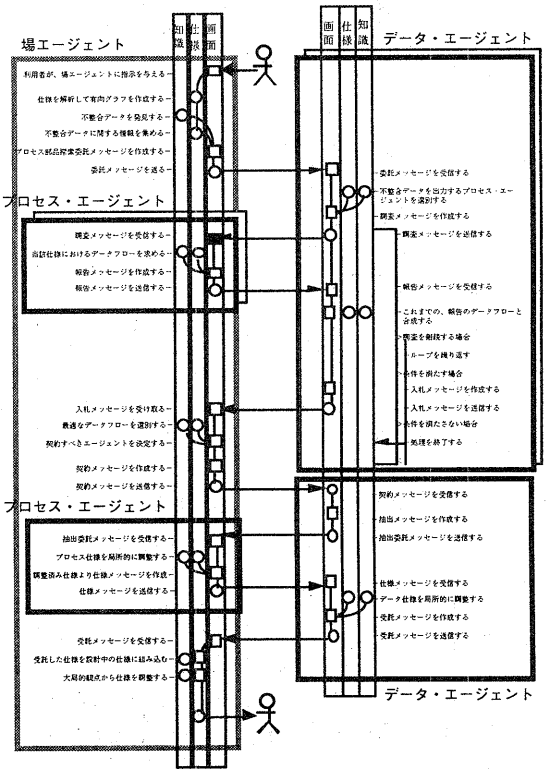

図14　ObjectFlowの適用例

5．　おわりに

本報告では，多くの計算機に分散したソフトウェアが互いに協調しながら業務を実行するような分散ソフトウェアの図式仕様について提案した．

この記述法によると，分散ソフトウェアの設計の課題であったメッセージの送受信による複雑な処理のタイミングを分かりやすく表現することができた．特に，分散するソフトウェアが協調して実現する処理の内容を，データフロー，制御フロー，状態遷移を相互に関連づけて１つの図式で表現できることが特徴である．また，いくつかのオブジェクトの集合が１つの機能を果たすようなコンポーネントの設計にも適していることを示した．このような特徴は，今後増大するであろう分散ビジネス・アプリケーションの開発に有効であると考えている．

しかしながら，今後の課題として，以下のような点が挙げられる．

まず，設計や検証のための支援ツールを開発する必要がある．通常のソフトウェア開発を目的としたObjectFlowの設計支援ツールは既に試作が完了しているので，分散ソフトウェアの設計に対応させるのはそれほど困難ではないと考えている．また，本仕様記述を実際の大規模分散ソフトウェアに適用し，その効果を評価することも重要な課題である．最後に，分散環境を利用して，別の組織に所属する設計者同士が同時に設計・開発を行う分散設計，分散開発に対応できるような方法についても研究したい．

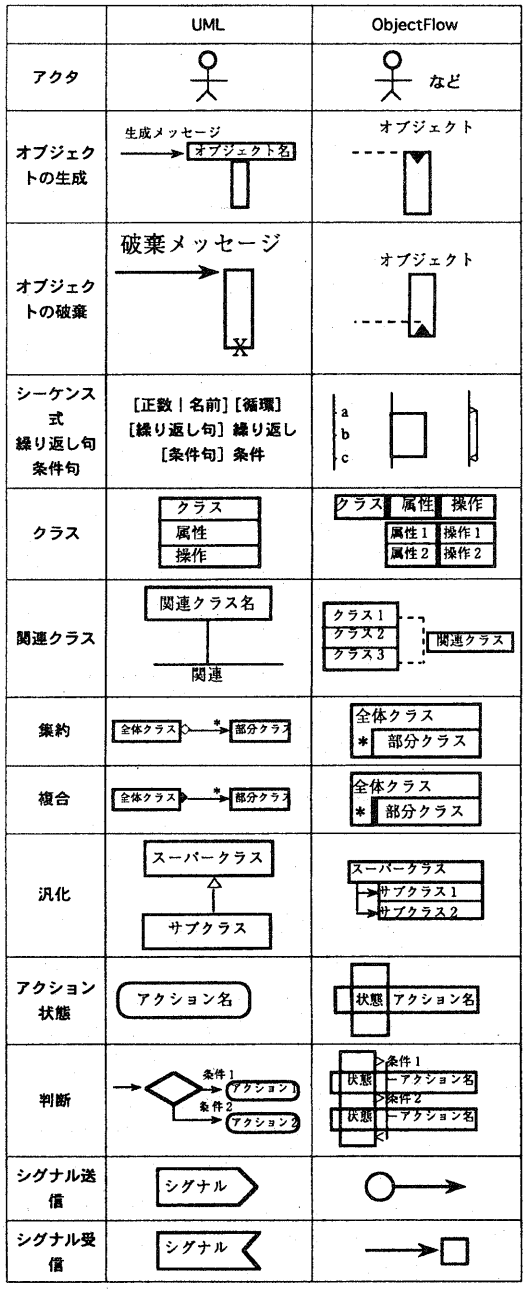

図15　UMLとObjectFlowの記述法の比較

参考文献

[1] 長野宏宣，宮地利雄：分散ソフトウェア開発，共立出版(1996).

[2] Timothey W.Ryan:*Distributed Object Technology*, Prentice Hall(1997).

[3] 臼井義美：オブジェクト指向設計と構造化設計を組み合わせた設計支援ツール，第15回ソフトウェア生産における品質管理シンポジウム論文集，pp.233-240(1994).

[4] 臼井義美：SP-FLOWによるデータ構造に基づくシステム設計法，情報処理学会誌，Vol.25，No.11，PP.1228-1236(1984).

[5] 臼井義美：Show-CASEにおける仕様部品の再利用支援環境，情報処理学会ソフトウェア再利用技術シンポジウム論文集(1992).

[6] 臼井義美：データフローによる部品探索機能を持つ仕様エディタの試作，情報処理学会ソフトウェア工学研究会報告，Vol.94,No.55,pp.89-96(1994).

[7] 長野宏宣，宮地利雄：分散ソフトウェア開発，共立出版(1996).

[8] 本位田真一，山城明宏：オブジェクト指向システム開発，日経BP出版センター(1993).

[9] Ivar Jacobson:*Object-oriented software engineering A use case driven approach*, ACM(1992).

[10] S.Shlaer,S.J.Mellor: *Object-oriented systems analysis*, Prentice-Hall(1988).

[11] P.Coad,E.Yourdon:*Object-oriented design*,Yourdon Press(1991).

[12] J.Rumbaugh et al.:*Object-oriented modeling and design*, Prentice-Hall(1990).

[13] G.Booch:*Object-oriented design with applications* (1990).

[14] 今野睦監修：オブジェクトモデリング表記法ガイド，プレンティスホール(1998).

[15] Peter Coad:*Java design*, Prentice Hall(1997).

[16] 羽生田栄一：統一モデリング言語UML1.1の概要，Dr.Dobb's JOUNAL JAPAN，翔泳社Vol.3(1998) .

[17] RobertOrfali,Dan Harkey:*Cliant/server programming with Java and CORBA*, John Wiley & Sons(1997).