データベースシステム 113－11
（1997. 7. 14）

動的分散システムを支援するオブジェクトアダプタの設計

山崎 顕治　　都司 達夫　　宝珍 輝尚
福井大学工学部

本研究では、より柔軟な分散システムの構築・運用を目的として、動的再構成の可能な分散オブジェクト指向プログラミングシステム「DROPS」(Dynamically Reconfigurable Object-oriented Programming System) の設計、開発を行っている。DROPS のプログラミングモデルは、現状の分散システムの動的再構成を困難にしている要因であると思われるソフトウェアモジュールの独立度の低さを改善し、実行時にもそのソフトウェア構造を保つよう設計された。このことにより、設計時と同レベルまできめ細かい再構成が可能となり、かついくつかのメンテナンス操作を部分的に隠蔽したままで実行時に動的に行うことも可能である。本稿では、DROPS のプログラミングモデルの概要について述べるとともに、DROPS で提供される動的再構成の主な要素である自己置換や複合オブジェクト操作などを実現するオブジェクトアダプタの機能について述べる。

Design of Object Adapters for Supporting Dynamically Reconfigurable Distributed Systems

Kenji YAMASAKI　　Tatsuo TSUJI　　Teruhisa HOCHIN
Department of Information Science, Fukui University

In this research, aiming at the flexibly configurable and manageable distributed systems, a distributed object-oriented programming system DROPS (Dynamically Reconfigurable Object-oriented Programming System) has been designed and developed. The programming model of DROPS was designed to preserve the logical structure of software modules even at runtime and admit the runtime reconfiguration of the structure according to the dynamic environment of the objects. This runtime reconfiguration is accomplished by improving the poor independency of software modules, which makes the existing distributed systems hard to be reconfigurable. The preservation makes it possible to reconfigure the structure at the same level of granularity as the design stage, and to perform the reconfiguration dynamically at runtime with the maintenace activities being hidden. In this paper, the outline of the DROPS programming model and the feartures of the *object adapters* are described. The object adapters realize the mechanism of self replacements of objects or various operations on complex objects, which are the major elements of the dynamically reconfigurable features that DROPS provides.

1　はじめに

ネットワーク環境にはネットワーク環境特有の性質があるので、分散環境において実用的な (すなわち非分散システムと同じくらいに高性能・高品質な) システムを構築することは一般に困難である。

分散システム研究で常に問題になる点として、1) 実用的なパフォーマンスが得られない、2) 大規模分散 (複数の管理単位が混在するケースなど) への対策が必要、3) 動的に変化するリソース (ハードウェア、プロセス、オブジェクト) への対策が必要 などが挙げられる。またさらに、4) 1〜3 を解決するためのサポート技術が開発レベル (プログラミングレベル) で十分に与えられない ということも、重要な問題のひとつとして挙げられるだろう。

これらの問題に目をつぶるならば、最も単純に分散システムを構築する方法は、システムに関与するすべてのハードウェアを固定的なものと見なし、システムの各機能を分散配置して、全体でひとつの仮想的計算機と見なして構築することである。しか

し、ネットワーク環境特有の性質をうまく活かし、積極的に利用することができるならば、分散システムは非分散システム以上に実用的なシステムになり得るはずである。我々はネットワークの性質から生まれるこれらの分散システムの問題を考慮し、始めから分散システムを想定したオブジェクトモデルと、それを用いたプログラミングモデルやランタイムモデルを設計する方針を立て、積極的にネットワーク環境の性質を利用することのできる分散オブジェクト指向プログラミングシステムの作成を目指して、日々検討を重ねている。

我々は、先述の問題の2)と3)、とりわけ3)の動的に変化するリソースへの対応を中心とした研究を行っている。具体的には、「動的関連モデル」と「関連表モデル」という2つのモデルを提案し、分散システムの動的再構成を実現することで、柔軟な実行時環境への対応を実現し、管理単位によるシステム分割ならびにアクセス制限を実現することを提案した[1]。また、これらのモデルを用いた最初の分散オブジェクト指向プログラミングシステム「DROPS」(Level1)を設計し、簡単なプロトタイプシステムとして実装した[2]。しかし、これらのモデルは非常に原始的であるため、実際問題として問題解決を行うためには、まだまだより多くの工夫が必要である。そこで本研究では、これら2つのモデルにいくつかの機能拡張を施し、より具体的なDROPSプログラミングモデルの定義を行った。本稿では、その中から特に動的再構成のためのオブジェクトアダプタ周辺の機能設計について述べる。

2 オブジェクトモデル

動的再構成を実現するためには、オブジェクト間の機能的独立度は高いほど好ましい。しかしそれ以前に、オブジェクト間の境界がはっきり示せなければ、システム構造自体が複雑になり、事実上再構成の余地がなくなってしまう可能性もある。DROPSプログラミングモデルのオブジェクトモデルではこの対策として、1)関連の分類、2)動的関連による接続、3)複合オブジェクトモデルの3つを挙げる。

2.1 「関連」の分類

DROPSプログラミングモデルにおけるオブジェクトは、(おそらく必ず存在するであろう)他のオブジェクトとの関係を明確にして個々の役割を見定め、その機能境界を明らかにして設計されなければならない。

オブジェクト設計者は、メソッドの実装やデータ属性に用いる「他のオブジェクトの参照」(以降、関連と呼ぶ)を、「仕様として公開すべき意味的な関連」とするか、「隠蔽すべき実装上の関連」とするかを選択しなければならない。そして後者を選択した場合にはさらに、参照先オブジェクト(に相当するもの[1])を、「設計時から自分のコードに静的に埋め込む」か、あるいは「実行時に動的に割り当てられるのを期待する」かの選択を行う。DROPSプログラミングモデルでは、最初の選択における前者を「意味依存外部関連」、二番目の選択における前者を「内部関連」、後者を「実装依存外部関連」と称して、各々区別する。

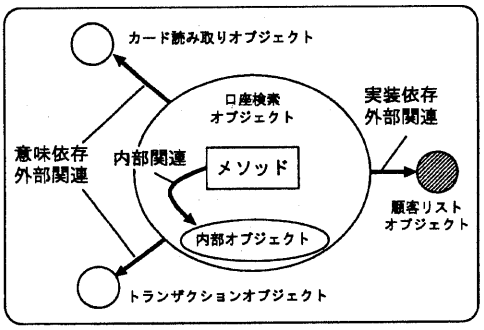

図1: 3種類の関連

例として、カード読み取りオブジェクトから顧客のナンバーを受け取り、内部に持つ顧客リストから口座情報を検索してトランザクションオブジェクトに渡す処理を行う口座検索オブジェクトの場合を考える(図1)。カード読み取りオブジェクトとトランザクションオブジェクトはいずれも、口座検索オブジェクトの内部に隠蔽されるような設計にはならなかったとする。この場合、両オブジェクトと口座検索オブジェクトの間で築かれる関連は、意味依存外部関連である。また、口座検索オブジェクトが顧客リストをどのような内部表現で管理するかは口座検索オブジェクトの実装上の問題であり、他のオブジェクトの感知するところではない。したがって、顧客リストオブジェクトは口座検索オブジェクトの内部に隠蔽されるべきであり、特に実行時割り当てにする理由もないので、これは内部関連にするのが

[1] どのような実行時表現になるかは、オブジェクトの実装方法によって異なる。

妥当である。実装依存外部関連は、実装上使用するオブジェクトに何らかの制限(必ず特定のマシン上に存在しなければならない、など)によって静的に組み込めない時などでの使用が想定される。

2.2 動的関連

オブジェクト間の関係は、「オブジェクト参照」によって実現される。オブジェクト参照は見かけ上は一方向であるが、実際には参照元・参照先両オブジェクトへのポインタを組として持つ双方向の「動的関連」によって実現される[2]。DROPSプログラミングモデルではこの動的関連を実行時に修正することが可能であり、基本的にこの操作によって動的再構成を実現する。また参照先オブジェクトが参照元オブジェクトを逆探知し、能動的にアクションを起こせる機構も提供する。これにより、参照先側の都合によって再構成が行われた場合にも、参照元オブジェクトへは隠蔽したまま、関連の一貫性を保証することが可能となる。

2.3 複合オブジェクト

DROPSプログラミングモデルでは、動的関連を直接に修正することよりもより高度な動的再構成の手段として、複数のオブジェクトを有機的に組み合わせて1つの「複合オブジェクト」を作成する方法を提供している。複合オブジェクトはまた、システムをオブジェクトより大きな機能単位に分割し、複合オブジェクト内部の再構成を隠蔽するという意味で、従来の関連表[1][2]に相当する概念となる。

複合オブジェクトは、外見上は通常のオブジェクトと変わりはなく、クライアントは両者とも同等に扱うことができる。オブジェクトは、メソッドやデータなどの公開属性の一覧を定義する「インターフェイス」と、インターフェイスの実際の実現解である「実装」、そして2.1節で述べた「意味依存外部関連の集合」と「実装依存外部関連の集合」の4つのパートから構成される(図2)。

複合オブジェクトの場合も同様だが、複合オブジェクト操作(後述)によって、インターフェイスを複数個持つことができる点が異なる。複合オブジェクトの内部構造や操作方法については後述する。

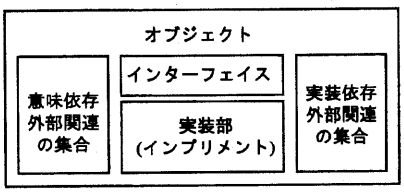

図2: オブジェクトの構造

3 操作モデル

DROPSプログラミングモデルの操作モデルには、オブジェクトの生成、保持(永続化)、削除、評価(同期/非同期/一方向)、移動(能動的/受動的)、複写、検索といった通常の操作モデルの他に、特徴的な操作モデルとして「自己置換」、「バインド(関係付け)」、「複合オブジェクト操作」が加わる。

3.1 オブジェクトアダプタの役割

先述の操作モデルのうち、オブジェクトアダプタはオブジェクトの検索を基本として、オブジェクトの生成や入手、自己置換など、オブジェクト間の関連に関する処理全般を担当する。一般的なオブジェクトアダプタ(例えばCORBA[3]のBOAなど)との違いは、オブジェクト間の関連に対する関心度である。従来のオブジェクトアダプタでは、再構成に関する方針や接続方針などの自由度が低く、複雑な状況下における柔軟な再構成やオブジェクト間接続を確立することは難しい。本モデルにおけるオブジェクトアダプタでは、オブジェクト間の関連の構築に際し、後述するオブジェクト側の様々なポリシーを受け付ける幾つかの機構を設けている。また、オブジェクト側が能動的に関連に対する主導権を握ることも可能である(ただし、自分に関する関連のみ)。

3.2 自己置換

自己置換は、自分への関連参照を他のオブジェクトへ切替える操作である。例えば、AがBを参照し(A→B)、BがCを参照している(B→C)場合には、BはAからの参照をCへの参照に切替える(A→C)ことができる。これにより、クライアントに何ら通知することなく、サーバ側の都合に合わせた応対を実現することができる。

3.3 バインド (関係付け)

ここで言う「バインド」は、自オブジェクトから参照される他のオブジェクトが関係付けられる (動的関連が生成される) 操作ではなく、自オブジェクトに対して、他のオブジェクトからの参照が関係付けられる操作のことを示す。このような時、オブジェクトアダプタは、参照先オブジェクトに対し、接続元オブジェクトが作成した動的関連へのポインタと接続対象となった接続先オブジェクトのインターフェイス情報を引数にして、接続先オブジェクトに対しバインドメッセージを発行する。メッセージを受けたオブジェクトは、相手オブジェクトへの動的関連と接続対象のインターフェイスを知るので、インターフェイスや相手オブジェクトの種類によっては、自己置換などを用いて関連にフィルタリングを施したり、別のオブジェクトへ関連をリダイレクトしたりすることを、相手オブジェクトに隠蔽したまま、自発的に行うことが可能となる。

3.4 複合オブジェクト操作

複合オブジェクト操作によって、自由度は高いが規則性のなかった従来の動的関連 [1][2] による動的再構成に比べ、ある程度規則だった動的再構成を実現することができる。オブジェクトを組み合わせて複合オブジェクトを作成する方法に係わる操作 (複合オブジェクト操作) には「包含」と「集成」があり、それを拡張する機能として、「多重定義」を定義する。これらの操作は、オブジェクトアダプタのサポートによってすべて実行時に動的に行うことが可能である (図 3)。

3.4.1 コンテナオブジェクト

複合オブジェクトを生成するためには、あるオブジェクトを「コンテナ」に設定し、しかるのちに、包含や集成を用いて他のオブジェクトを追加・合成する。複合オブジェクトのベースとなるこのオブジェクトのことを、「コンテナオブジェクト」と称する。

3.4.2 包含

包含操作は、あるオブジェクトをコンテナオブジェクトの内部に登録する操作である。登録されたオブジェクトは、そのコンテナオブジェクト内に登録されている他のオブジェクトと動的関連を構築す

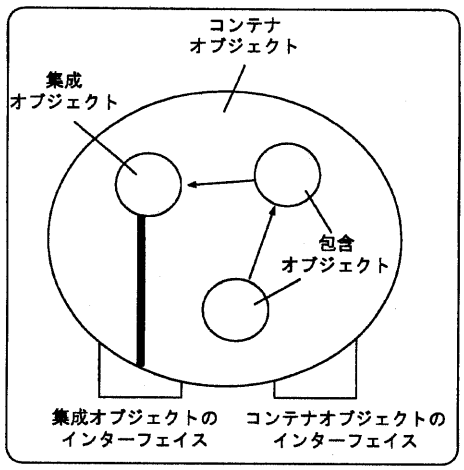

図 3: 複合オブジェクト操作

ることができる。単に包含されただけのオブジェクトは、そのインターフェイスはコンテナオブジェクト外部には公開されない (意味依存／実装依存外部関連の現状は維持される)。包含されたオブジェクトには、コンテナオブジェクト外部に対する公開／非公開のアクセス属性が新たに設定される。アクセス属性が非公開である包含オブジェクトは、コンテナオブジェクトの外部から直接的に関連を受け取ることはできない。なお、現段階では既存のオブジェクトをコンテナオブジェクト内に包含させる (すなわち内部状態を保ったままコンテナ内に移動させる) ことはできない。

3.4.3 集成

集成操作は、包含操作にともない、包含したオブジェクトのインターフェイスを、コンテナオブジェクトのそれに並列に追加する操作である。集成されたインターフェイスは、コンテナオブジェクト外部からは、それぞれ明確に区別される。

3.4.4 多重定義

包含されたオブジェクトは、その後集成／集成解除操作を適用することも可能である。インターフェイスの等しい幾つかのオブジェクトを包含しておいて、常にそのいずれか一つを切替えて集成する (すなわちインターフェイスを公開する) ようにコンテナオブジェクトを設計すると、外部から見て多重定

義をエミュレートすることができる。

4 応用例；オブジェクトの提供に関する実行時解決

分散システムは、いたる場面で要求を実行時に解決できる能力が必要とされる。

現在の分散システムは、閉じられたLANの中で構築されることが多いが、セキュリティの保証が進められた将来には、LANを越えてのネットワーク的交流、すなわち複数の管理単位(ドメイン)が混在してひとつのシステムを構築する、階層型の分散システムが主流になるものと思われる。そしてこのような時には、分散システムの実行時解決能力はより柔軟かつ強力なものでなければならない。

ここでは例として、「オブジェクトの提供問題」について考える。これは、「クライアントが接続を希望するオブジェクトを準備し提供するサーバ」に関する問題である。クライアントは、オブジェクトの条件としてインターフェイス(あるいはその集合)を示すので、サーバは指定されたインターフェイスを有するオブジェクトを準備し、クライアントに返答しなければならない。

クライアントが指定したインターフェイスに該当するオブジェクトがサーバの管理下に存在しなかった場合、サーバは自己置換を用いて、別のドメインのサーバにその接続をリダイレクトすることができる。ドメインを遷移する順序をあらかじめ定めておけば、分散システムのサーバを一通り巡って検索することが可能になるが、クライアントの視点から見れば、ひとつのサーバに依頼をだし、結果をひとつ受け取るだけに過ぎない。サーバ側に新しいサーバが追加された場合や、一部のサーバがダウンした場合にも、自己置換の遷移順序を一部改めるだけで、クライアントには何ら影響なく連続したサービスを提供することが可能である。

5 まとめと今後の課題

分散システムを始めから意識した、分散システムの性質を活かした柔軟な動的再構成ならびに柔軟な実行時処理を実現するためのDROPSプログラミングモデルについて、オブジェクトモデルと操作モデルの概要を述べた。そして、オブジェクト間の接続関係の操作を行うオブジェクトアダプタとして、バインド(関係付け)、案内置換を定義し、そして動的関連を用いた直接的な動的再構成よりも規則性のある動的再構成法として、複合オブジェクトとその操作を定義した。しかし、現段階では、再構成の隠蔽に関して、複合オブジェクトが媒体になるとしか定義されていない。具体的に、どのようなアクセス制限を設け、どのようなケースでどのような認証を行うのかをより具体的に定義し、検討する必要がある。

参考文献

[1] 山崎顕治, 都司達夫, 宝珍輝尚. “動的再構成が可能な分散オブジェクトシステムを支援するORBに関する考察”. 情報処理学会データベースシステム研究会報告, 97-DBS-111, pp.1-8, 1997.

[2] 山崎顕治, 都司達夫, 宝珍輝尚. “分散オブジェクトシステムの動的な再構成とカプセル化を実現するモデルの提案”. 第8回データベース工学ワークショップ (DEWS'97) 論文集, pp.167-172, 1997.

[3] Object Management Group. “The Common Object Request Broker: Architecture and Specification”. Revision 2.0, 1995.