マルチメディア通信と分散処理 76－7
（1996．5．16）

ワールドワイドな分散オブジェクト指向環境の構想

濱崎 陽一† 西岡 利博‡ 塚本 享治†
hamazaki@etl.go.jp nishioka@mri.co.jp tukamoto@etl.go.jp
† 電子技術総合研究所 ‡ 三菱総合研究所
〒305 つくば市梅園 1-1-4 〒100 千代田区大手町 2-3-6

分散アプリケーションの開発を容易にするワールドワイドな分散オブジェクト指向環境を目指して設計を進めている OZ+++ について、その構想を述べる。

OZ+++ はプログラムやサービスといったソフトウェア資源をネットワーク上で共有・利用し、それらを部品として利用する事によって分散アプリケーションの構築を容易にする。また、そうして開発されたプログラムなどをネットワーク上に環流することにより、ネットワーク上のソフトウェアの品質が向上してゆく。

ここでは OZ+++ のベースとなるOZ++システムの概要ならびに開発の過程で明らかになった問題点を示し、OZ+++ の構想と実現の方針を示す。

The Plan of Worldwide Distributed Object-Oriented Environment

Yoichi Hamazaki †, Toshihiro Nishioka ‡ and Michiharu Tsukamoto†
† Electrotechnical Laboratory ‡ Mitsubishi Research Institute, Inc.
1-1-4, Umezono, Tsukuba, Ibaraki, 301 Japan 2-3-6 Otemachi, Chiyoda-ku, Tokyo, 100 Japan

The plan of worldwide distributed object-oriented environment, OZ+++, which makes the development of distributed applications easy is described.

In OZ+++, software resources which involves programs and services are shared and used over networks help to develop distributed applications. Developped software are pro- vided as shared resources again and this cycle improves quality of software on the networks.

In this paper, the outline and problems of OZ++system which will be a base of OZ+++ are presented and the implementation plan is shown.

1 はじめに

近年、インターネットやWWWの普及などによって、ネットワークを使った情報の発信・共有・流通が、社会や産業に影響を始めている。しかし、ソフトウェアについては、その共有・流通が円滑に行われておらず、世界中の多くの人によって類似した内容のソフトウェアが繰り返し開発されており、このことが低い生産性や低い信頼性の要因となっている。殊に、分散アプリケーションは単一計算機上のアプリケーションに比べて、プログラミングが難しく、プログラムの改変が一部で行なわれると全体の相互運用性が損なわれるなどの問題がある。

これらの問題の解決をめざして、分散アプリケーションの記述が容易で、その実行が簡単に行なえるワールドワイドな分散処理環境 OZ+++ をオブジェクト指向技術を用いて構築しようとしている。

OZ+++ は、情報（データ）のみならず、ソフトウェア（プログラム）やサービス（サーバの提供する機能）をネットワーク上で発信・共有・流通することにより、ワールドワイドに分散した利用者や計算機がソフトウェアを共有・再利用できるようにし、拡張・改良されたソフトウェアがネットワーク上に環流するようなシステムである。

プログラマは、ネットワーク上のソフトウェアを部品として利用することによってプログラムの開発が容易となり、拡張・改良したソフトウェアを部品として再度ネットワーク上に環流することでソフトウェアが共有され、流通してゆく。また、ソフトウェアが多くのプログラマによって、いろいろな応用に利用される事により、バグの発見やプログラムの改良がなされ、ネットワーク上のソフトウェアの品質が次第に向上することが期待される。

本稿ではワールドワイドな利用を想定したオブジェクト指向分散環境 OZ+++ の構想について述べる。

2 想定する利用者

計算機の利用者は大きくわけて、質の良いプログラムの開発を行なう技術を備えたデベロッパと、プログラムの開発を行なわない、あるいは個人が利用する範囲での簡単なプログラミングを行なうエンドユーザに分けられる。

OZ+++ はデベロッパおよびエンドユーザに魅力的な以下のような特徴を持つ。

2.1 デベロッパ

分散アプリケーションの作成を行なうのは主としてデベロッパであるから、システムの第１の利用者、受益者としてデベロッパを想定する。他人が利用する事を前提としてプログラムを作成するには、相当の技術が必要とされ、それをしうるのがデベロッパである。ネットワーク上に環流するソフトウェアの作成はデベロッパによって行なわれるのであるから、デベロッパにとって魅力のないシステムは運用できない。そこで、デベロッパに魅力的な以下のような要素をもつ OZ+++ システムをめざす。

- 分散アプリケーションの記述が容易な言語とその処理系
- 分散アプリケーションの実行が簡単で、プログラムのデバッグが容易な実行環境
- 部品としてネットワーク上から入手でき、安心して使えるソフトウェア群
- ソフトウェアの改変による影響が理解可能で、また新旧のソフトウェアが同時に使用できる機構

2.2 エンドユーザ

エンドユーザは、デベロッパによって作成・環流されたネットワーク上のソフトウェアや、サービスを利用することによって、利益を受ける。エンドユーザは簡単なプログラミングもするであろうが、ソフトウェアを利用することが主である。

現在、FTP などで入手できるソフトウェアも多いが、エンドユーザにとってそれらを ftp し、システムにインストールする必要がある事が技術的、あるいは心理的な障壁になっている。OZ+++ を用いる事によって、エンドユーザは OZ+++ で開発された分散アプリケーションをインストール作業することなく利用できる。

エンドユーザでも、アプリケーションの設定変更（カスタマイズ）や、簡単なプログラミングが可能であるアプリケーションもある。そうしたエンドユーザ向けの分散アプリケーションは OZ+++ 上で開発され、実行されるが、エンドユーザが操作やプログラミングする言語（スクリプト言語）は、かならずしも OZ+++ の提供する言語である必要はなく、分散アプリケーションに依存する。

エンドユーザ向きのアプリケーションは、つぎのような特徴を備えている事が望ましく、OZ+++ はそうしたアプリケーションを作成するのに充分な機能を備え、そのためのフレームワークがライブラリとして提供される。

- GUI を用いて操作が容易であること

- 直観的で簡単なスクリプト言語あるいはGUI操作をもつこと
- インストールなどの付随作業が不要であること
- 不用意な操作がシステムに重大なダメージをあたえないこと

3 OZ+++ システムの構想

OZ+++ は、次のような分散環境を目指している。

1. ワールドワイドに分散した大規模な分散アプリケーションを開発できる、

   分散アプリケーションの開発が容易で、大規模な分散アプリケーションも開発することができる。

2. ワールドワイドに分散したソフトウェア・サービス・情報が共有・利用できる、

   ネットワーク上に提供されたアプリケーションを容易に利用することが出来る。

   また、新たなソフトウェアの開発に、ネットワーク上の部品が利用できる。

3. ワールドワイドにソフトウェア・サービス・情報が提供できる。

   開発したソフトウェア・サービス・情報をネットワーク上に環流することができ、それらは利用されることによって不具合の発見が促進され、ソフトウェアの品質向上がなされる。

このようなシステムを実現するためには、次の要求を満たすことが必要となる。それを実現するための方針を以下に示す。

- モジュラリティ

  ワールドワイドな環境では、ソフトウェア資源（ソフトウェアやサービス）の提供者と利用者が相互に関係を持つ事は稀であり、コンセンサスを仮定することはできない。

  そのため、ソフトウェア資源を有効に利用しあうには、明確なインタフェースを持ち、モジュラリティが高い事が必要である。

  OZ+++ では、オブジェクト指向の高いモジュラリティに注目して、オブジェクト指向に基づいたシステムとし、クラスをソフトウェア共有の単位とする。

- 分散透過性

  分散アプリケーションでは当然ながら分散透過性が実現されなくてはならない。特に、オブジェクトの位置とプログラムの位置の分散透過性が重要である。

  OZ+++ では分散アプリケーションを構成する分散配置されたオブジェクトは、分散透過に相互にメソッド起動ができる。オブジェクト間でメソッド起動に伴い引数あるいは返り値としてオブジェクトが授受できる。

  オブジェクトの授受に伴い、それらのクラスは自然に共有されることになる。このようなプログラムの実行時に発生するクラスの共有に対して、ワールドワイドな環境のいろいろな場所で不定期に生成されるクラスを事前にすべてのクラスを準備する事により分散透過性を実現するのは現実的ではない。

  そこで、OZ+++ では、必要に応じてクラスを自動的にロードして利用する機構を提供することによって、クラスの分散透過性を実現する。

- セキュリティ

  ワールドワイドな環境では、悪意のあるユーザによるソフトウェアやサービスの提供、サービスへのアクセスを想定して対処しておく必要がある。

  殊に、クラスを自動的にロードして利用するOZ+++ では、悪意のあるクラスをロードした際に、それを実行しないことが重要である。

  そこでOZ+++ では、配送されるクラスの検査が容易な中間コードによるクラスの配送を行ない、実行時のコードの検査によって安全に実行する方式をとる。

- ソフトウェアのバージョン管理

  ソフトウェアは通常、機能拡張や改良のために更新されるが、公開されたソフトウェアは他のユーザが利用している可能性があるために不用意に変更してはならない。

  そこでOZ+++ では、ソフトウェアのバージョン管理を行ない、新旧さまざまなバージョンのソフトウェアが同時に存在できるようにする。また、使用するバージョンの決定をインスタンス生成時まで遅らせることにより、最新のバージョンの部品が再コンパイルせずに利用できる

- 既存環境との共存

現在すでに開発された多くのソフトウェアがあり、それらによって蓄積されたデータがある。そうした蓄積を利用せずに、独自にシステムを最初から構築して行くことは、非現実的であり、そうしたシステムの用途は限られたものにならざるを得ない。

OZ+++ は他のシステムと共存するシステムであり、WWW や SMTP といったネットワークエージェントとつながるためのフレームワークや、他の言語で書かれたアプリケーションとつながるためのフレームワークを提供する。

こうしたフレームワークを用いて、既存環境を利用しあるいは既存環境から利用できる分散アプリケーションを開発することができる。

4 ベースとなるOZ++システムの特徴と問題点

OZ+++ に先だって、分散アプリケーションの開発を容易にするオブジェクト指向分散環境OZ++を開発した [1]。

OZ++では次のような機能が実現され、これらの機能は OZ+++ に引き継がれる。

- オブジェクトの共有と転送

  ネットワーク上で共有され、サービスを提供するオブジェクト（グローバルオブジェクト）と、グローバルオブジェクトを構成する部品となるオブジェクト（ローカルオブジェクト）からなるオブジェクトのモデルを用いた。

  グローバルオブジェクトはそのシステムで一意な ID(Object ID:OID) を持ち、ネットワーク透過にアクセスされることにより共有される。ローカルオブジェクトは、オブジェクト間のメソッド起動の引数または返り値として転送される。単純なデータ（整数や文字列など）のみではなく、オブジェクトが交換できる事で、複雑な構造を持つ情報でも容易に交換できる。

  オブジェクトをネットワークを介して転送できるように、オブジェクトのエンコード・デコードの機能を備えたオブジェクト転送機構を開発した [2]。

- クラスの共有と配送

  クラス（プログラム）は、分散システムにおいて当然共有されなければならない。しかし、分散した計算機のすべてに、システムのいろいろな場所で不定期に生成されるクラスを備える事は不可能であり、また非効率である。また、いっぽうでオブジェクトが転送された先で実行される際には、そのクラスが必要になることは明らかであり、必要となる場所は転送されるまで予測不能である。

  これを解決するために、必要なクラスが計算機上に存在しない場合には、ネットワークを介して、その所在を確認し、配送する機構を開発した。これによって、オブジェクトの転送を転送先でのクラスの有無を気にすることなく出来る。

  クラスの探索には、ネットワーク上のブロードキャストを用いた。

- クラスのバージョン管理

  ネットワーク上に公開されたクラスは不特定多数のユーザに利用されている可能性があるために、他のユーザに影響を与えずに変更することは困難である。そこで、クラスにバージョンを付与し、事なるバージョンのクラスが同時に存在しても矛盾が起こらないようにし、拡張・改良がいつでもできるようにした。

  バージョンはインタフェースの差異とセマンティックスに基づいて管理し、クラスを継承して利用した場合とインスタンス変数として利用した場合の影響の有無を区別して、再コンパイルがなるべく不要となるように工夫した。

  また、利用するクラスのバージョンの決定をインスタンス生成時まで遅らせることにより、より新しいバージョンのクラスを再コンパイルなしで利用できるようにした。

OZ++はこれまでLAN上で利用してきており、一応の機能を満たしている。しかしながら、OZ++をワールドワイドな分散環境とするためには、次のような問題や機能の不足がある。

- ネーティブコード配送方式の安全性

  OZ++はクラスのコードとしてネーティブコードを転送・実行する方式をとっている。この方式は実行速度の点で優れているが、転送されるコードが大きい事とクラスを悪意あるプログラムにすりかえられるなどの攻撃などへのセキュリティ対策が困難である欠点がある。

  対策としては、ネーティブコードではなくソースコードを転送して、それを再コンパイルしてチェックしたものを用いる方法や、クラスの登録機構を設けて、ネーティブコードのインテグリティを検査する方法などを検討した。しか

し、再コンパイルする方式は速度の点で、インテグリティを検査する方式はクラスの登録機構を信頼性高く運用する事が困難である点で問題がある。

また、ネーティブコード方式では計算機の機種毎にコンパイラを用意する必要があるために、多数の計算機への移植の手間が大きい。

- サイト間通信のセキュリティ機能

  アプリケーションゲートウェイによって、サイト外からのアクセスや転送されるオブジェクトに外来という特性を持たせ、外来のものの機能を制約する事により、インターネットセキュリティの確保を行なった。

  しかし、外部からのアクセスをすべて悪意あるもとのして扱う現在のアプリケーションゲートウェイは、信頼できるサイト間どうしでも操作を限定してしまうなどの問題がある。これは開発時点で利用できる広域認証システムが無かった事が一因である。最近、実用的なものがでてきた広域認証システムを利用するなどセキュリティ機能の高度化が必要である。

- ソフトウェア開発環境

  各ユーザのプログラム開発環境は一応揃っているが、広域で分散してソフトウェア開発をするための環境が整っていない。

  OZ++ では、広域に分散したクラスやオブジェクトを転送する機構は実現されたが、ソフトウェアを開発する際に必要とする適切なクラスや必要とするサービスを提供するオブジェクトをネットワーク上から捜し出したり、情報を公開したりための環境は整っていない。こうした環境の実現に必要となる管理方法や運用方法についても、システムの開発直後であることから利用・運用実験が充分でなく、未検討である。

  クラスではなく、インスタンスの形で部品を提供する試みも行なったが、まだ実験段階にある。

- 大きなシステム

  OZ++ではクラスのネーティブコードが大きいことから、クラスを管理するファイルシステムが大きなものになり、また実行系も大きなメモリを必要とする。

  これは、普及しているパーソナルコンピュータなどへの移植を困難にしている一因である。

5 OZ+++ システムの実装の方針

ワールドワイドな分散オブジェクト指向環境を構成するために、OZ++の問題点や機能の不足を次のように解決する。これらの開発に重点を置く。

5.1 セキュリティ

コードを配送するシステムではコードの安全性が重要である。OZ++のようにネーティブコードを配送する方式では、配送されてきたコードの検査が困難であるので、OZ+++ ではコードの検査が可能な構造を持ったバイトコードを中間コードとして採用し、中間コードによる配送を行なう。バイトコードでは、システムの停止などの危険な操作をする命令を含まないようにでき、またファイルへのアクセスなどは実行時にチェックする事ができるので安全である。

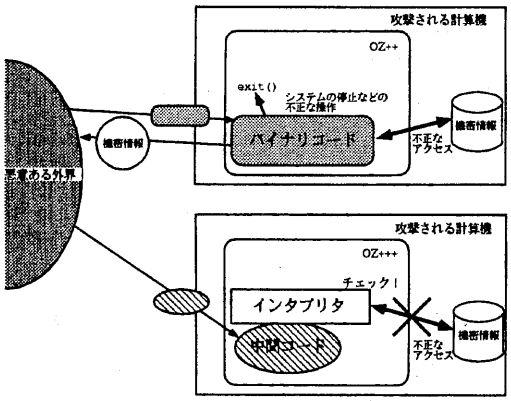

図 1: ネーティブコードと中間コードの安全性

また、アプリケーションゲートウェイについては、広域認証システムの利用などを検討して、高度化を図る。

5.2 システムの軽量化

ネーティブコードから中間コードに変更することにより、メモリ上にロードされるコードの占める大きさは小さくなる。その反面、実行速度の低下が予測される。オブジェクトを管理するオブジェクトマネージャやクラスを管理するクラスオブジェクトといった管理オブジェクトの性能が低下するとシステム全体におよぼす影響が大きい。そこで、OZ+++ ではクラスのコードで実現されている管理オブジェクトの機能部品をエグゼキュータに移行し、速度の低

下を抑えるとともに、システムの軽量化を図る。

OZ+++ を普及するために、今後広く使われると思われる OS(Solaris, Windows など) 上にシステムを実装する。

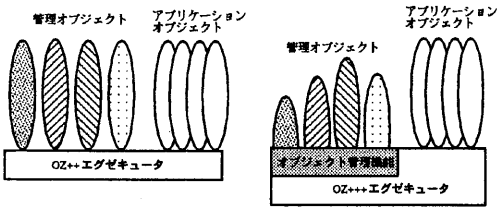

図 2: OZ++と OZ+++ のエグゼキュータと管理オブジェクト

5.3 ソフトウェア開発環境

ワールドワイドなソフトウェアを共有し、利用する環境を構築するためには、各種ブラウザを用意し、また運用レベルでの情報の管理が重要となる。次のようなツール群を拡充する。

- カタログ (公開されるクラスのパッケージのサーバ) およびジャンクショップ (公開されるオブジェクトのテンプレートのパッケージのサーバ)
- カタログおよびジャンクショップのブラウザ
- クラスのブラウザ (クラスの情報と、クラスの階層、クラスのバージョン階層、メソッドクロスリファレンスなどの表示)
- トレーダ (サーバ利用支援)
- バージョン管理支援ツール (クラスのバージョンアップに影響を受けるクラス群の検出と通知、インタフェース保存的な変更の支援など)
- 分散開発支援ツール (依存関係に基づく工程管理、ソースプログラムの管理、開発者グループ内の名称管理)

また、管理オブジェクトやサーバなどを堅牢なものにするため使える、サーバの複製管理やトランザクションのためのクラスライブラリを拡充する。

5.4 エンドユーザ向けアプリケーション

エンドユーザには、利用できるアプリケーションを検索するアプリケーションカタログのブラウザが提供される。エンドユーザはブラウザからアプリケーションのオブジェクトを得て起動するだけで、クラスの自動配送によりインストール作業なしに分散アプリケーションを利用できるようになる。

どのような分散アプリケーションを提供するかは検討中である。

6 まとめ

OZ++をベースに、ワールドワイドな分散環境を目指して検討を行なっている OZ+++ について、その構想を述べた。OZ+++ は分散アプリケーションの開発を用意にする環境であり、そこではソフトウェアやサービスといったソフトウェア資源が共有・利用され、それらが改良されて環流する仕組みが提供される。

OZ+++ によってワールドワイドな環境でソフトウェアの共有・流通が促進され、ソフトウェアの生産性や信頼性の向上に寄与できる事を期待している。今後、更に検討をすすめ、開発に着手する予定である。

OZ++の研究は、情報処理振興事業協会「開放型基盤ソフトウェア研究開発評価事業」の一環として行なわれたものである。

参考文献

[1] 塚本、濱崎、西岡、音川：" クラスの共有と配送にもとづくオブジェクト指向分散システムの設計と実現"、情報処理学会論文誌、Vol.37、No.5、1996.

[2] Hamazaki,Tsukamoto,Onishi,Niibe:"The Object Communication Mechanisms of OZ++ : An Object Oriented Programming Environment", Proc. of Conf. on Information Networking (ICOIN-9), 1994.