7L－3

オブジェクト指向分散環境OZ++における オブジェクト指向設計モデル

中川 祐*(富士ゼロックス情報システム)　音川 英之*(シャープ)
新部 裕*(三菱総合研究所)　塚本 享治(電子技術総合研究所)
*:開放型基盤ソフトウェアつくば研究室研究員

1 はじめに

OZ++は分散システムをオブジェクト指向で開発し運用できる環境である。現在までにその基本システムがほぼ完成し、動作している[1]。また開発、運用に必要なツール、ライブラリ群を整備中である。開発の進行に伴い、設計時点では十分に把握できなかったシステムの特徴、問題点がある程度わかってきた。

本稿ではOZ++システムをオブジェクト指向の観点から検証し、分散オブジェクト指向プログラミングにおける問題点を抽出するとともに、設計法を検討する。

2 OZ++のオブジェクト指向的特徴と設計モデル

オブジェクト指向言語にはデータ抽象、カプセル化、モジュラリティー、継承階層といった要件が必要とされている。さらに型検査、並行性、永続性といった要件を満たすことが望ましいとされている[2]。

OZ++はこれらすべての要件を満たし、さらに分散処理のための機構を採り入れている。

以下、OZ++の特徴をオブジェクト指向の観点から列挙する。

2.1 並行実行

OZ++は複数のスレッドがメモリを共有し得る並列処理環境である。並行プログラミングにおいて排他制御を実現するために「モニタ」の概念が広く知られている[3]。オブジェクト実行系ではその機能を提供している。これは排他処理を行なうデータとそれをアクセスする手続きとの集合である。

OZ++言語ではオブジェクトがモニタを構成する。モニタを構成するデータと手続きはそれぞれオブジェクトの属性とサービスが該当する。wait、signalの機構は言語上の構文として用意されている。

モニタはオブジェクト単位で提供されるため、継承の関係（is-a関係）では下位クラスは上位クラスのモニタを共有して用いる。対してオブジェクト間の関連（has-a関係）ではそれぞれのオブジェクトがモニタを保持する。

2.2 継承

OZ++での継承は下位クラスが上位クラスのメンバの公開レベルを変更できる点に特徴がある。これは継承がインターフェースの再利用だけでなく、実装の再利用としても用い得ることを示している。

オブジェクト指向の基礎的な手法では継承は上位クラスのインタフェースを下位クラスが再利用するために用いる。しかし先に述べたようにOZ++では継承はモニタを共有するための手段でもあるため、インタフェースの観点だけで設計を進めることは誤りである。

例えば入出力ストリームクラスを設計するとして、インタフェースの観点からいえば入力ストリームクラスと出力ストリームクラスを多重継承して用いるのが正解といえる。しかし入出力ストリームクラスにおいては入力と出力を別々のモニタとして利用したいことが多い。もし2つの上位クラスが直接モニタを保持していたならば、入出力クラスの実装は少々困難である。

一般には外部モジュールに対してインタフェースを提供するクラスはモニタを直接には保持せず、モニタを提供するオブジェクトを保持する。すなわち1つの機能を提供するためにインタフェースを提供するクラス（外部クラス）と機能を提供するクラス（内部クラス）とに分割する設計が望ましい。OZ++ではすべてのクラスはモニタとして利用できるが、1つのモニタは1つのクラスとして設計し、他のサービスは別クラスで実現する。

Methods of Object-Oriented design in OZ++: An Object-Oriented Distributed Systems Environment
Yu Nakagawa*(Fuji Xerox Information Systems, Co., Ltd.),
Hideyuki Otokawa*(Sharp Corporation),
Yutaka Niibe*(Mitsubishi Research Institute, Inc.),
and Michiharu Tsukamoto(Electrotechnical Laboratory)
*:Researcher, TsukubaLaboratory, Open Fundamental Software Technology Project

またモニタを共有するクラスのみを下位クラスとして設計する。

2.3 オブジェクトの保存

OZ++はオブジェクトをファイルシステムに保存して繰り返し利用できる環境である。保存する単位は「セル」というオブジェクトの集合である。セルはそれを代表するオブジェクトを1つ持ち、そのオブジェクトから直接または間接に参照されるオブジェクトすべてをセルに属するオブジェクトとみなす。1つのオブジェクトを複数のセルで共有することはできない。

2.4 オブジェクトの分散

OZ++システムは同一のセルに属するオブジェクトを同一のメモリ空間上に置くことを保証している。OZ++は遠隔ホストにおけるサービスをオブジェクトに対するメッセージパッシングの形で実現しているが、セル間のメッセージとセル内のメッセージとをシステムは明確に区別している。セル内メッセージは呼出側と呼ばれ側がメモリを共有できるため引数、返り値が参照で渡される。対してセル間メッセージはメモリを共有し得ないのでコピー渡しである。コピーされたオブジェクトはコピー元のオブジェクトとクラスを同じくする別のオブジェクトである。実行コードもコピー元のオブジェクトと同じものが利用される。

セル内のオブジェクトはメモリ空間を共有するため、セル内メッセージは安全、高速に動作する。またオブジェクトを参照で渡すので、通常のオブジェクト指向のモデルと適合する。セル内でのサービスは粒度の細かい緻密なものが可能である。セル間のサービスはセルを配置する位置が自由である反面、サービスは粒度の粗い、リスクを伴うものとなってしまう。

オブジェクト指向設計ではオブジェクトの関連を設計し、必要に応じてクラスの分割を再設計する。OZ++では2つのオブジェクトがサービスのやりとりを行なう場合、その頻度と情報量を考慮してそれらをどのセルに置くべきかを決定する必要がある。

セルのインタフェースはセルを代表する1つのオブジェクトのインタフェースと同等である。クラスのインタフェースとクラス間の関連を設計するのと同様に、セルのインタフェースと関連を注意深く設計する必要がある。

2.5 オブジェクトの生成

OZ++ではクラスの1つのインタフェースに対して複数の実装が混在して動作することが可能である。

OZ++はオブジェクト指向の処理系であり、メッセージの呼び出しは動的に束縛される。あるオブジェクトの顧客はオブジェクトをインタフェースとして捉え、オブジェクトにメッセージを送信し、その時実行されるコードは特定できない。OZ++ではこれがバージョン方向にも行なわれる。

バージョンはオブジェクトを生成する時点で決定するが、これはプログラムではなく実行時環境によって決定する。このためクラスの顧客が予期していないコンフィグレーションのオブジェクトが生成され、それによってプログラムが正常に動作しないということも起こり得る。生成されるオブジェクトを限定するためにはプログラムを実行する前にコンフィグレーションセットというデータを用意する必要がある [4]。

そのためクラスを設計する場合、オブジェクトの生成関係（Creates-a 関係）は他の関係と区別して抽出する必要がある。設計者は自身が生成するオブジェクトを抽出し、そのクラスのバージョン変更の可能性も考慮しなければならない。

3 課題

OZ++は分散システムであるため不正な計算資源の利用等、悪意を持った利用に対する抑止を考慮する必要がある。特にOZ++ではメッセージが動的に束縛され、必要な実行コードが自動配送される特徴を持つために、さまざまな危険が考えられる。

現在システムの観点からその対策を検討中であるが、特に公共性の高いサービスを提供するクラスにおいてはプログラミングの観点からも安全な設計法というものが求められるであろう。

謝辞

本研究は、情報処理振興事業協会 (IPA) の「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] 西岡他：「オブジェクト指向分散環境 OZ++ システム第一版の実現」,Swopp '95, Aug.1995

[2] Grady Booch:"Object-Oriented analysis and design with applications second edition", Benjamin/Cu Publishing Company, 1994

[3] C.A.R. Hoare:"Monitors, An Operating System Structuring Concept", commun. of the ACM, vol.17, Oct. 1974

[4] 音川他：「オブジェクト指向分散環境 OZ++ のプログラミングパラダイム」,Swopp '95, Aug.1995