オブジェクト指向は今：世紀末から新世紀へ

中　所　武　司†

オブジェクト指向技術．それは30年前のモデリング技術から始まった。Simula67のクラスの概念は、70年代の構造化の時代を経てSmalltalk-80に引き継がれ、80年代には既存の言語をベースにしたマルチパラダイム型の言語が開発され、最近のJavaに至る。80年代後半のオープンシステム化の急激な変化の中でオブジェクト指向技術は新しいソフトウェア構築技法としての地位を確立し、90年代には広範なソフトウェアの分野に適用されていった。しかし、オブジェクト指向プログラミングにおける成功体験がそのままの形で適用できるわけではない。今、それぞれの分野で新しい時代のソフトウェア技術に向けての努力がなされている。

Object-Oriented Technology : from the End of a Century to the Next

Takeshi Chusho †

The history of object-oriented technology started 30 years ago with Simula67. The concept of a class was inherited from Simila67 via structured programming languages of 70's to Smalltalk-80. In 80's, many object-oriented programming languages based on conventional languages were developed and, in 90's, Java has come out. Object-oriented technology has established fundamentals for building open systems of the mainstream and is spreading over various software fields.

1. はじめに

オブジェクト指向。それは30年前のモデリング技術にはじまり、その時々の追い風を受けて発展してきた。最近では、オープンシステム、インターネット、マルチメディアという時代の流れの中で、新しいソフトウェア構築技法としての地位を確立し、広範な分野に浸透している。

プログラム内蔵方式のコンピュータが開発されて以降、ほぼ50年続いた手続き型パラダイムが、西暦2000年問題に象徴されるような世紀末の様相を見せているときに、オブジェクト指向技術は新世紀に向かってなおも発展し続けている。ここでは、幾つかの視点からオブジェクト指向技術の過去、現在、未来について述べる。

2. 30年の歴史

2.1 Simula67から構造化言語へ

60年代後半から70年代半ばにかけてオブジェクト指向技術の基本概念が確立していった。30年前にノルウェーで開発されたシミュレーション用の言語Simula67[1)]では、すでにオブジェクト指向技術の重要な概念の幾つかが導入されている。その当時、シミュレーションモデルはデータ構造とその演算から構成され、それらの再定義と修正の繰返しによって作られていた。そこで、対象ごとにモデルを簡単に作るためにデータ抽象化、クラスからのインスタンス生成、クラスの階層と継承などの機能が導入された。また70年頃にはMITのC. Hewittがメッセージパッシングを主体とした計算モデルであるActorモデルを提案している。

70年代は構造化技法に代表されるように、プログラムの記述容易性よりも理解容易性が重視され、オブジェクト指向概念と関連の深い情報隠蔽、カプセル化、データ抽象化などの技法を取り入れた言語として、CLU(MIT), Mesa(Xerox PARC) , Ada(DoD) などが実用化された。

2.2 Smalltalk-80からマルチパラダイム型言語へ

80年代は、ソフトウェアの諸問題は手続き型パラダイムに起因するという観点から、種々の新しいソフトウェアパラダイムが追求された。その中で理想解とされた宣言型パラダイムではなく、オブジェクト指向パラダ

† 明治大学理工学部情報科学科
Department of Computer Science,Meiji University

イムが現実解としての地位を確保した。

そのさきがけとなったのがSmalltalk[2]である。Smalltalkは、Simula67のクラスの概念、およびオブジェクト間通信というアイデアを基礎として、Smalltalk-72、Smalltalk-74を経て、1980年にビジュアルなプログラミング環境とともにSmalltalk-80として公開された。

このSmalltalkをきっかけにオブジェクト指向への関心が急速に高まり、既存の言語をベースにオブジェクト指向機能を導入したハイブリッド型またはマルチパラダイム型と呼ばれる言語が数多く開発された。その例として、論理型言語Prologをベースとした ESP(ICOT)、Lispをベースとした CLOS(ANSI)、C言語をベースにしたC++(ATT)、などがある。

一方、この時期には人工知能の分野でも知識工学の研究が活発になり、知識表現言語にオブジェクト指向概念を取り入れたエキスパートシステム構築ツールが数多く開発された。人工知能分野ではすでにフレーム表現が階層化と継承機能を有していたため、それを拡張する形でオブジェクト指向概念が取り入れられた場合が多い。

2.3 オープンシステムとの歴史的出会い

90年前後からオブジェクト指向概念の応用技術分野が急速に拡大している。その代表的なものとして、ソフトウエアの使い勝手の良さを決めるグラフィカルユーザインタフェース(GUI)とその構築ツール、マルチメディアデータを扱うためのオブジェクト指向データベース、分散システム構築のためのオブジェクト指向オペレーティングシステムやネットワーク管理などの分野がある。

さらに、アプリケーション開発のためのオブジェクト指向分析・設計技法や部品化・再利用をめざしたコンポーネントウェアなどのソフトウエア生産技術やその共通プラットフォームとしての分散オブジェクト管理システムなど、オブジェクト指向ベースのソフトウエア開発環境と実行環境が充実しはじめている。

この背景としては、ネットワーク下でのアプリケーションの相互運用性や移行性を重視したオープンシステムへの急激な変化がある。すでに20年以上の歴史を有していたオブジェクト指向技術がこのオープンシステムを実現するための最適なソフトウェア構成法として採用されたという意味において、この時期はオブジェクト指向技術とオープンシステムとの歴史的出会いがあったといえる。

3. 世紀末のパラダイムシフト

19世紀末のデカダンスがどのようなものであったかは知る由もないが、西暦2000年問題は50年間続いた手続き型パラダイムの世紀末を象徴していると言えないだろうか。希望的観測をまじえて言えば、今こそ、手続き型パラダイムからオブジェクト指向パラダイムへとソフトウェア開発のパラダイムシフトが完結する最後の段階に来ていると思われる。

ここでは、オブジェクト指向プログラミングの実績と現時点でのオブジェクト指向技術の応用分野の広がりについて述べる。

3.1 オブジェクト指向プログラミングの利点

3.1.1 アプリケーション・アーキテクチャ

標準化を求めるオープンシステム化に対応して、稼動環境を含めたアプリケーションのアーキテクチャは次のように構成されることが多くなっている。

・階層化：基本ソフトやミドルウェアを含む階層構造
・部品化：各階層を部品集合で構成
・標準化：階層間のインタフェースを標準化

オブジェクト指向技術はこのようなソフトウエア構成に適した技術として実際に適用され、オープンシステム時代の標準インタフェースを提供している。アプリケーション・フレームワークやコンポーネントウェアはその具体例である。

このようなアーキテクチャは3種類のソフトウェア関与者、すなわち開発者、保守者、利用者に以下のような利点をもたらす。

3.1.2 開発者のための部品化・再利用

開発者は部品化・再利用に関して以下の利点を得る。
(1) 分散協調型モデルにより部品の抽出、利用が容易
(2) データ抽象化により独立した機能部品の構築が容易
(3) インスタンス生成により部品のカストマイズが容易
(4) 継承機能により部品ライブラリの構築が容易

3.1.3 保守者のための拡張性と移行性

保守者は拡張性と移行性に関して以下の利点を得る
(1) データ抽象化により機能変更や移植が容易
(2) 継承機能により機能追加が容易

3.1.4 利用者のための操作性と統一性

利用者は操作性と統一性に関して以下の利点を得る。
(1) 分散協調型モデルにより画面上のビジュアルオブジェクトにメッセージを送るという簡単な操作を実現
(2) インスタンス生成機能により共通部品を用いてGUIを統一

3.2 応用分野の広がり

新しいトレンドが追い風となって、オブジェクト指向技術の応用分野が広がっており、ソフトウェア技術のキーテクノロジーの地位を確立しつつある。以下では代表的なソフトウェア分野の動向を簡単に述べる。

3.2.1 開発方法論

アプリケーションの短期開発と頻繁な機能拡張という社会的ニーズに対応して、90 年前後から多くのオブジェクト指向分析・設計技法が提案された。それはちょうどオブジェクト指向プログラミング技法がオープンシステム実現のための基本技術として定着した時期にあたる。その実績に支えられて、その上流工程にもオブジェクト概念が適用されはじめた。

最初に何に着目するかという開発手順の観点でこれらの技法を大まかに分類すると次のようになる。

(1) 最初にクラス間の静的な関係に着目する方法
(2) 最初にシステムの動的なふるまいに着目する方法
(3) 複数の技法の融合方式（Fusion, UML, VMT）

これらの３つのグループはおおむねこの順に提案されてきた。オブジェクト指向分析・設計技法においては適切なオブジェクトをどのようにして見つけだすかという課題が重要であるが、この分類はそのためのアプローチの違いともいえる。

3.2.2 プログラミング言語

オブジェクト指向プログラミング言語の代表例として Smalltalk、既存の言語をベースにしたマルチパラダイム型言語の代表例としては C++ がある。最近では、Java がインターネット向きであることや C++ より安全であることなどのためにかつてないほど急速に普及している。

3.2.3 GUI ツールキット

オブジェクト指向技術に基づく GUI は次のような特徴があり、すでに多くの GUI 構築ツールが広く利用されている。

(1) メタファとしてのアイコンによる操作対象の表示。
(2) そのアイコンを指定して、操作メニューから必要なメソッドを選ぶ、という直感的な直接操作。
(3) すべてのアプリケーションをこの「オブジェクトにメッセージを送る」という直接操作で統一。

3.2.4 コンポーネントウェア・パターン・フレームワーク

このように GUI の分野ですでに部品化・再利用の効果をあげている理由としては、個々のボタンやスクロールバーなどの GUI 部品はその機能が容易に推測できるので利用しやすいことがあげられる。しかしながら一般には抽象データ型のレベルのクラスライブラリでは部品としての粒度が小さく、プログラミング技術レベルの知識が要求されるので、なかなか部品の有効活用が難しい。そこで、従来の部品の概念を発展させた技術として、コンポーネントウェア、アプリケーション・フレームワーク、デザインパターンなどが注目されている。

コンポーネントウェアの定義は確立していないが、ビジュアルプログラミングの分野では「クラスからインスタンスを生成する」という概念を意識させないで、すぐに機能するインスタンスオブジェクトをコンポーネントとして扱うものが多い。インターネット経由で取り込んですぐに利用できるコンポーネントも増えている。このようなコンポーネントは、従来のクラスライブラリの部品を利用するときのようにスーパークラスも含めたプログラムの定義内容を知る必要はなく、ブラックボックスとして扱えばよいので利用が簡単になっている。

アプリケーション・フレームワークは、ある程度応用分野を特定することにより、そのソフトウエア・アーキテクチャの基本的な枠組みを構成するクラスライブラリを与えるものである。個々のアプリケーションはそのフレームワークをカストマイズして作成することになる。

デザインパターンは複数のクラス間の典型的な協調作業をパターン化したものである。クラス部品を利用するときの工法や定石にあたるものであり、見かけ上の部品の粒度を大きくして使いやすくする役割を有する。

3.2.5 分散オブジェクト管理

オブジェクト指向ベースのアプリケーション・ソフトウエアの普及に対応して、アプリケーション間に共通のオブジェクトの管理をアプリケーションから分離、独立して共有化するためのプラットフォームを標準化しようとする活動が活発に行なわれている。

標準化団体 OMG が普及をはかっている CORBA では、アプリケーションは必要とするサービスを ORB(Object Request Broker) に依頼するが、そのオブジェクトがネットワーク上のどこにあるかは知る必要はない。そのためにネットワークオブジェクトの名前を管理する機構が用意されている。さらにアプリケーション間の相互運用性の範囲を広げるために、異なる ORB 間の接続やインターネット経由の接続も検討されている。マイクロソフト社の DCOM も同様の機能を有する。

3.2.6 データベース

従来のデータベースが主に数値や長さの限られた文字列を扱ってきたのに対し、最近の情報システムでは文章、図表、画像、音声、動画など、データ構造が複雑なものを扱う必要が生じてきた。このようなマルチメディアデータを扱うために 90 年頃にオブジェクト指向デー

タベース（OODB）が登場した。従来の RDB と異なる特徴は次のようなものである。

(1) マルチメディアデータの統一的管理
(2) データの同じ複数の実体を個別に識別して管理
(3) 複合オブジェクトの統一的管理
(4) クラス階層による実世界の知識の直観的な管理
(5) データと手続きのカプセル化による処理の簡素化
(6) プログラミング言語との連携

OODB の標準化団体 ODMG では、オブジェクトモデル、オブジェクト定義言語、オブジェクト問い合わせ言語などを制定している。一方、 RDB 用の言語 SQL の標準化を推進してきた ISO でもオブジェクト指向機能を取り入れた SQL3 を開発している。

3.2.7 オペレーティングシステム

オープンシステム時代には、 OS には次のような新たな機能が求められる。

(1) 抽象的なレベルでのアプリケーションとのインタフェースの標準化
(2) OS 自身の拡張性と移植性
(3) 分散処理や並列処理の機能の実現と安全性や信頼性の確保の両立

オブジェクト指向 OS では、 (1) に関しては、各種のリソース管理をオブジェクトという概念でカプセル化することにより、リソース固有の処理をアプリケーションから見えなくしている。

さらに OS の基本部分はマイクロカーネルとして実現しておき、独立性の高いオブジェクトを組み合わせる方式により (2) を実現できる。

(3) に関しては、従来のプロセスやタスクに代わり、オブジェクトにメッセージを送るというメッセージ駆動型の分散協調型モデルを用いて、柔軟な分散、並列システムを構築することができる。この場合、オブジェクト単位にアクセス権を設定することにより安全性と信頼性を確保できる。

3.2.8 ネットワーク

ネットワーク管理に関しては、管理する側と管理されるシステムを明確に分離し、相互にメッセージをやりとりするオブジェクト指向モデルが採用されている。管理対象のネットワーク機器や回線はそれぞれの属性を持ったオブジェクトとして扱われる。これらの管理対象オブジェクト群はエージェントが監視している。管理する側はマネージャと呼ばれ、管理対象オブジェクトへの操作の指示や管理対象オブジェクトで発生した事象の通知はこのエージェントとのメッセージ交換により行う。このようなモデルにより管理対象の拡張や変更が容易になる。

4. 新世紀に向かって

変化の激しい時代にオブジェクト指向技術の将来展望を語ることは難しいが、ソフトウェア工学的課題を解決するための通常の技術としてなおも発展し続けていくと思われる。今世紀の手続き型パラダイムがそうであったように、新世紀の乗り越えられるべきパラダイムとしての立場に位置することはまちがいないであろう。

インターネットとマルチメディアの時代のキーとなるアプリケーションを支えるインフラストラクチャとして、アプリケーションフレームワークやデザインパターンを含めたコンポーネントウェアや分散オブジェクト管理システムなどの基盤は強固なものになっていくであろう。[3)〜5)]

そしてそのようなアプリケーションの主要なユーザが情報処理技術に無関係な一般の人達であることから、エンドユーザコンピューティングという言葉に新たな意味がもたらされるものと思われる。

5. おわりに

今、新しい時代にふさわしい新しいソフトウェアの作り方が求められている。インターネットの時代を迎えて、これまでの「初めにハードウェア（機械）ありき」から「初めにエンドユーザ（人間）ありき」という発想への転換期にある。ソフトウェア工学における 90 年代を特徴づけるオープンシステムとオブジェクト指向技術の歴史的な出会いも、オブジェクト指向技術の本質がモデリングであったという意味においてはエンドユーザ指向という流れの中での歴史的必然であったと思われる。

参考文献

1) Dahl, O. J. and Hoare, C. A. R. : Hierarchical Program Structures, Structured Programming, Academic Press, 175-220, 1972.
2) Goldberg, A. and Robson, D. : Smalltalk-80 : The Language and its Implementation, Addison Wesley, 1983.
3) Mellor, S. J. and Johnson, R.(Ed.) : Special issue of Object Methods, Patterns, and Architectures, IEEE Software, 14, 1, 27-72, 1997.
4) Schmidt, D. C., Fayad, M. and Johnson, R. E.(Ed.) : Special issue of Software Patterns, Commun. ACM, 39, 10, 36-82, 1996.
5) 青山幹雄：コンポーネントウェア：部品組立て型ソフトウェア開発技術, 情報処理学会誌, 37, 1, 71-79, 1996.
6) 中所武司：ソフトウェア工学 ーオープンシステムとパラダイムシフトー、朝倉書店、 1997.