合同 ソフトウェア工学 96－7
プログラミング－言語・基礎・実践－ 15－6
（1994．1．20）

パネル討論：

オブジェクト指向は本当に役立っているのか
― ソフトウェア工学の立場およびソフトウェア科学の立場から ―

パネリスト：　本位田真一 ((株) 東芝)
佐伯元司 (東京工業大学)
久世和資 (IBM 東京基礎研究所)
加藤和彦 (筑波大学)
司会：　大蒔和仁 (電子技術総合研究所)

概要

パネルの目的はエンジニアリングとサイエンスの立場から「オブジェクト指向」を見つめなおすことである。いまのところ、両方の側共に「オブジェクト指向」が優れたソフトウェアパラダイムと考えているように思われる。しかし、パネルでは、本当にそうだろうかと問い直したい。今回の研究会はソフトウェアに対してそれぞれ工学の立場および科学の立場という基本的異なる文化を有している二つの研究会の連続開催として実施されている。そこで現在研究の最も前線で活躍されている方々を両方の研究会からパネリストとしてお願いし忌憚なきご意見を頂く。

Panel Discussion:

Are "Object-Oriented Approaches" really Useful?
- From the Viewpoints of both Engineering and Scientific Sides -

Panelists:　Shin-ichi Hon-iden(Toshiba Corporation)
Motoshi Saeki(Tokyo Institute of Technology)
Kazushi Kuse(IBM Tokyo Research Institute)
Kazuhiko Kato(University of Tsukuba)
Coordinator:　Kazuhito Ohmaki(Electrotechnical Laboratory)

Abstract

The purpose of the panel session is to discuss usefulness of object-oriented (OO) approaches on software from the viewpoints of both engineering and scientific research areas. Many researchers or practitioners seem to believe that OO approaches are promising ones. During the panel discussion, we would like to re-discuss on the usefulness of OO. The four panelists are very active researchers from two special research interest groups of engineering and scientific research fields. We can expect that they will give us their positions and perspectives.

合同 ソフトウェア工学 96-7-1
プログラミング－言語・基礎・実践－ 15-6-1
（1994. 1. 20）

オブジェクト指向はソフトウェア開発者の意識を変えた ーソフトウェア工学の立場からー

本位田真一
（株）東芝 研究開発センター システム・ソフトウェア生産技術研究所

平成6年1月20日

1 はじめに

企業の立場では、オブジェクト指向が本当に役に立っているとは、従来からの懸案事項であった

- 上流工程の作業の合理化
- 部品化再利用の促進
- 仕様変更に対する作業コストの低減

などが、オブジェクト指向を導入することによって大幅に改善され、ソフトウェアの生産性が飛躍的に向上し、結果としてソフトウェア事業が大きく社業に貢献している状況を指す。では，現状において、上の懸案事項が大幅に改善されているであろうか？

現時点での答は当然ＮＯである。しかし、10年先においても、やはりＮＯだとしたら、その企業においては、もはやソフトウェア事業は成立し得ないと考えなければならないだろうと断定したい。なぜならば，筆者は2000年初頭においては，オブジェクト指向が本当に役に立っているという，オブジェクト指向擁護派の立場をとっているからである．

2 現時点ではどれだけ役に立っているのか

先のことはともかくとして、現時点において、どれだけ役に立っているのかを考える。

少なくとも次のように意識の改善ができたと言えるのではないか。

1. 下流工程が楽になるから，上流工程からきちんとやろう．
2. 再利用によって生産性を向上させるために，部品の標準化を促進させよう．
3. 役割分担を明確にするために，ソフトウェア開発体制を見直そう．

これらについて少し述べることにする。

2.1 上流工程

従来は、上流工程で頑張っても、その成果物が素直に下流工程に活きてこないため、上流工程に作業コストをかけても、下流工程の作業は決して楽にはならないと信じられてきた。実際のシステム開発では、いわば2度手間をする時間がないため、(上流工程の重要性は充分認識しているのにもかかわらず)上流工程に工数をかけている余裕がなかった。その弊害として、大きな手戻りが発生することになり、多くの工数が余計に割かれることになる。

オブジェクト指向によって工程間の一貫性が実現され、上流工程における苦労(成果物)が下流工程において報いられるという期待感が高まる。下流工程のオブジェクト指向言語のユーザが、できる限り早い段階で安定性の高いオブジェクトを見つけたいという素直な欲求も生まれる。こうした期待感によって，上流工程への作業コストの注入が始まった。これはオブジェクト指向による貢献ではないか。

2.2 標準化

従来より、標準化の重要性は広く認識され、部門によっては、標準化チームが編成され、ソフトウェアの生産性を向上させるポイントとなる標準化部品

の構築を目指してきた。しかしながら、部品の単位をどうするか、体系化をどうするか、検索、修正のやり方といった課題は、未解決の状態であった。そのため、標準化チームの作業は遅々として進まず、場合によっては，解散の憂き目にあうことになる。この原因は、標準化を促進し得る技術的な裏付けのある手段が存在していなかったといえる。

それに対して、オブジェクト指向は、部品化における上記の課題に対して、オブジェクトとしての単位、クラス・ライブラリによる体系化、検索の容易性、差分プログラミングによる修正の容易性を実現している。部品の標準化を真剣にやってみようという意識を持たせた意義は大きいと考える。言うまでもなく、オブジェクト指向方法論による生産性向上のポイントは、クラス・ライブラリを用いた部品化再利用である。今後は、部門ごとに、その部門に特化した標準部品ライブラリ（クラス・ライブラリ）が数多く生まれ、また、ある目的に沿ったクラス・ライブラリの商品化も活発になる。

また、ソフトウェア開発の様々な工程で生じる仕様変更に対しても、オブジェクトによって耐性は強化され、従来技術と比較して、コストの低減化がはかれる。

2.3 ソフトウェア開発体制

どの部門にも、そこで扱っているソフトウェアシステムについて、何でも良くわかっている人が存在するものである。そうした経験豊富な限られた人たちが、その部門のソフトウェア事業を支えている。しかし、従来の方法論では、そうした人たちを優遇してきたとは言えないだろう。あるいは、彼らの経験やパワーを十二分に活かしてきたとは言えないだろう。

オブジェクト指向方法論はそうしたスーパプログラマの地位を高めることに貢献できる。現在のオブジェクト指向方法論の問題点として、オブジェクトの切り出しが難しい（これをオブジェクト獲得ボトルネックと呼ぼう）と言われている。これは確かに事実であるが、必ずしも、プロジェクトに属す多くのメンバがオブジェクトを切り出す必要はなく、ごく限られた信頼のおける、経験豊富な人（スーパプログラマ）がオブジェクトを切り出せば良いと考える。すなわち、スーパプログラマが切り出したオブジェクトを標準部品として登録し、それ以外の人は、標準部品ライブラリに登録された部品を修正しながら再利用する形態になる。オブジェクト指向方法論によって、ソフトウェア開発体制において，スーパプログラマと一般プログラマの役割分担が明確になったといえる。

また，従来より，経験豊富な人の有しているノウハウを，どのように蓄積し，活用するかが大きな課題であった．上流工程におけるクラス・ライブラリは，ノウハウの集合体である．なぜならば，抽象的な仕様をより具体的かするプロセス (これが重要なノウハウ) を自然に表現しているからである．

3 結論

ソフトウェアの生産性向上には、方法論の普及が必要であるが、そのためには、まず意識革命が必要である。オブジェクト指向が本当に役に立つまでには、時間が必要である。そうした意味では意識の変化が徐々に浸透し始めたところであり、現時点では役に立っているとすべきではなかろうか。

以上は、オブジェクト指向方法論を擁護する立場で述べたが、当然のことながら、克服しなければならない問題点は山積みである。研究者の立場からは、本当に役に立つために，やることのできる新しい研究テーマがいくらでもあると言いたい。

参考文献

[本位田 93] 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社，1993

合同 ソフトウェア工学 96-7-2
プログラミング－言語・基礎・実践－ 15-6-2
（1994. 1. 20）

オブジェクト指向は役に立つか？
― ソフトウェア工学の立場から

佐伯 元司
東京工業大学工学部情報工学科

1 はじめに

世の中にオブジェクト指向 (Object Orientation) という言葉が登場して１０年以上が経過した．最初は、SmallTalk80 に代表されるオブジェクト指向型のプログラム言語から始まり、Booch のオブジェクト指向設計法を経て，オブジェクト指向を基礎とするソフトウェア開発法やソフトウェアプロセス支援環境が開発されるに至っている．果たして，対象世界をオブジェクトというもので捉えようというオブジェクト指向の考えは，ソフトウェアの生産性を向上させるのに役に立っている，あるいは役に立つのであろうか．本稿では，上記の問題に対して，現時点[1]での私なりの考えを述べてみたい．このような新しい概念を持ったパラダイムは，ソフトウェア開発プロセス全体を支配する考えとなり得る，つまり，ソフトウェア開発の各段階にオブジェクト指向概念が取り込まれていくことにより，開発プロセスも変化していく可能性がある．本稿では，まず，各開発段階で現状でどのようにオブジェクト指向概念が導入されているかから見ていきたい．

2 分析・設計段階

伝統的なウォータフォール型のモデルに従えば，ソフトウェア開発には，まず要求の仕様化を行い，それに基づいて設計を行う段階がある．これらの段階は，ソフトウェア開発のライフサイクル中で先頭に位置するため，これらの段階で行う作業の質が，ソフトウェアの開発効率や開発されたソフトウェアの品質に大きな影響を及ぼす．それゆえ，この段階へのオブジェクト指向的な考え方の導入は盛んである．現在までに種々のオブジェクト指向分析・設計法が提案されているが [1]，それらのすべてがオブジェクトの識別作業から始めるように支援している．仕様化・設計プロセスは，人的要因にかなりの部分を依存しているため，方法論を用いたとしても，人間が行なわなければならない高度な知的作業はプロセス中に残されている．このような作業は方法論の教科書には記載されておらず，それが必ずしも教科書どおりに開発作業が進められない原因にもなっている．

オブジェクト指向分析・設計法は，従来の Structured Analysis や Structured Design に代表される機能分解に基づく手法や状態遷移図などのシステムの振舞いに基づく手法とは異なり，まず「具体的なもの」を抽出し，それを抽象化の単位にしていくため，最初の作業としてはとりかかりやすいと思われる．しかし，最初の作業が難しいという方法論の常として，オブジェクトの抽出作業が他の作業に比べて難しいことには変わりがない．通信ソフトウェアのように，抽出すべきオブジェクトが比較的はっきりしている分野によっては，最初の作業は軽減されるかもしれない．その他，実際の開発事例を調査し，オブジェクト指向分析・設計法の利点，問題点を分析した研究もいくつか報告されている [2]．オブジェクト指向分析・設計法はまだ整備されて間もないため，本当の評価は今後の適用事例を待つ必要があろう．

国際標準化などの動きと合わせて，形式的仕様記述法の実際的な適用が盛んに行なわれるようになってきた．しかし，形式的仕様記述言語を用いてどのように対象世界のモデルを作り上げ，仕様を書いていけばよいのかが問題である．形式的仕様記述言語をオブジェクト指向的手法と組み合わせて使用し，成功した例がいくつか報告されている [3]．形式的手法と融合できる可能性があることも利点の一つとしてあげられよう．

3 コーディング

SmallTalk80[4] に代表されるオブジェクト指向型言語は，メッセージパッシングという計算メカニズムを変えただけでなく，インクリメンタルプログラミングというプログラミングの新しいスタイルをも確立した．また，

[1] 1993年12月15日伊豆河津にて

表 1: オブジェクトの概念とソフトウェア開発段階

	分析・設計	コーディング	テスト・デバッグ	プロセス
クラス抽象	○	○	?	△
多態	△	△	○	○
継承	△	○	?	△

その豊富なクラスライブラリと可視性のよいブラウザにより，エンドユーザコンピューティングへの道も開いてきた．それに対し，C++などの既存の言語にオブジェクト指向の概念（の一部）を取り込み，既存のソフトウェア資産との互換性を保つようなオブジェクト指向言語もある．これらオブジェクト指向言語を用いると，コーディング時に大域変数の扱いが簡単になるという利点があげられている．

4 その他の段階，プロセス全体

オブジェクト指向は，コーディング段階までのいわゆる上流工程に浸透してきた．テスト・デバッグ段階や保守段階での工学的手法はいずれも手続き型言語をターゲットとしたものが主流である．オブジェクト指向に基づいた仕様やプログラムに対する手法（例えばオブジェクト指向プログラムの Slicing 法など）が考え得るはずであり，これらの確立が期待される．

ソフトウェア開発中には，種々のドキュメント（コードも含む）が生成され，それらが構造を変えながら進化していく．これらのドキュメントを一括して蓄積，管理するためにオブジェクト指向データベースの導入の研究開発が進められている．このような互いに関連を持ち，進化していくものを扱うには，オブジェクト指向の考えは有用であろう．

5 役に立つか？

オブジェクトが持っている特徴として，Rumbaugh らは 1) クラス概念，2) 多態 (Polymorhisim)，3) 継承 (Inheritance) をあげている [5]．各段階でこれらの概念のうちのどれがもっとも貢献しているかを私なりの独断と偏見でまとめたのが，表 1である．

ソフトウェア開発は，人間集団の社会活動である．関与する人間も顧客，ユーザ，設計者，プログラマ，マネジャーと様々であり，また所属している組織やその環境も様々である．従って，オブジェクト指向の適用効果もこれらの社会活動，社会環境といった要因を加味して論じなければならない．例えば，方法論の使用に関する組織の方針や方法論の教育体制などが大きな影響を及ぼすと考えられる．従って，オブジェクト指向の概念が十分に教育されて，行き届いてからそのソフトウェア開発に対する効用を議論すべきであろう．現在の状況では，オブジェクト指向概念に対する教育・訓練は必ずしも旧来の手法（例えば構造化手法やそれに基づく支援ツールなど）に比べて十分とは言えない．情報処理技術者試験にもオブジェクト指向概念があまり出題されていないことからも教育の動機づけの低さが伺われる．オブジェクト指向が現状のソフトウェアの生産性向上の役に立つかあるいは立ったかは，これらの社会的な条件，前節で述べた技術的条件が旧来の手法と同じになって始めて評価できるのではないだろうか．

参考文献

[1] 本位田真一．オブジェクト指向分析・設計総論．オブジェクト指向分析・設計チュートリアル，pp. 1–16. 情報処理学会，1993.

[2] M. Aksit and L. Bergmans. Obstacles in Object-Oriented Software Development. Technical Report Memoranda Informatica 92-15, Unversiteit Twente, 1992.

[3] S. Gehart, D. Craigen, and T. Ralston. Observations on Industrial Practice Using Formal Methods. In *Proc. of 15th ICSE*, pp. 24–33, 1993.

[4] A. Goldberg and R. Robson. *SMALLTALK-80: The Language and Its Implementation.* Addison Wesley, 1983.

[5] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, and W. Lonrensen. *Object-Oriented Modeling and Design.* Prentice-Hall, 1991.

合同 ソフトウェア工学 96-7-3
プログラミング－言語・基礎・実践－ 15-6-3
（1994. 1. 20）

オブジェクト指向は本当に役に立っているのか

久世 和資
日本アイ・ビー・エム 東京基礎研究所

はじめに

オブジェクト指向は、ソフトウェア開発に着実に導入されてきている。しかし、その本来の能力は、十分に発揮されていない。その原因となっている問題点や技術課題を議論する。

基本技術

オブジェクト指向プログラムは、オブジェクトと呼ばれる単位を中心に作成される。オブジェクトは、データ群とそれらに関連する手続き（メソッド）群を一体化したものである。複数のオブジェクト間のメッセージ通信（メソッド呼び出し）で、プログラムが実行される。オブジェクトの雛型として利用されるのがクラスである。継承は、このクラスを系統的に構築するのに使う。

オブジェクト指向を実践するための基本技術として、まず、プログラム記述のための言語が必要である。これまでに数多くのオブジェクト指向プログラム言語が実現されている。その中でもC++とSmalltalkが、最も広く利用されている。

次にクラスライブラリが重要となる。クラスライブラリは、ある分野に有効なクラスの集まりで、各クラスは一般に継承を用いて関連付けられている。ライブラリとしては、Xウィンドウ用のInterViewsなどがある。

クラスライブラリの再利用性をさらに高めたものが、フレームワークである。フレームワークは、対象とするアプリケーション分野に有効な問題解決の枠組である。代表的なものに、MacAppやUnidrawなどがある。

また、ソフトウェア開発の上流工程においては、オブジェクト指向分析や設計が、基本技術として重要である。

支援ツール

オブジェクト指向の基本技術に対して、それらを側面から支援するツール群がある（図1）。まず、C++やSmalltalkなどの言語に対して、コンパイラやインタープリタなどの言語処理系がある。言語によるプログラムの記述やクラスライブラリの作成および理解には、クラスブラウザーなどが利用される。作成したプログラムのデバッグや実行効率のチューニングには、オブジェクト指向特有の機能を持つデバッガーやプロファイラーが有効である。

フレームワークを支援するツールとしては、アプリケーション・ビルダーがある。これは、ソフトウェアの部品を、視覚的かつ対話的に組み合わせて、アプリケーションを構築するツールである。アプリケーション・ビルダーには、オブジェクト指向でないものや、オブジェクト指向でもフレームワークを持たないものもある。オブジェクト指向をベースにしたアプリケーション・ビルダーには、PARTSなどがある。

オブジェクト指向分析と設計では、各手法ごとにツールが用意されている。複数の手法を同時に支援するツールも開発されている。

技術課題

基本技術の分類に従って問題点や課題を述べる。

1）言語

C++などのコンパイル型の言語では、コンパイルやリンクの時間が増えて、プログラムの修正時の

言語	←	コンパイラ
	←	インタプリタ
	←	デバッガー
ライブラリ	←	ブラウザー
フレームワーク	←	アプリケーション・ビルダー
分析・設計	←	分析・設計ツール

図1　オブジェクト指向の基本技術とツール

サイクルが大きくなっている。継承などの導入によりプログラムの各部分の依存関係が、非オブジェクト指向言語に比べて、増えていることが一つの原因である。特にC++では、プライベートな変数や関数までクラス定義に現れるため、局所的な変更でも、そのクラスを利用しているファイルすべてを再コンパイルすることになる。

また、クラスライブラリの利用時にも、コンパイル時間が問題になる。アプリケーション自体が小さくても、利用しているライブラリの膨大なヘッダーファイルをコンパイルする必要があるためである。

オブジェクト指向の言語機能を処理するために、言語処理系やリンカー、また、ビルド・プロセス全体が複雑になっている。たとえば、C++のテンプレートの処理もコンパイラやリンカーに大きな負担となっている。

オブジェクト指向プログラムのデバッグも、手続き型の言語のデバッグに比べて一般に難しくなっている。これは、プログラムの実行時に動的に決まるものが多くなっているためである。実行時に、オブジェクトのネットワークができ、その間でメッセージをやりとりし、処理が進む。この実行時のオブジェクトを中心にしたデバッグ手法やツールが必要である。

また、オブジェクトを動的に作成したり消去したりするためメモリー管理が煩雑になっている。C++のプログラムの実行時エラーは、メモリー管理に関係することが多い。

オブジェクト指向プログラムの実行時の効率も問題となってきている。コンパイラによる、よりオブジェクト指向の言語特性を考慮した最適化が必要となる。メモリー使用量などのチューニングに適したプロファイラーも有効である。

2）ライブラリ

クラスライブラリの利用は、プログラムの生産性を向上させるが、より有効利用するためにはツールによる支援が不可欠である。実用のライブラリでは、クラスが数百の単位で含まれる。その中から最適な部品を短時間で検索したり、そのクラスの機能を理解するには、クラス間の類似性の利用やドキュメントとの結合といった機能を持ったツールが必要になる。

また、再利用性の高いクラスライブラリを構築するための手法や技術が不十分である。これに関連してライブラリの品質の評価基準の確立や評価ツールが重要になってくる。

3）フレームワーク

ユーザーインタフェースに近い部分では有効なフレームワークがいくつかあるが、これからは、より特定のアプリケーション分野に有効なフレームワークの開発が必要になる。さらに、フレームワークを有効活用するツールの研究、開発も望まれる。

4）分析・設計

手法やツールは、確立されてきているが、実際の適用事例が少ない。我々は、対象がプログラミング環境ということもあり、分析や設計手法は、ほとんど使用していない。外部仕様と上記のフレームワークの設計を中心に開発している。仕様の変更が頻繁なので、それに柔軟に対応できることを重視してフレームワークを設計している。オブジェクト指向分析や設計を、正しく使用するためには、かなりの学習時間を要する。

また、アプリケーションの新規開発には、オブジェクト指向を導入し易いが、既存のソフトウェアをオブジェクト指向に移行するには、まだ、多くの課題が残されている。

おわりに

ここで紹介できた技術課題は一部である。オブジェクト指向技術は、部分的に活用されているのが現状である。大規模なソフトウェア開発に本格的に利用するためには、分析・設計からテストや保守にいたるソフトウェア開発全般で統一的な技術基盤が必要である。

参考文献

久世：オブジェクト指向プログラミングの利用価値、情報処理、Vol.34、No.1、pp.2-10 (1993)
(pp.9-10に関連する参考文献)

合同 ソフトウェア工学 96-7-4
プログラミング－言語・基礎・実践－ 15-6-4
（1994．1．20）

オブジェクト指向は本当に役に立っているのか
－ソフトウェア工学の立場およびソフトウェア科学の立場から－

加藤 和彦
筑波大学 電子・情報工学系

1．はじめに

ソフトウェア構築の方法論として「構造化プログラミング」をはじめとしていくつもの標語が提唱され，おびただしい議論がこれまでになされてきた．方法論は標語化されることによって議論の俎上に上がるようになり，多くの人々に浸透していくようになる．また各方法論の実践を支援するために，プログラミング言語処理系などさまざまなソフトウェア開発ツールが提供されている．それらのツールが推奨するやり方でソフトウェア作りを行えば，自然とその方法論を実践できるという仕掛けである．

そもそも「オブジェクト指向」という標語には，計算機ソフトウェアのさまざまな分野で議論され，蓄積されてきたいくつかの手法をまとめて総括的な名前を与えたものであるという側面がある．このため「オブジェクト指向」方法論が主張している内容は，現在のソフトウェア科学およびソフトウェア工学の分野において，その有用性の評価はある程度定まっているものが多い．にもかかわらず今回のようなテーマのパネルディスカッションが組まれた背景には，「オブジェクト指向」に限らず，一般にソフトウェア開発のための方法論の有用性の本質を定量的に論ずることが困難であることがあろう．

本稿では，「オブジェクト指向」を用いるということはどういうことなのかを考えることから始めて，オブジェクト指向を「役に立たせる」可能性を探ってみたい．

2．オブジェクト指向の使い方

あるソフトウェアシステムを開発するにあたってオブジェクト指向の概念を利用しようと考えたとき，オブジェクト指向の使い方が一意に定まるわけではない．「オブジェクト指向」の使い方には少なくとも次の３つのレベルが存在すると考えられる．

(a) ソフトウェアシステムの実装をオブジェクト指向言語を用いて行う．
(b) ソフトウェアシステムの構成要素間のインターフェースがオブジェクト指向の概念に基づいている．
(c) ソフトウェアシステムのアーキテクチャ自体がオブジェクト指向の概念に基づいている．

(a)はもっともミクロな利用法であって，オブジェクト指向の考え方に基づいて設計されたプログラミング言語を用いてソフトウェアシステムを実装するというやり方である．例えばあるソフトウェアシステムを実装するときに実装言語としてC++やSmalltalk-80などのオブジェクト指向言語を採用する場合がこれに相当する．オブジェクト指向言語を使用はしないが，オブジェクト指向言語で書いたかのようなプログラミング作法をとる場合も，(a)に含めることにしよう．

実装言語がオブジェクト指向であるからといって，ソフトウェアシステムそのものがオブジェクト指向的である必要は必ずしもない．例えばあるデータベースシステムがオブジェクト指向言語を用いて実装されたからといって，そのデータベースシステムがオブジェクト指向的になっているとは限らない．オブジェクト指向言語を活かした実装を行うことにより，実装対象であるソフトウェアシステム内の情報構造がオブジェクト指向の方法論でモデル化される．(a)の場合に「オブジェクト指向」が役に立ったとすると，それはこのソフトウェアシステム内の情報構造のモデル化にオブジェクト指向に適していたか，あるいは採用したオブジェクト指向言語が提供する機能がモデルの表現もしくはモデルの実装に役に立つものであったかのいずれか，もしくは両方である．

(b)は各ソフトウェアコンポーネントをオブジェクトと見立て，一定のインターフェースを満たす

オブジェクトを組み合わせることによって一つのシステムを構成するという方法である．この方法の一つの典型的な利用例は，実装者がたくさんのコンポーネントを提供しておき，ユーザが利用する際に必要なコンポーネントを取捨選択することで，半オーダーメードのシステムを構成するというやり方である．今一つの典型的な利用例は，異種の計算機ハードウェアやオペレーティングシステムにまたがって構築された分散オブジェクトシステムである．これら２つの例は外部モジュールに対して開かれたシステムという意味でオープンシステムと称されることがある．オブジェクト指向の方法論は，個々のコンポーネントの内部を隠蔽しつつ，外部インターフェースを合わせてコンポーネントを組み合わせるオープンシステムの構築に誠に都合がよい．(b)の場合，各コンポーネント内の実装が(a)のようにオブジェクト指向言語を用いて行われていてもよいし，そうでなくとも構わない．

(c)はある一つのシステムのアーキテクチャそのものがオブジェクト指向の枠組みに基づいて設計が行われている場合である．オブジェクト指向データベースシステム，オブジェクト指向オペレーティングシステム，そして多くのグラフィカル・ユーザインターフェース・システムがこれに相当する．これらのシステムの実装にはオブジェクト指向言語が採用されることが多い．

3．オブジェクト指向的なもの

筆者は日頃，オペレーティングシステムや分散システムの分野で研究を行っている．これらの分野を例としてオブジェクト指向の有効性について考えてみよう．

UNIXは通常オブジェクト指向オペレーティングシステムとは言われないが，その構造をよくみてみると，ファイルシステムを中心に「オブジェクト指向的なもの」をみつけることができ，しかもその「オブジェクト指向的なもの」がUNIXが使いやすいと言われる大きな要因になっている．Machになると，オブジェクト指向の度合いは更に進んでおり，その基本的アーキテクチャはオブジェクト指向そのものといってよい．UNIXにせよMachにせよ，オブジェクト指向ありきで始まったというよりも，その設計と開発の過程で自然にオブジェクト指向的なものが実践されたように思える．ではこれらの例では，どうして自然とオブジェクト指向的なものが実践されたのであろうか．それはオブジェクト指向的なものがうまく働く典型として，「複雑性の制御」と「異種性の標準化」があるからではないかと筆者は考えている．

最近の実用的ソフトウェアは巨大化の一途を辿っている．ソフトウェアが巨大化しても破綻を来さないようにするためにはソフトウェアの複雑性を制御することが非常に重要である．オブジェクト指向は，モジュール化の概念を進化させたものであり，複雑性の制御に対しオブジェクト指向の考え方は非常に有効に働く．オペレーティングシステムは従来から巨大ソフトウェアの典型的な例とされているが，monolithicな構造のUNIXカーネルが破綻を来すことなく，ここまで肥大化しつつもやってこれた要因として，UNIXのカーネル構造の中の「オブジェクト指向的なもの」が大きな役割を果たしているように思える．

量的な面以外に，ソフトウェアの複雑性を増加させる要因は異種性の取り扱いである．オペレーティングシステムの最も重要な機能の一つは計算機資源の仮想化と抽象化である．この機能がさまざまな計算機資源の異種性を吸収してくれるおかげで，個々のアプリケーションプログラムは，資源の異種性に煩わされることなく資源を利用できる．UNIXとMachはそれぞれ別のやり方で計算機資源の仮想化と抽象化を行っているが（前者はシステムコール，後者はメッセージパッシング），それらがやっていることの本質は，計算機資源のオブジェクト指向的なモデリングに基づいた仮想化と抽象化であると理解できる．

4．おわりに

オブジェクト指向の方法論を理解することは，ソフトウェアの研究および開発に携わる現代の人々にとって必須になったと言ってよいであろう．オブジェクト指向の方法論は明らかにいくつかの有用な概念を含んでおり，その有用な概念の活かし方にはいくつもの方法がある．また風雪に耐えたソフトウェアをよく観察してみると，「オブジェクト指向的なもの」を見てとることができる．「オブジェクト指向は役に立つか」というよりも「オブジェクト指向を役立てられるか」と考えてみるとよいかもしれない．