情報処理学会
オブジェクト指向'95シンポジウム論文集
pp.327-329, 1995年6月

「オブジェクト指向'95シンポジウム」平成7年6月

オブジェクト指向技術へのパラダイムシフト
- Why > What > How -

中所武司
明治大学理工学部情報科学科
email : chusho@cs.meiji.ac.jp

Abstract

最近の情報システム構築に関する大きな変化は，分散コンピューティングとエンドユーザコンピューティングの動きの中でのオープンシステムへのパラダイムシフトである．それに対応したソフトウェアの作りやすさと使いやすさを実現する新しいソフトウェアパラダイムとして，オブジェクト指向技術が重要な役割を果たしつつある．

しかし，このようなソフトウェア工学分野の技術移転の難しさは，問題認識の空間的ギャップと時間的ギャップの克服である．理論／研究開発の立場からも実践／適用の立場からも，Why（問題）→What（解決案）→How（実現方式）の流れの中で，常にユーザの視点から，いま何が問題かを認識しておく必要がある．

1．はじめに

今回のテーマ「オブジェクト指向によるシステム開発の理論と実践」に関して，まず2章でオブジェクト指向技術に関する見解を述べ，3章で新技術の実用化の一般的な困難性について述べ，4章で実用化シナリオについて述べる．

2．オブジェクト指向技術

2.1 ソフトウェア動向（Why）

最近の情報システム構築に関する大きな変化は，システムのグローバル化に伴うオープンシステムへのパラダイムシフトである．システム化の目的が業務の合理化から情報の資源化へ拡大し，システムのインフラがホスト集中から分散コンピューティングへ移行し，システム化の主体がシステム部門からエンドユーザ部門になってきた．

オープンシステムでは，このような分散コンピューティングとエンドユーザコンピューティングの中で，異機種コンピュータ間でのアプリケーションソフトウェアの接続性や移行性の要求に応えるために，アプリケーションソフトウェアのアーキテクチャをできるだけ標準的に作る努力がなされており，以下のようなソフトウェア構成（図1）[2]をとる．

- 階層化：ハード，基本ソフト，ミドルウェア，応用ソフトウェアの階層構造化．
- 部品化：各階層を部品の集合で構成．
- 統一インタフェース化：階層間のインタフェースの統一．

このような動向の中で，ソフトウェアの作りやすさとソフトウェアの使いやすさを実現する新し

A Shift in Paradigm to Object-Oriented Technology
- Why > What > How -,
Takeshi CHUSHO,
Meiji University, Department of Computer Science

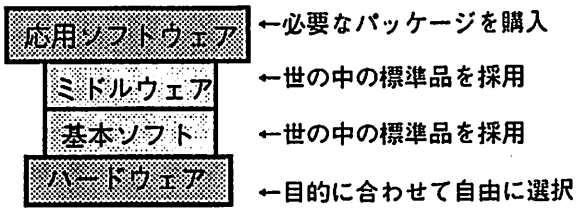

図1　オープンシステムのソフトウェア構成

いソフトウェアパラダイムとして，オブジェクト指向技術が注目され，既にプログラミングの分野では実用的効果[1]をあげている．筆者は，オブジェクト指向のもつ”わかりやすさ”がシステムのグローバル化にうまくマッチした結果と考える．

2.2 オブジェクト指向の概念（What）

オブジェクト指向概念の基本的な要件として，以下の4項目をとりあげる．

●分散協調型計算モデル
●データ抽象化機能
●クラスからのインスタンス生成機能
●クラスの階層化と継承機能

これらは各々独立した概念であるが，前の概念を前提に後の概念が定義されるので，前のものほど本質的である．そこで，筆者はオブジェクト指向概念に関して以下の立場をとる．

・機能指向とデータ指向の両者を統合したもの．
・計算モデルの観点をデータモデルよりも重視．

オブジェクト指向の実用性について議論するときには，どの概念がどのようなソフトウェアの問題解決に役立つかということを明確にする必要がある．

2.3 オブジェクト指向の応用（How）

ソフトウェア開発のどの局面に注目するかによって，適用すべき概念とその効果[3]が決まる．

例えば，ＯＯＰの場合は，以下の効果が期待される．

●開発者のための部品化，再利用
・データ抽象化により，独立性の強い，汎用的な機能部品を作りやすい．
・階層化と継承機能により，部品ライブラリを構築しやすい．
・インスタンス生成機能により，部品のカストマイズがしやすい．
・分散協調型モデルに基づくプログラム設計により，部品の抽出，利用が容易である．

●保守者のための拡張性と移行性
・階層化と継承機能により，機能追加が容易である．
・データ抽象化により，オブジェクトの外部インタフェースと内部構造を完全に分離でき，機能変更や他機種への移行が容易である．

●利用者のための操作性と統一性
・データ抽象化とメッセージ駆動型制御方式により，画面上のオブジェクトが持っている機能の一つを選ぶという直接操作で簡単に対話できる．
・種々の応用ソフトウェアのユーザインタフェースは共通部品での構築により，統一できる．

ＯＯＡ／ＯＯＤの場合は，要求分析における問題領域（ドメイン）のモデリングおよび設計におけるドメインモデルから計算モデルへの変換の容易化が期待される．もちろん，モデリング技術としては，オブジェクト指向のモデルが適している分野とそうでない分野があることになる．

3．実用化の困難性

3.1 問題認識の空間的ギャップ

ソフトウェア工学の分野の技術移転の難しさがしばしば指摘されるが，その原因の一つは，研究におけるユーザの視点の欠如ではないだろうか．筆者がよく使う以下の図式で言えば，技術移転と逆方向の問題認識の移転が重要である．

●問題認識の移転：

ソフトウェア科学 ← ソフトウェア工学 ← ソフトウェア産業 ← エンドユーザ

●技術移転：

ソフトウェア科学 → ソフトウェア工学 → ソフトウェア産業 → エンドユーザ

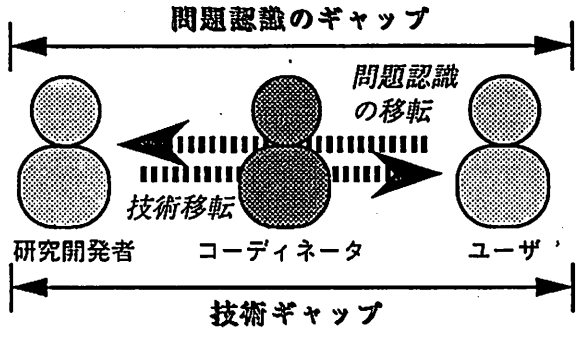

図2　問題認識の空間的ギャップ

大規模ソフトウェア開発技術については，特に研究開発者がユーザの問題を認識することが難しいため，図２[5]に示すようなコーディネータの存在が不可欠である．コーディネータはユーザ側のサポート部門の担当者がふさわしく，技術移転の段階では伝道者の役割を果たす．

一方，フリーウェアなどでよく利用されているプログラミングツールは，研究開発者自身がユーザであることが多く，この問題はない．

3.2 問題認識の時間的ギャップ

研究開発の時間的流れは，一般的には，

Ｗｈｙ → Ｗｈａｔ → Ｈｏｗ
（問題）　（解決案）　（実現方式）

となるが，新しいコンセプト（解決案）が提示されてから実用になるまでに１０年位かかるものが多く，その間に問題そのものが変化していることも多い．オブジェクト指向技術の関連でＯＯＰとＯＯＡ／ＯＯＤを例にとって見ると図３のようなことが起こりうる．

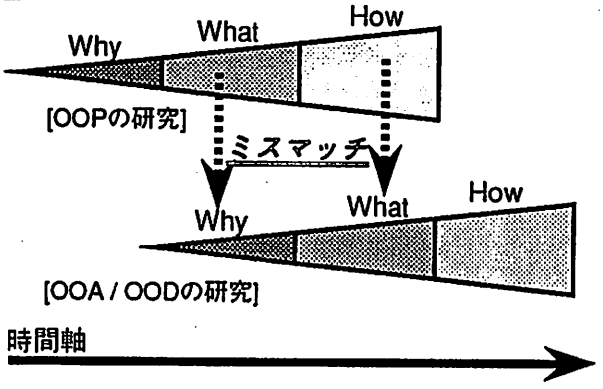

図３　問題認識の時間的ギャップ

解決すべき問題が変化すれば，対応するパラダイムもシフトする．古いパラダイムの下での研究成果を新しいパラダイムに適用すればミスマッチが生じる．「未来学 やってるつもりが考古学」とならないように，常にユーザの視点に立って，いま何が問題かを認識しておく必要がある．

４．実用化のシナリオ

オブジェクト指向技術の効果を議論する前に，解決したい問題がなければ何も始まらない．Why > What > How の観点[4]では以下の手順になる．

(1)何が問題か，何を解決したいかを明確にする．
(2) オブジェクト指向のどの概念がその問題の解決策としてどのように有効かを見極める．
(3) その実現方法を検討する．

個別の議論をするだけの紙面はないが，解決したい問題としては，／開発期間を短縮したい／システムエンジニアの生産性を上げたい／再利用率を高めたい／開発をエンドユーザ主体でやりたい／保守はエンドユーザだけでやりたい／拡張性／接続性／移行性／等々，が考えられる．

システム開発技法の観点では，分析・設計に注目すれば，オブジェクト指向をモデリング技術としてとらえ，ドメインモデル，計算モデル，データモデルの視点での適性を検討する必要がある．分析では，domain-specific なアプローチが必須となる．設計では，ソフトウェアアーキテクチャを規定するアプリケーションフレームワークが重要になろう．

プログラミングに注目すれば，カプセル化・部品化の視点でのクラスライブラリ構築，平行処理の視点でのメッセージ処理が重要である．

５．おわりに

アプリケーションシステム開発の立場からも，その開発技法の研究開発の立場からも同一の問題認識を持つことがすべての始まりになる．

ソフトウェアの研究は，他の分野以上にユーザの視点に立って，身近なところから始めて本質に迫るというアプローチが必要[6]ではないだろうか．

参考文献

1) 中所：使いやすいソフトウェアと作りやすいソフトウェアーオブジェクト指向概念とその応用ー,電気学会雑誌, 110, 6, 465-472 (1990).
2) 中所：エンドユーザコンピューティングーソフトウェア危機回避のシナリオー,情報処理, 32, 8, 950-960 (1991).
3) 中所：ソフトウェア危機とプログラミングパラダイム, 啓学出版 (1992).
4) 中所：." わかりやすさ" を追求すればオブジェクト指向になる, CASEジャパン'92セミナー「方法論はどこまでシステム開発を支援してくれるか」(1992).
5) Chusho,T.：What makes software tools successful?, IEEE Software, 10, 5, 63-65 (1993).
6) 中所：CS-life, コンピュータソフトウェア, 11, 6, 1-2 (1994).