ソフトウェア工学 102-10
（1995. 1.26）

ビジネス・アプリケーションを対象とした RAD型オブジェクト指向開発手順の考察

武田和彦、四國 修、廣岡 龍哉、青木弘之

NTTデータ通信

概要

オブジェクト指向システムは、現在の所比較的高機能なアプリケーションの開発事例は良く知られている。 しかしいわゆるビジネス系のアプリケーションの様な比較的単純かつ定型的なプログラム製造に適用したと言う事例はあまり聞かない。

そこでこのような種類のプログラムの製造に於いても、オブジェクト指向の利点が発揮できるかを検証する為、システム分析から実際のインプリメンテーション、デバッグに至るまで、オブジェクト指向の各種技法を取り入れた開発の手順を設定した上で、モデルシステムを用いて試行開発を行う。

さらにオンラインリアルタイムシステムの形態を採っている場合が多い実際のビジネスシステムをどの様な形態でオブジェクト指向システムとして実現すれば良いかの考察を行う。

Consideration on RAD type object oriented application development method targeted for business domain.

Kazuhiko Takeda, Osamu Shikoku, Tatsuya Hirooka, Hiroyuki Aoki
NTT Data Communications Systems Corporation

Nowadays, most of the well known applications developed using object oriented programming are relatively complicated applications. Few examples are known to be developed by object oriented programming in the field of relatevely simple business applications.

So we tried to find out the advantages of object oriented technology for such relatevely simple business application development.

And we have practiced and examined object oriented method in each development stage, from domain analysis phase to program debugging phase.

Finally we considered a way for applying object oriented technology to popular online realtime business system.

1　まえがき

ビジネスシステムの開発の各工程で行う開発技法に付いて、オブジェクト指向技術を用いて基本的な技法、機能の確認を行った。

なお開発は一般的な販売在庫管理システムである既存の訓練システムを用いて行い、システム分析書は既に存在したが、開発はなるべく実際の行程に近い形を採って行った。

さらにターゲットシステムは、今後ビジネスシステムの分野でも主要な形態と成るであろう、ワークステーションを使った分散システムの形態にし、開発そのものも分散方式で開発を行った。

開発期間は３ヵ月／２人で行い、分析段階ではさらに複数が加わった。

2　開発内容の検討

検討内容は開発方式に関するものと、実装方式に関するものにしぼって検討した。

（1）　開発方式

開発手順の基本は既存のウォーターフォール技法を使うようにと言う要求が強かったが、既存の方式に囚われずに、プロトタイプを用いてリカーシブに開発すると言う通常のオブジェクト指向プログラム開発の方法を定型化して行った。

オブジェクトの抽出には簡単にかつ形式的に出来る様に名詞を用いた。

オブジェクト分析の初期の段階では、エンドユーザも含めて大勢で検討出来るようCRCカードを使った方法で分析を行い、後にこれをCOADのオブジェクト図[1]にオブジェクト図エディタObjectCast Light[2]を使って描き直した。

オブジェクトの動的振る舞いの記述にはADM3のCFD図[3]を手書きで用いたが、ビジネスシステムは複雑なシステムではないのでSTD等は使わないことにした。[4]

（2）　実装方式

開発期間が短いので実環境に殆ど左右されないほど抽象性の高いSmalltalk言語を用いてプログラムし、開発においてネックになると思われるGUIのインプリメントはVisualWorks Smalltalkに標準装備のGUIビルダを用いて自動生成した。[5]

実装方法の点では、ビジネスアプリケーションで最も重要な原簿ファイルをどう扱うかが問題になった。原簿ファイルそのものをオブジェクト化する方式も考えたが、新たにオブジェクト分析して原簿項目に相当するオブジェクトが分散するに任せた。これはODBMSの存在を前提にしたからでも有る。しかし、実際のインプリメントでは全てのオブジェクトをはSmalltalkのイメージファイルとして固定するに止めた。

3　個別の試行項目の検討

実装方式の検討事項を図1～6に示す。

4　試行開発結果の検討

OOA

名詞によるオブジェクトの抽出はシステム概要さえ不足なく記述されていれば、機械的に行えて有効な方式だった。

CRCカード技法を用いたオブジェクト分析では活発な意見が多く出て、グループディスカッションには有効な技法であった。そしてよりオブジェクト指向的なシステムに仕上げる為に、基本構造の組直しを何回も行った。

OOD

静的なオブジェクト構造図は基本図面として全体構造の把握に有効だったが図がおおきく成りすぎる傾向が有った、これはプログラムの変更に同期して修正した。

動的なCFD図はデバッグ時にシナリオとして用いると有効で有ったが、枚数が多く成りすぎる傾向が有った。これは手書きラフスケッチの為もあって完全な修正は行わなかった。

オブジェクト図CFDはプログラム時二度手間に成るので基本的な属性、メソッドを記述するに止めた。

またSmalltalk言語の場合はC++の様にオブジェクトのdeleteを行わないので、オブジェクト分析で論じられているような、単なる関係以外の細かな関係等の記述の必要性はあまり感じなかった。

OOP

オブジェクト設計はウインドウ設計を手がかりに行う予定だったが、試行の結果Smalltalkの場合実環境への依存度がきわめて小さい事もあって、抽象的なオブジェクト設計から始めたほうが良い事が分かった。またウインドウのフォーマット設計の為には、オブジェクト構造に付いての情報が必要だと言うことも分かった。

さらにC++の場合はウインドウ関連のクラスが特殊関係構造を持っている場合が多く、また言語が型を持っているので、クラス階層やオブジェクト関係をあまり自由に動かせない。

しかしSmalltalkでは全てのクラスがクラスObjectから継承している事もあって、全てのオブジェクトはほぼ同等に扱う事が出来る。さらにSmalltalkの場合はクラスライブラリが豊富な事も手伝ってシミュレーション言語だから当然とも言えるが、設計時に実環境をあまり考慮する必要が無いと言えるだろう。

この特性は実環境をあまり考慮しないビジネスアプリケーションの記述には適していると言えるだろう。

しかしSmalltalkは開発環境として開発された為、自由度が高すぎて何でも出来るのでエンドユーザの利用に供するには操作性と安全のため、メソッドをオーバライドするなりして余分な機能を殺す必要が有るだろう。

オブジェクト指向分散開発

原簿アイテムに相当するオブジェクトをインタフェースにして、業務単位に分散開発を行った。処理的な問題点は無かったが、分散開発機構が無いので分散した開発環境の同期に手間が掛かった。

実際の開発では、ばらばらになったこれらのオブジェクトをどう管理するかが最も問題になるだろう。

デバッグ

特にSmalltalk言語で実装したので、インプリメント途中で実行して見る事が可能で、適時に実際に動かしてレビューすることも出来た。しかしSmalltalkの場合は実行時メッセージチェックなので完全なデバッグには相当時間が掛かると思われる。

5 感想

試行結果の項でSmalltalkの場合はと言うキーワードを多用したが、同じオブジェクト指向言語で有っても、C++とSmalltalkでは大分趣が異なっていると言える。一般的に言って良く使われる様な機能はSmalltalkで記述した方が楽である。しかし良く使わない特殊な機能で枯れていない機能、特に特殊なインタフェース系のクラスはどうしてもバグが残っていて完全な修正も困難なのでC++でプログラムした方が良いかもしてない。

ビジネスシステムではODBMSを介して適材適所でSmalltalkやC++等をハイブリッドで使うなりすべきであろう。

6. 参考文献

[1] Object-Oriented Programming
Peter Coad and Jill Nicola Prentice-Hall International, Inc. 1993

[2] ObjectCast Light Ver 1.0 ユーザーズガイド
富士ゼロックス情報システム株式会社 1993

[3] Object-Oriented Requirements Analysis and Logical Design
Donald G. Firesmith John Wily & Suns, Inc 1993

[4] 第49回全国大会 講演論文集（5） ビジネスアプリケーションを対象としたRAD型オブジェクト指向開発手順の考察
武田和彦、四國 修、廣岡龍哉、青木弘之 1994

[5] VisualWorks RELEASE 1.0 User's Gide
ParcPlace Systems 1992

◎ オブジェクト指向ビジネスシステムのシステムアーキテクチャの検討

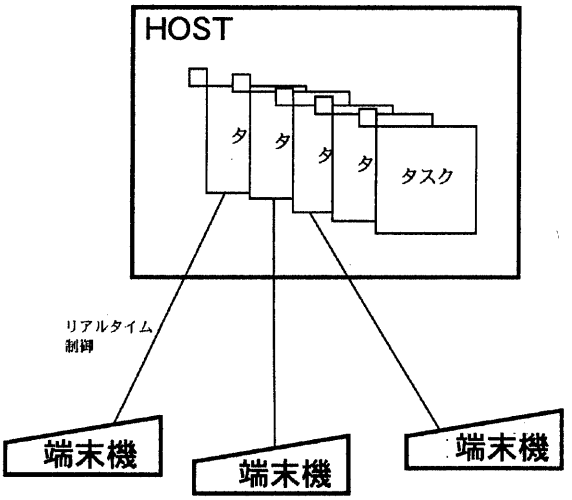


図1　汎用機を使った非オブジェクト指向ビジネスシステム

現在の集中型のビジネスシステムは業務処理ごとに複数タスクで構成されたいわゆるオンラインリアルタイムシステムに成っていることが多い。

しかし現在の所OOPLは単一のプロセス内に閉じた機能しか言語レベルでもクラスでもサポートしておらず、コンカレント処理や通信系の処理が弱い。

↓

それには現在行っているシステム構成法を基本的に見直す必要が有る。
物理構成も分散型と成った。

↓

それにはアプリケーションからリアルタイム性を排除して、コンカレント処理、排他処理、ネットワーク処理等をODBMSに委ねるしかない。
またアプリケーションは全てクライアント側に移して、1ノード1プロセスとする。

↓

この様な形態にすれば1ノード=1プロセスとして、一般のオブジェクト指向アプリケーションの作成と同じ形に成る。ノード間はODBMSでオブジェクトを共有する。
シングルプロセスのシステム構成を考えれば良い事になった。

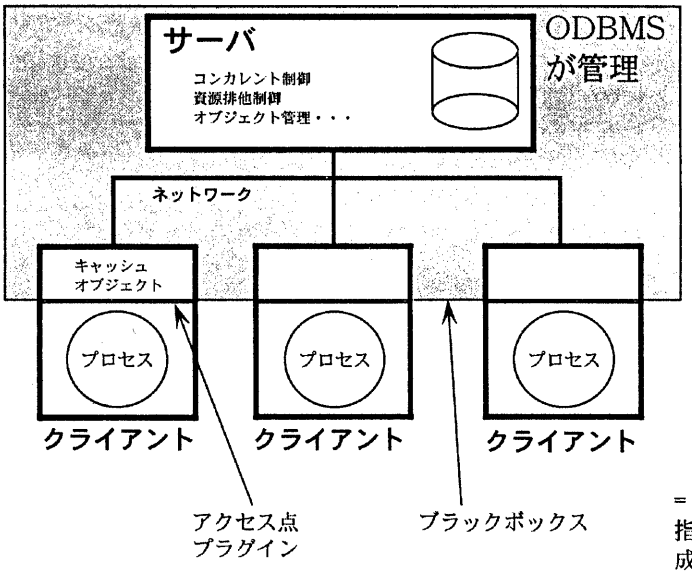


図2　オブジェクト指向ビジネスシステム
原簿オリエンテッドなシステムは殆どがこの形態に成ると考えられる

判定

単純化、分散化、情報の隠蔽と言うオブジェクト指向の考えに合致している。
端末機がインテリジェント化したのでこの様な構成が可能に成った。

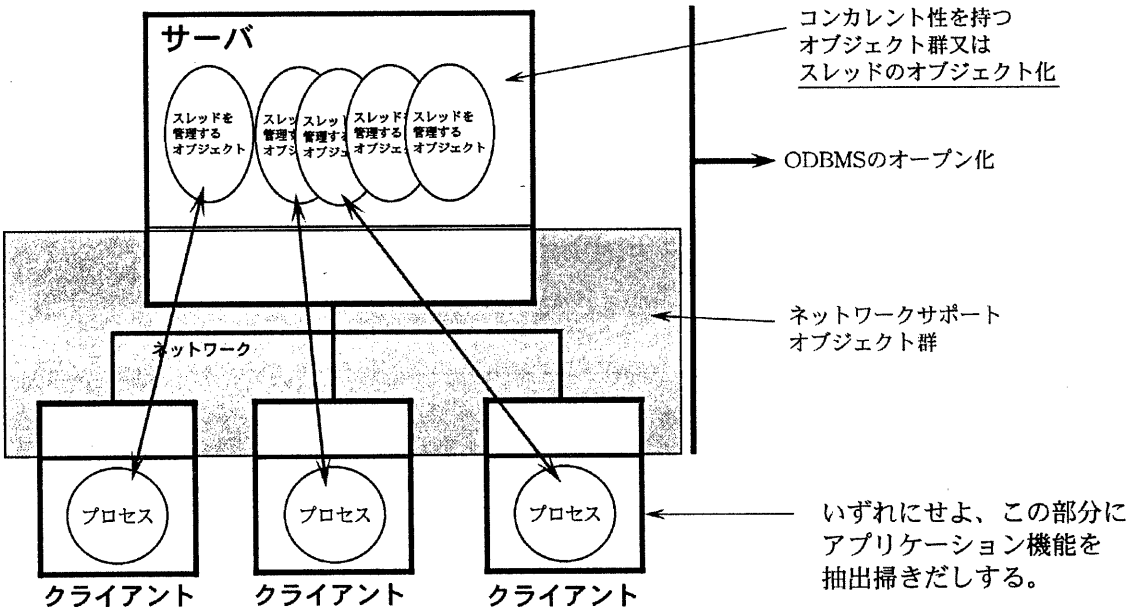
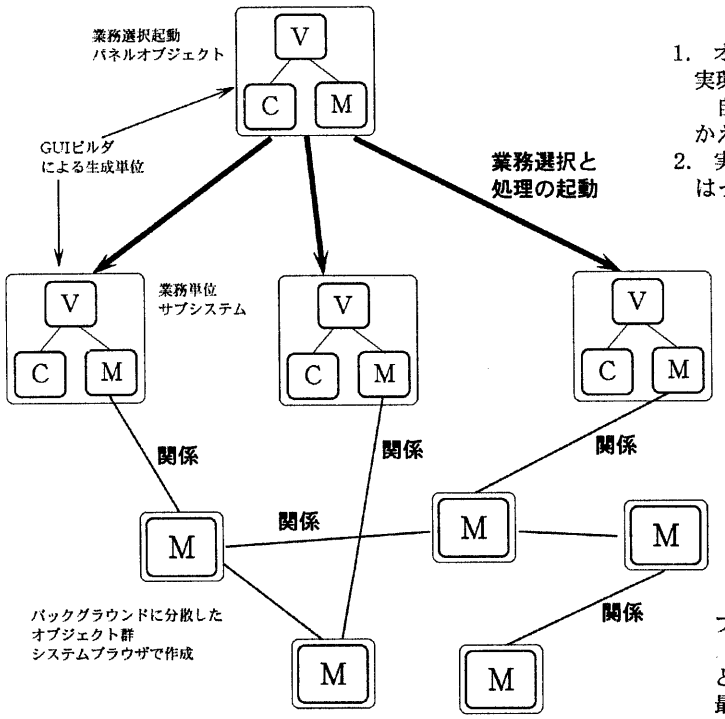


図3　より大規模高トラフィックなシステムでパフォーマンスが要求される
場合又はサーバで固有のプロセスを行いたい場合。
一般的なビジネスシステムではこの様な構成が必要とは思えない。

◎　クライアントアプリケーションの構成を考える



1. オブジェクト分析した後で、それをどの様に実環境にインプリメントするかが問題になった。
自由度が高すぎて何から取りかかればよいかかえって分かりずらい。
2. 実際のビジネスシステムは、業務処理ごとにはっきり別れた形でサブシステム化されている。

そこで便宜的にSmalltalkの基本構成単位であるMVCペアをサブシステムと考える事にした。

（注）SmalltalkではこのMVC構造のみが実環境依存機構と言えるかもしれない。

試作例では全ての業務をSmalltalkプロセスに詰め込んだ。
この構成全体をスケルトン（骨組み）とし、MVCオブジェクトの内部は必要最小限に実装してプロトタイプとした。

図4　クライアントのオブジェクト構成

◎　MVCペアの実装

SmalltalkのView（ウインドウ）の表示には、若干込み入ったプログラムが必要なので開発のネックになると考えられた。そこでViewとViewとModelのインタフェース部分をVisualWorks SmalltalkのGUIビルダを使って自動生成する事にした。

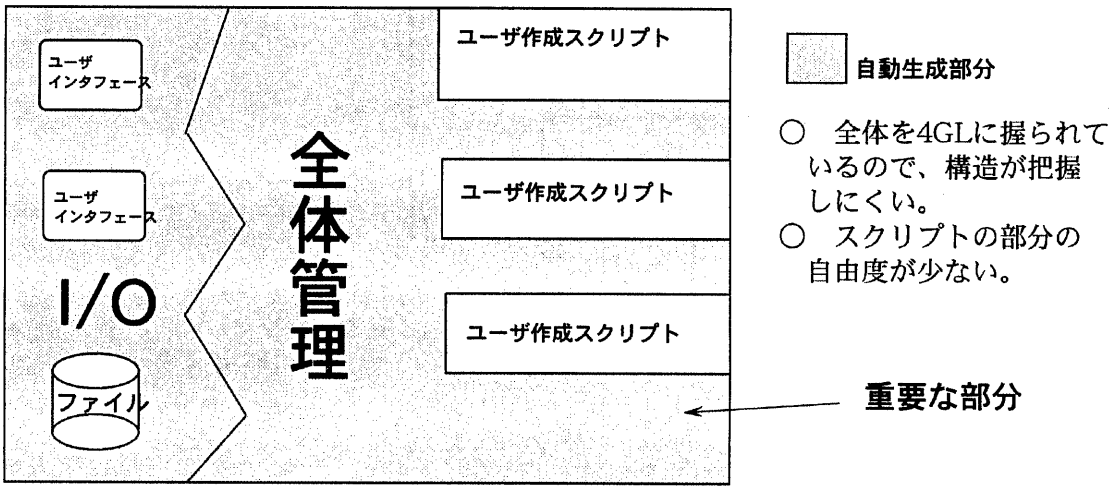


図5　4GLを使ったシステム開発

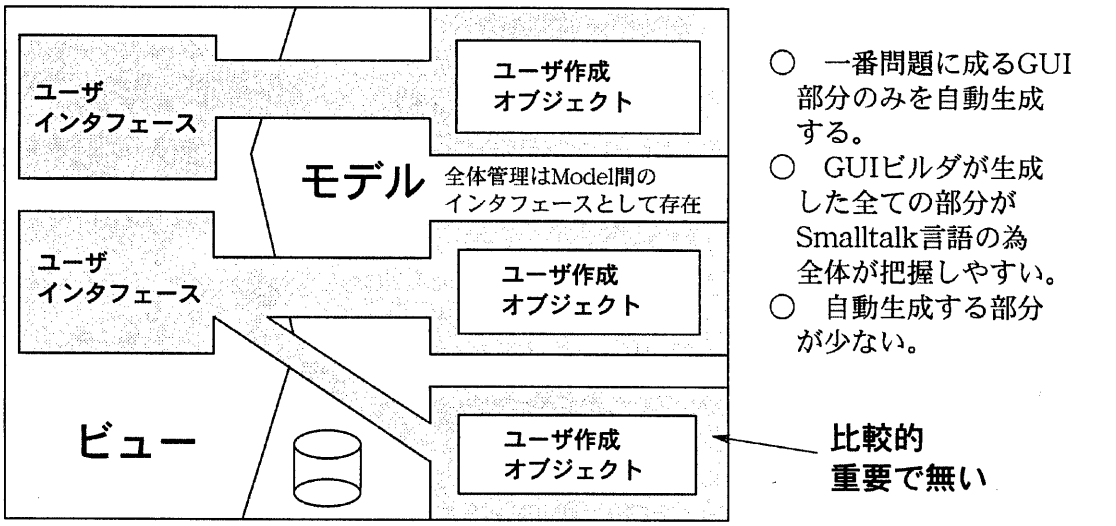


図6　GUIビルダをを使ったオブジェクト指向システム開発

4GLを使った開発と違い、Viewは単にModelをユーザとインタフェースするだけのオブジェクトなので、システムとしては重要ではない。

そしてシステム作成者としては単体としてのView内の機能を把握する必要がないので、Viewを隠蔽された形で自動生成するのは不要な情報の隠蔽と言う観点で見ても良い考えだ。

（注）必要ならViewのみを取り替えるのも可能。