4U－5

ビジネス系システム開発におけるオブジェクト指向開発技法（３）
～再利用性向上のための実装設計～

原田道明，萬木優子，北畠重信，上原憲二
三菱電機（株）情報技術総合研究所

1. はじめに

オブジェクト指向設計によって得られた論理設計と実装設計とのシームレス性、および、データ構造の変更に対する柔軟性を目的とした3階層システム向けの実装アーキテクチャを検討し、実装規約・雛形部品・基本部品の集まりの形で実現した。

本稿では、3階層システムの実装における実装効率・再利用性の観点から課題点を整理し、それらの課題に対する本実装アーキテクチャのアプローチと実際の構成を説明する。

2. 3階層システムの実装上の課題点

3階層システムは画面・業務処理・DBアクセスを独立化したことによって業務処理を効果的に部品化・再利用できるという利点を持っている。一方、普及の進んだ2階層システムと比べて環境整備や実装技法の確立が不十分なために以下のような新しい課題が生じている。

2.1. 適切な実装指針

3階層システムの構築は2階層システムに比べて実装の定型化が進んでおらず、各種の実装要素に対する各オブジェクトの責務配分の捉え方に開発者間でのばらつきが生じる。従って、各種の実装要素に対する適切な実装指針を準備することが重要になる。

2.2. 部品間の自動連携

多くの2階層システム構築ツールはGUI部品・DBアクセス部品の連携により定型的処理の自動化を確立している。一方、3階層システムでは画面とDBアクセスが別ノードに存在するため、従来と異なった部品体系によって連携の自動化を図る必要がある。

2.3.データ構造変更の吸収

データ構造の変更がオブジェクト間のインタフェースに影響すると修正が広範囲に波及する。従ってデータ構造の変更を吸収する工夫が重要になる。

3.本実装アーキテクチャのアプローチ

本実装アーキテクチャでは、開発者が現在利用可能な構築環境の利用を前提として、実装規約と部品の提供によるアーキテクチャの提示を行った。前節で述べた実装上の課題に対して、以下のように解決を図った。

(1) 適切な実装設計指針の提供
トランザクション文脈管理等の実装方式、アプリケーションの類型に応じたオブジェクト構成の指針を設計パターン[1]の形でまとめ、実装設計の指針とすることを試みた。また、実装指針に即した雛形部品を提供を行った。

(2) 部品間の自動連携
分散に対応したデータバッファ部品を用意し、GUI部品やユーザコードとの連携を実現した。

(3) データ構造変更の吸収
ビューに対する可塑性をもつデータバッファ部品を用意し、これを用いてエンティティオブジェクトのデータ部分を実装することで解決を図った。

4.本実装アーキテクチャの構成

4.1.実装環境

言語：Microsoft Visual Basic 4.0 Enterprise Edition
OS:　WindowsNT 3.51 (server) / Windows95 (client)
DBMS:　Oracle 7.1, Oracle Objects for OLE
RPC:　Remote OLE

4.2.実装構造

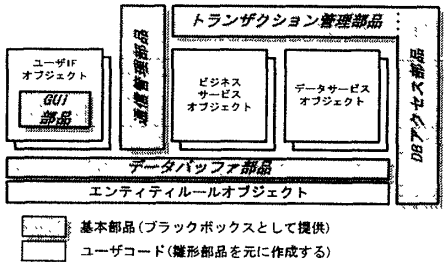

図 1:実装構造の概念図

図1に示すように、論理設計により作成されたユーザインタフェース・ビジネスサービス・データサービスの各オブジェクトはそのまま実装設計に引き継ぐ。一方、エンティティオブジェクトは、データを保持するデータバッファ部品(後述)と挙動を実装するエンティティルールオブジェクトとの組合せにより実装する。

4.3.実装規約

ユーザ定義のオブジェクトの構成と責務を規定する実

Object Oriented Software Development Method for Business Systems(3): - An Implementation Technique to Improve Reusability -, HARADA Michiaki, YURUGI Yuko, KITABATAKE Shigenobu and UEHARA Kenji, MITSUBISHI Electric Corporation

装規約を、設計パターン[1]とクラス／メンバ命名規則によって構成した。現在、以下についての実装規約を規定している。

- C/S間の接続の管理
- 検索データの管理／DB資源の確保・開放
- トランザクション文脈の管理
- 画面遷移の管理
- DB処理パターン[2]ごとのオブジェクト構成

図2に本実装規約で規定されるアプリケーションの全体構造を示した。

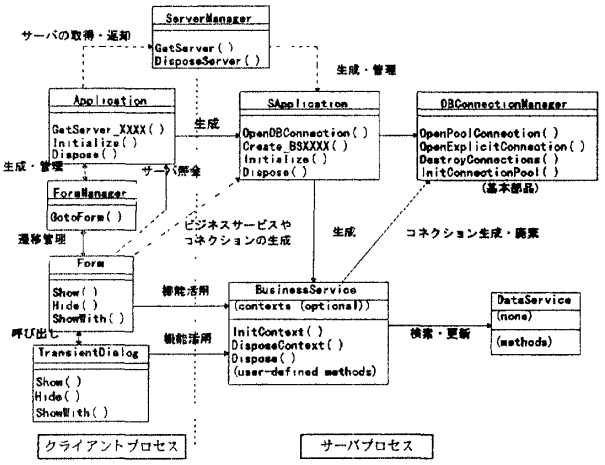

図2:アプリケーション全体の構造を示す設計パターン

4.4.基本部品

3階層システムで汎用的に使用される実装部品をブラックボックス化し、実装の複雑性の解消を図った。

(1) DBアクセス部品

トランザクション文脈の管理、検索・更新等のDBアクセス機能を提供する。

(2) データバッファ部品

検索データ等のワークデータを保持する部品であり、以下の特徴を持つ。

- ビューの柔軟な利用

  データバッファ部品は動的SQLにより生成された任意のビューを保持し、カラム名をキーとして属性へのアクセスを提供する。適切にビュー設計により同一のビジネスサービス・エンティティルール等を用いて異なるビューを加工するように構成すれば、再利用性の向上が図られる(図3)。
- データ構造の変更に強い柔軟な構造

  同様の理由により、データ構造の変更の際、変更された属性を用いているオブジェクトのみに修正範囲を極小化できる。
- リモートノード間でのデータの一括転送

  動的SQLで生成されたビューをノード間で転送可能な形式に自動的に符号化／復号する機能をもつ。これによってデータ構造の変更がノード間の通信インタフェース設計に波及するのを防いだ。
- GUI部品・エンティティルールとの連携

  データの複写や入力のアンドゥ、GUIイベントに応じたルールの呼出しなどの定型的な連携を自動化する。これによってコード量の削減が図られる。

(3) GUI部品

前記のデータバッファ部品との連動機能を持つ。

(4) 通信管理部品

クライアントに対するサーバ側オブジェクトの割付・解放を管理し、負荷分散を実現する。

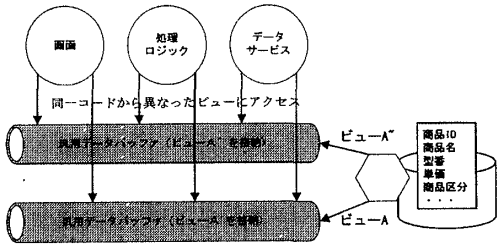

図3:ビューの柔軟な利用

4.5.雛形部品

Visual Basicでは継承機能がなく、抽象クラスによるフレームワーク提供は不可能である。ユーザによる修正を前提としたコードの雛形を部品として提供することにした。以下の2つの観点から部品を切り出した。

(1) 実装規約に即したオブジェクトの雛形

4.3.で述べた実装規約に含まれる設計パターンに即したオブジェクトの雛形を提供した。

(2) DB処理パターン[2]に応じた雛形部品

定型的業務アプリケーションのデータ構造と処理パターンを類型化して設計パターンを抽出し、パターンに即したオブジェクトの雛形を提供した。

5.おわりに

オブジェクト指向設計によって得られた論理設計とのシームレス性とデータ構造の変動に対する柔軟性を目的とした3階層システム向けの実装アーキテクチャを示した。今後はこれらの利用を自動化することによる開発効率向上を検討する。

参考文献

[1] E.Gamma他, "Design Patterns", Addison-Wesley, 1995.

[2] 鈴木由美子, 萬木優子, 原田道明, 徳本修一 他, "ビジネス系システム開発におけるオブジェクト指向開発技法" (1)〜(4), 本大会予稿集, 1997.