ビジネスオブジェクトを用いたアダプティブシステムの開発: 変更要因分析の導入とその影響*

5R－8

北山文彦　久世和資　今野和浩†
日本アイ・ビー・エム（株）東京基礎研究所‡

1　はじめに

本稿では、ビジネスオブジェクトを用いて柔軟に変更可能なアプリケーションを構築する際に、将来の変更要因を分析・設計に反映させる方法について述べる。変更要因分析の目的は、変更する部分の特定を容易にし、変更作業を最小にし、変更に伴う影響を最小限にするためである。我々の研究目標は、ビジネスオブジェクトレベルでの変更要因分析の手法とツールを開発することにより、エンドユーザの言葉で変更に適応できるようにし、実装レベルの変更を自動化することである。

まず、変更要因の分類を行ない、初期開発や変更時の保守においてどのようにビジネスオブジェクトの分析や設計をすべきかを提案する。次に、実際に金融分野のアプリケーションプロトタイプを作成し、いくつかの実験的な変更を行ない、変更要因分析の効果の検証を行なった。

2　変更要因の分類とその分析・設計法

従来からの開発方法論では、将来の機能的、論理的な変更は、分析や設計の上流ではあまり考慮されていないが、設計から実装レベルでは経験的な手法が数多くあり実践されてきている (例えばデザインパタン [1])。実際に変更が起きた場合も、実装レベルで対応することが多い。ビジネスオブジェクトを用いた開発でも、そのモデル化や設計ではこのような分析はされておらず、より抽象度の高い機能や論理的な面での再利用性や柔軟性を阻害していると思われる。

初期開発時にすべての変更を予測することはできない。そこで、このような様々な変更を抽象化・分類し、分析や設計オブジェクトに反映させることを考える。このような抽象化された変更を変更要因と呼び、開発対象にどのような変更要因があるかを特定する作業が変更要因分析である。ここでは、情報システムの機能的変更要因として、(a) ロジックの変更、(b) 概念の詳細化、(c) 様々な要因の増大、(d) ワークフローの変更、に分類して考え、それぞれモデルにどう反映させるべきかを議論する。

ロジックの変更は、情報処理における様々な計算の仕様やアルゴリズムが変更されることであり、数値計算のみならず、判断や決定なども含む。情報システムを開発する際に、あるロジックの変更可能性が高いことが予想される場合は、その部分を特定し独立したオブジェクトとして表現する。変更が生じた場合、変更部分の特定 (=ロジックのオブジェクト) が簡単であり、詳細化のインヘリタンスにより変更作業も軽減される。図 1(a) に変更要因のオブジェクト図による記述と変更例を示す。破線内の部分が変更要因の記述であり、点線が変更例である。

ビジネスオブジェクトは情報処理に必要な概念を表わしたものであるが、単一の単純な概念だったものが、詳細化され分類されてくることがある。このようなオブジェクトの詳細化に対しては、あらかじめ考慮する点はあまりないが、変更する時点では、詳細化のインヘリタンスを用いる (図 1(b))。

情報処理を行なう際には、様々な要因を参照して行なう。これをビジネスオブジェクトでモデル化した場合、要因をオブジェクトとして表わし処理を行なうオブジェクトへ関連をもたせて表現する。これらの要因が、環境の変化に合わせて増大することは充分に考えられる。ここで問題となるのは、新しい要因 (オブジェクト) を増やしたときに、既存のオブジェクトやその処理コードに対する影響が少なくなるようにすることである。これには、あらかじめ抽象化のインヘリタンスを用いて要因の抽象クラスを導入し、処理コードではポリモルフィズムを用いる記述をする。変更時には、サブクラスを追加するだけでよい (図 1(c))。

最後に、ワークフローの変更要因はオブジェクトモデルのみならず、ワークフローやユーザインタフェースのモデルを独立して考えることによって行なう。従って、これらの 3 つのモデルからシステムを構築する方法論が必要となる。さらに、実装上の影響を考慮するとワークフローの実現モデルの導入も必要となる [2]。

---

*Adaptive Systems Development with Business Objects: Change Analysis

†Fumihiko Kitayama, Kazushi Kuse, Kazuhiro Konno
‡IBM Research, Tokyo Research Laboratory

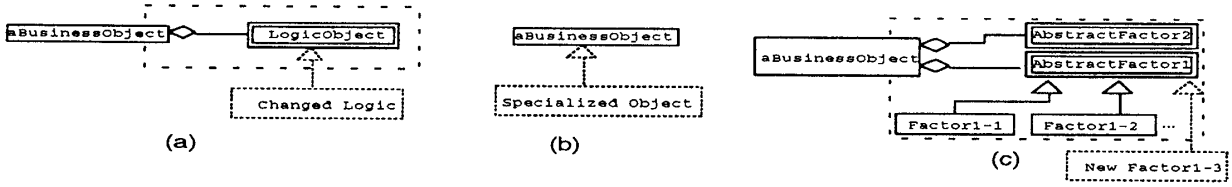

図 1: Changing Analysis and Description

3 実験的開発による変更要因分析の検証

3.1 実験プロトタイプシステム概要

金融分野の営業システムプロトタイプ (GUI付分散システム) を用いた。このシステムはオブジェクト指向分析・設計で開発され、実装はSmalltalk(GUI)、C++(分散オブジェクト)、ワークフローマネージャーを用いている。規模は、分析時のオブジェクト数が17, 実装段階では分散オブジェクト (C++)29、GUI オブジェクト (Smalltalk) 約 20 個からなり、C++部分で約 10000 行のプログラムである。

変更項目	事前分析	追加クラス数 (行数)	変更クラス数 (行数)
(1) 新顧客の導入と与信計算の変更 (詳細化, ロジック変更)	あり	2 (285 行)	0 (0)
(2) 新取引追加 (要因の増大)	あり	1 (117 行)	1 (211 行)
(3) 新チャネル要因導入 (要因の増大)	なし	0 (0)	3 (1928 行)

表 1: Influence of Experimental Changes

3.2 変更要因分析

プロトタイプ作成段階で、いくつかの変更要因分析を行なった。具体的には、「顧客」対する与信を計算する部分は変更が多いため、この計算を主に行なう「与信」オブジェクトをモデルに導入している。また、顧客との種々の取引に関してそのバリエーションの増大に対処するため、取引に関して抽象オブジェクトを導入し、ポリモルフィズムを用いた設計を行なっている。

3.3 変更実験

いくつかの変更を実際に行ない、それぞれコードの変更量を測定した。具体的には、顧客の種類を増やしそれに合わせて与信計算のロジックを変更した (変更項目 (1))。これは、変更要因として想定されたものである。さらに、銀行振込取引を新設した。これは、顧客との取引のチャネルを店頭だけでなく、銀行振込の形態に拡張したものである。この変更は、「取引」のバリエーション増加という点 (変更項目 (2)) では、あらかじめ想定したものであったが、取引チャネルという要因増加 (変更項目 (3)) に関しては想定外である。

3.4 評価

表1に各変更に対する追加および変更クラス数とその行数 (ヘッダファイルも含む) を示す。変更項目 (1) と (2) では、変更要因分析が行なわれその反映がされているので、サブクラス化によるクラスの追加だけでほぼすむ。

一方、対応する変更要因分析が行なわれなかった変更項目 (3) の場合、もともとのモデルにこの要因に相当する (抽象) オブジェクトがないため、サブクラス化等によるオブジェクトによる設計が困難であり、既存のコードをかなり変更することになる。しかも、この変更による設計・実装では、オブジェクト指向的なものでなくなる。

4 おわりに

本稿では、柔軟に変更可能なシステム構築において、変更要因を分類し変更要因分析の方法について述べた。また、簡単な変更実験を行なうことにより、その効果について検証した。

このような変更要因を分析・設計方法に取り入れ、分析から設計・実装まで一貫して支援するツールを用意することにより、柔軟に変更可能なシステム構築に役立つと予想される。このようなツールのプロトタイプを作成し検証していく予定である。

参考文献

[1] E. Gamma, R. Helm, R. Johnson, and J. Vlisiddes. *Design Patterns.* Addison-Wesley, 1995.

[2] 今野他. 分散オブジェクトを用いてワークフローを実現するシステムモデル. 第52回情報処理全国大会, 1996