2005－IS－93（1）
2005／8／25

MDA によるコンポーネントベースモデリングの実例

浜口弘志　原口拓也　桐越信一　大場みち子
株式会社 日立製作所 ソフトウェア事業部

情報システムを効率良く開発するためには、モデリングを利用して組織全体のビジネスプロセスやデータ、情報システムの関係構造をEA(Enterprise Architecture)の体系に合わせて整理し、業務モデルからアプリケーションモデルへの連携を実現する開発方法論の策定が必要である。この連携を実現させるためには、モデル変換の定義であるマッピングルールを確立しなければならない。我々のプロジェクトでは、Java™アプリケーションの開発において、国際標準化団体OMG(Object Management Group)の推し進める参照アーキテクチャMDA(Model Driven Architecture)とUML(Unified Modeling Language)をベースとし、業務分析の段階からコンポーネントを意識した分析モデルを作成した。さらに、その結果を設計・実装モデルにマッピングしてJavaソースコードを生成する開発スタイルを採用した。本論文では分析モデルの作成から設計・実装モデルに変換するまでの実例について報告する。

Report of component base modeling by MDA

Hiroshi Hamaguchi　Takuya Haraguchi　Shinichi Kirikoshi　Michiko Oba
Software Division ,Hitachi, Ltd.

In order to develop an information system efficiently, it is necessary to arrange a structure of the business process, data, and the information system of the entire organization to the system of EA(Enterprise Architecture) by using modeling to develop the information system efficiently, and to settle on the development methodology that achieves cooperation from the business model to the application model. It is necessary to establish the mapping rule that is the definition of conversion. In our project of Java application, the analysis model that is considered the components from the stage of the business analysis was made, based on reference architecture MDA(Model Driven Architecture) that international standardization group OMG(Object Management Group) promoted and UML(Unified Modeling Language). In addition, the development style that the result is converted to the design and the implementation model and generated the Java source code was adopted. It reports an example of development from creating analysis model to converting it to the design and the implementation model.

１．MDA 概略

従来の情報システム開発では、EA のように組織構造の共通化はされてなかったものの、各企業はそれぞれの方針に従って業務を分析し、情報システムの構築を行ってきた。ただ、従来の開発手法では、業務の分析結果と OS や言語などのプラットフォームやアーキテクチャとが密接に結びついているため、プラットフォームやアーキテクチャが変化した場合、それに応じて業務の分析から情報システム化というサイクルを繰り返す必要があった。

MDA では、業務の変化とプラットフォームの変化のスピードが異なる点に注目したことで、業務分析の視点と設計・実装の視点を分離してそれぞれ独立したモデルを作成することができ、プラットフォームやアーキテクチャが変わっても業務分析モデルを流用することを目的とし

ている。また、分析工程と設計・開発工程を分け、それぞれのプロセスで OMG が制定した UML を共通言語として使用する。MDA の分析・設計工程は次に示す３つのフェーズに分けられる。

(1) CIM（Computation-Independent Model、業務分析）

システムの構造は提示せず、業務の構造やフローを明確にするフェーズである。

(2) PIM（Platform-Independent Model、要求分析・システム分析）

プラットフォームに依存しないモデルを作成するフェーズである。要求分析ではシステムの外部仕様を、システム分析では内部仕様を定義する。

(3) PSM（Platform-Specific Model、システム設計）

PIM を特定の技術にマッピングし、プラットフォームやアーキテクチャに依存したモデルを導出するフェーズである。

MDA の CIM・PIM・PSM 工程と EA との関連、さらに分析フェーズごとの外部インターフェースおよび必要となる内部の振舞いを図　1 のMDA 開発プロセス概略に示す。

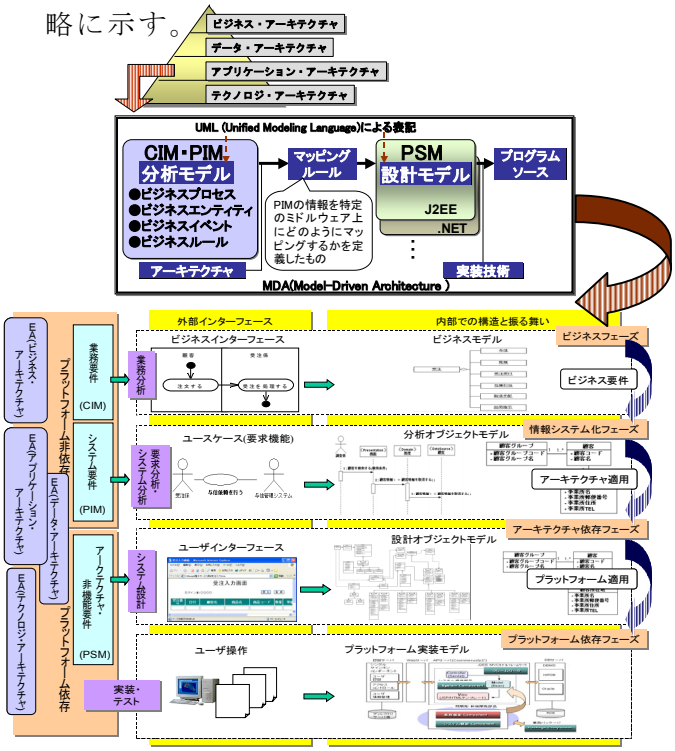

図　1　MDA 開発プロセス概略

２．コンポーネントベースモデリング

２．１　目的

以下に示す事項を目的として開発プロセスと方法論の策定を行い、コンポーネントベースモデリングを実現した。

(1) 分析工程でのコンポーネント導出

分析工程でコンポーネントを見出し共有することで、業務をコンポーネントの階層構造で実装することを目指し、実装時の工数短縮を実現する。

(2) 各開発プロセス関連の明確化

上位プロセスの成果物が下位プロセスの入力になることを明確にすることで、ある程度機械的な判断による次プロセスのドキュメント作成を可能とする。

(3) マッピングルールの確立

本論文の開発方法論では、マッピングルールをパターンで定義し、分析モデルに適用することで、PSM 以降の成果物を作成してゆく。この際、ツールを使用することで、機械的な作業でのマッピングが可能になる。また、プロジェクトで統一されたマッピングルールをモデルに適用することで、分析工程で見出したコンポーネント構造を設計・開発工程まで引き継ぐことを目指す。

２．２　コンポーネントの定義

本論文の対象システムの全体像を図　2 技術資料配布管理システム概要に示す。以降は図　2 破線部分の実例をもとにコンポーネント構造を示す。

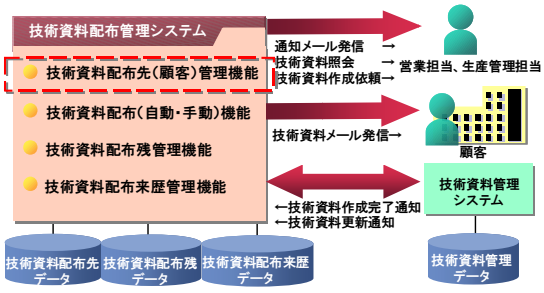

図　2　技術資料配布管理システム概要

このシステムは、製品に関する技術資料の配布を管理するシステムであり、営業支援システムの一部である。このシステムは、配布先や配布来歴の管理など図 2に示す4つの管理機能を持つ。図 2の業務に限らず、業務は小さな単位の業務機能の組み合わせで表現でき、さらにその業務機能はより小さなシステム機能の組み合わせで表現できる。ゆえに業務は機能階層表現が可能であり、実装の際にコンポーネントにマッピングできる。図 3にコンポーネントの定義と、これを導出するために分析工程で使用するダイヤグラムを示す。

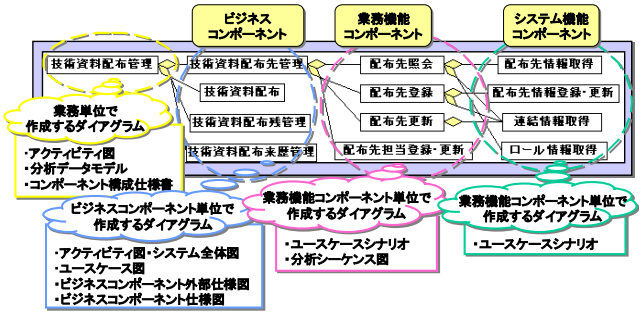

図 3 コンポーネントの定義

図 3より、技術資料配布管理は技術資料配布先管理や技術資料配布などの業務で構成される。さらに技術資料配布先管理は、配布先照会、配布先登録、配布先更新などの業務機能で構成される。図 3に示すように、それぞれのレイヤをビジネスコンポーネント、業務機能コンポーネント、システム機能コンポーネントと定義した。このようなの階層構造から、上位コンポーネントは下位コンポーネントの組み合わせで表現できることがわかる。

このように業務レベルのコンポーネントを導出・共有することで、実装時に共有コンポーネントを見出す場合に比べ、より大きな単位の共有コンポーネントを見出すことができるため、コンポーネント開発工数が短縮できる。また、システム全体を業務や機能の単位に分割してモデリングすることで、モデルの複雑化を防ぐことができる。

3．開発プロセスの策定

3．1 各工程の関与者

CIM、PIM、PSM、設計・開発各工程での関与者を図 4の各工程と関与者のように定義した。

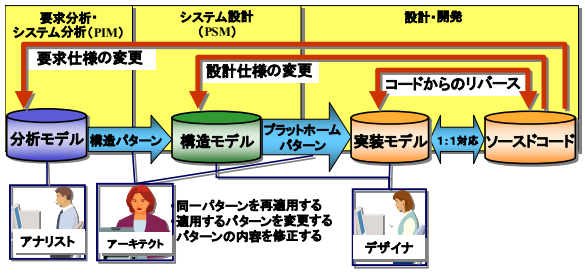

図 4 各工程と関与者

(1)アナリスト

CIM、PIM 工程を担当し、ユースケースや分析クラス図などの分析モデルを作成する。また、分析モデルに適用するパターンを決定する。

(2)デザイナ

PSM、開発・設計工程を担当し、構造モデルから実装モデルへ変換するパターンを決定する。また、クラスの性質に応じたメソッド定義やデータ定義など、実装モデルからソースコードの作成を行う。ただし、PSM 以降の工程では DB 周りの設計なども含まれるが、本論文では主に業務機能の設計を対象とする。

(3)アーキテクト

実装環境の規定やフレームワークの適用など実装を支えるアーキテクチャを定義し、分析モデルから構造モデル、構造モデルから実装モデルへのマッピングルール作成を担当する。マッピングの際にはモデル変換ツールを使用する。

３．２　開発プロセスの関連と成果物

開発プロセスの策定においては、各プロセスにそれぞれ UML での成果物を定義し、コンポーネント見出しが可能となるようなプロセスを検討した。UML では数種類の図が規定されているが、全てを利用する必要はないため、何を見出すのかという目的を明確にした上で必要なものだけを採用した。

モデリング方針によっては、UML で規定された図だけでは仕様を記述するには足りない場合がある。本論文においても、UML だけでは不足する図や資料は別途定義した。なお、PSM 以降の工程は CIM・PIM の成果物にマッピングルールを適用することで実現する。CIM・PIM・PSM 作成のためのプロセスと作成モデルを図 5 のように開発プロセスと成果物として定義した。

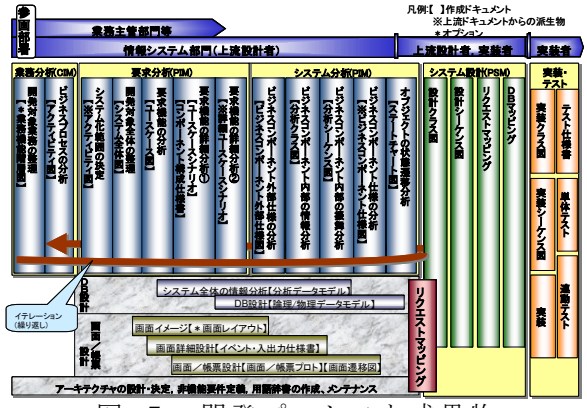

図　5　開発プロセスと成果物

これらのプロセスはお互いに関連を持っており、この関係も明確にした。図 6 に開発プロセス関連図を示す。

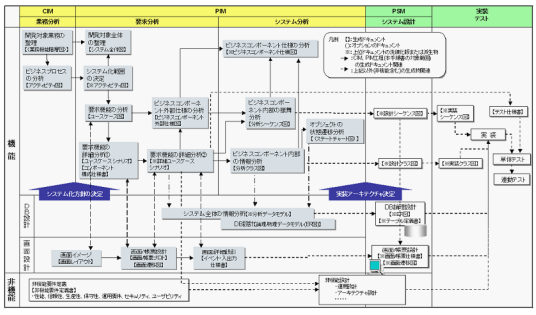

図　6　開発プロセス関連図

４．開発方法論の策定

次に、策定した開発プロセスの個々のプロセスについて、実例をもとに述べる。

４．１　業務分析工程(CIM)

この工程はアナリストが担当し、対象業務にどのようなビジネスプロセスが存在するかを明確にする。さらに、業務に関連する部署や人などを洗い出し、業務フローを明確にすることで業務を細分化し、業務の機能構造を明確にする。

(1)開発対象業務の整理【※業務機能階層図】

準備されている資料の利用や、業務主管部門への質問により業務を細分化し、業務機能階層図として整理する。

(2)ビジネスプロセスの分析【アクティビティ図】

対象業務に関連する部署や人などを洗い出し、業務の流れをアクティビティ図として表す。また、業務を分割して業務機能階層図へ反映し、業務機能コンポーネント、システム機能コンポーネントを導出する。

図　7 に技術資料配布先管理のアクティビティ図を示す。

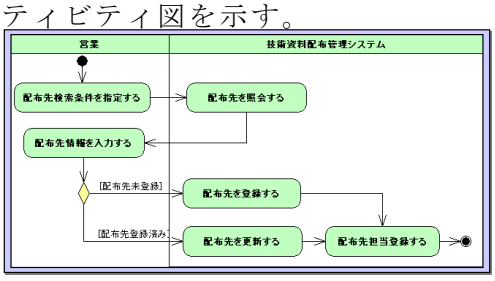

図　7　技術資料配布先管理

４．２　要求分析工程(PIM)

この工程はアナリストが担当し、業務分析の結果から対象業務のシステム化範囲を決定し、同時におおまかな入出力情報を洗い出す。その結果からシステム全体を俯瞰できる全体図を作成する。また、ユーザーや外部システムとの関係を定義し、システム化範囲の詳細な処理を記述する。これらの情報をもとに、ビジネスコンポーネン

トの仕様決定に先立ち、そのビジネスコンポーネントが外部に公開するインターフェースを明確にする。

(1)システム化範囲の決定【アクティビティ図】

業務分析の結果であるアクティビティ図を用いて、システム化する範囲を決定する。アクティビティ図は業務フロー図であるため、システム化対象外のアクティビティも導出されている。このため、それらの中から今回のシステム化の対象を決定する。図 ８にシステム化範囲の決定を示す。

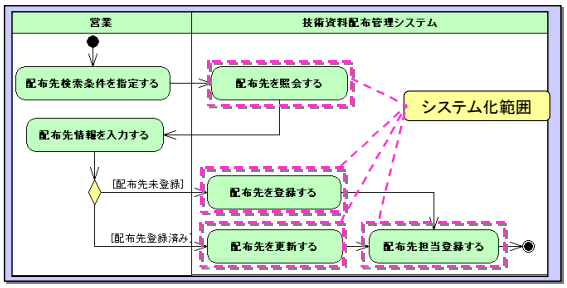

図 ８　システム化範囲の決定

(2)開発対象全体の整理【※システム全体図】

開発対象に存在するビジネスコンポーネントと、それがユーザーに提供する機能、入出力される情報の相関関係をシステム全体図として表す。

(3)要求機能の分析【ユースケース図】

アクティビティ図をもとに、業務機能コンポーネントの対象となるものをユースケース、外部要因（ユーザーや外部システム）をアクターとし、それらの関係をユースケース図として表す。

このプロセスは次のプロセスである「要求機能の詳細分析」結果を反映することで、要求機能のさらなる細分化を行う必要がある。

(4)要求機能の詳細分析【ユースケースシナリオ】

画面レイアウト、帳票などから入出力情報を明確にし、ユースケースの詳細な処理をユースケースシナリオとして記述する。ユースケースシナリオは、システム化範囲のアクティビティと入出力情報を参考にしてさらなる分析を行い、処理の内容をステップとして文章で記述する。ユースケースシナリオを(3)「要求機能の分析」の結果に反映することで、システム機能コンポーネントの細分化と洗練化を行う。図 ９にユースケース図への反映例を示す。

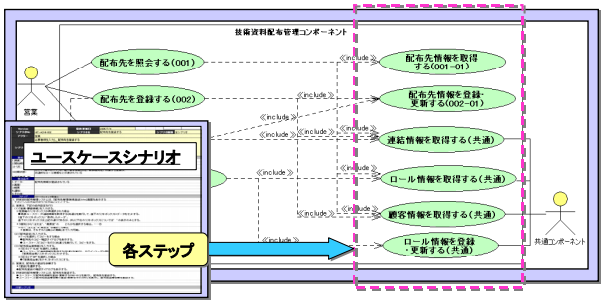

図 ９　ユースケース図への反映例

なお、このプロセスではパッケージの分割も考慮する。

(5) ビジネスコンポーネント外部仕様の分析【ビジネスコンポーネント外部仕様図】

ユースケース図とユースケースシナリオを基に、ビジネスコンポーネントが外部に公開する機能と、そこで入出力される情報をビジネスコンポーネント外部仕様図として表す。図 １０にビジネスコンポーネント外部仕様図を示す。

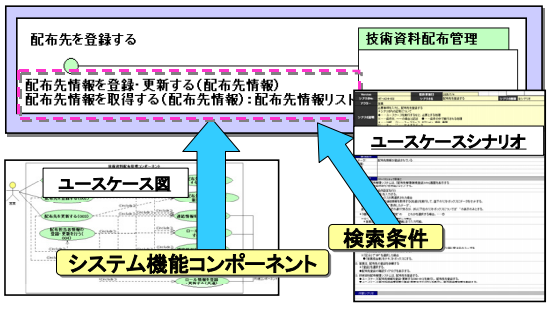

図 １０　ビジネスコンポーネント外部仕様図

４．３　システム分析工程(PIM)

この工程はアナリストが担当し、ビジネスコンポーネントで使用する情報とその関連を明確にする。これらの結果より、ビジネスコンポーネントの外部仕様と内部仕様を明確にする。さらにＤＢ設計の状態フラグ定義などに利用するために、オブジェクトの状態遷移を明確にし、システム全体のデータ構造を明確にする。

(1) ビジネスコンポーネント内部の情報分析【分析クラス図】

シナリオや帳票､コンポーネント外部仕様（インターフェース）を基に、その業務で使用する情報の静的な構造を分析クラス図として表す。分析クラス図の作成が困難な場合はオブジェクト図を作成してから分析クラス図を導き出す。図　１１に分析クラス図を示す。

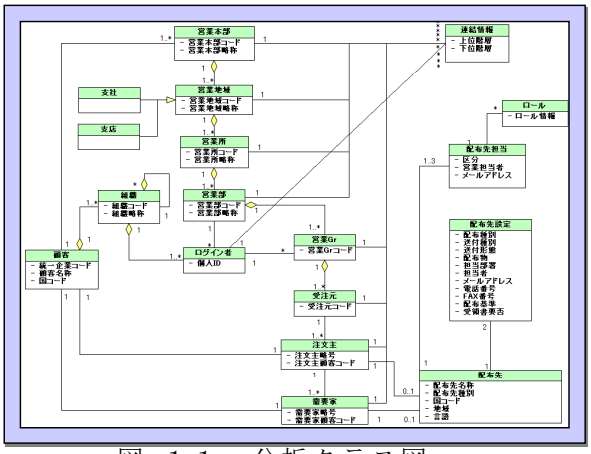

図　１１　分析クラス図

なお､このプロセスではパッケージの分割も考慮する。

(2) ビジネスコンポーネント内部の振舞い分析【分析シーケンス図】

シーケンス図とは､オブジェクト間の相互作用（メッセージのやりとり）を時系列にそって表現したものである。ここでは、シナリオのステップを参考にし、ビジネスコンポーネントが起動された際の内部処理手順をシーケンス図で表現する。シーケンス図はビジネスコンポーネントのインターフェース単位に作成する。

(3) ビジネスコンポーネント仕様の分析【※ビジネスコンポーネント仕様図】

ビジネスコンポーネントの仕様を明確にするために、要求分析の成果（外部仕様）とシステム分析の成果（内部仕様）をまとめたものである。ビジネスコンポーネント外部仕様図と分析シーケンス図と分析クラス図からビジネスコンポーネント仕様図を作成する。図１２にビジネスコンポーネント仕様図を示す。

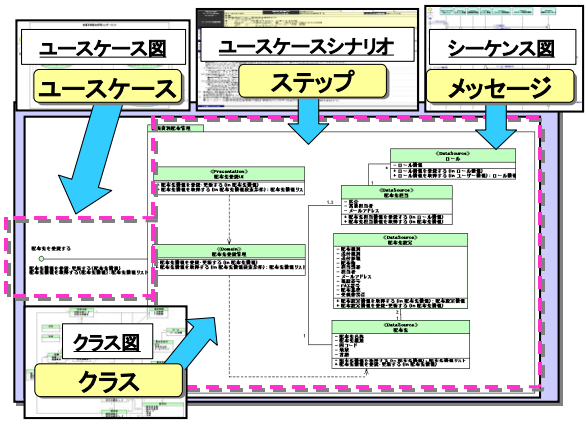

図　１２　ビジネスコンポーネント仕様図

(4) オブジェクトの状態遷移分析【ステートチャート図】

特に状態遷移がビジネスプロセスに影響を与えるクラスについては、分析クラス図と分析シーケンス図をもとに、ステートチャート図を作成する。ユースケースシナリオ中にステート（状態）が切り替わるイベント（タイミング）を明記している場合、それを参考にして作成する。

(5)システム全体の情報分析【分析データモデル】

ビジネスコンポーネント単位で分析した情報をシステム単位にまとめる。この内容は以降の ER 図作成など、DB 設計へと繋げていく工程である。分析データモデルは分析

クラス図の集合体であり、クラスの状態を表す属性（ステートチャート図から導く）なども含めて作成する。

４．４　PSM 工程、設計・開発工程

次に、分析モデルにマッピングルールとしてパターンを適用し、構造モデルを作成する。アナリストは分析モデルに適用するパターンを決定し、アーキテクトはそれを元にマッピングルールの作成を担当する。

本論文では分析クラス図から構造クラス図へのマッピングについて述べる。表 ４．１　パターンの種類にクラスに適用するパターンの一部を示す。

表 ４．１　パターンの種類

大分類	性格名	PIMでの解釈	PSMでの解釈
マスタ系エンティティ	コード型	コードをキーとした属性を管理するエンティティクラス	マスタ情報の入出力制御を行う
	区分型	区分値を管理するエンティティ(共通コードテーブル)クラス	コード情報の入力管理を行う
	再帰型	木構造を持つリソースを管理するエンティティクラス	関連する個別情報の入出力制御を行う

図 １１より、コードをキーとして持つ営業本部クラスに、表 ４．１のパターンの一つであるコードパターンを適用した例を図 １３のクラスのマッピングに示す。

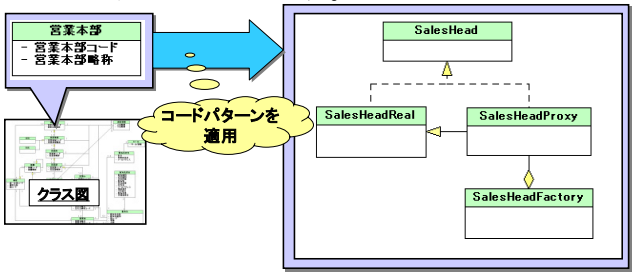

図 １３　クラスのマッピング

営業本部クラスはコードパターンを適用することで、共通インターフェースとしてのベースクラス、ベースクラスのインスタンス化を行うファクトリクラス、コードと名称だけを持つプロキシクラス、マスタ系情報を保持するターゲットクラスの４つのクラスとして表現される。

上記のように、CIM・PIM の成果物にパターンなどのマッピングルールを適用することで PSM 工程以降の成果物を作成してゆく。なお、PSM から設計・開発へのマッピングは、プラットホームパターンを適用して実装モデルを作成する。以降の実装・デバッグ以降の開発はデザイナが担当する。

５．まとめ

今回報告したコンポーネントベースの開発プロセスと開発方法論については、すでに実際のプロジェクトでこのモデリング手法を適用している。技術資料配布管理システムで見出した分析工程におけるコンポーネント総数と流用率を表５．１に示す。

表５．１コンポーネント総数と流用率

コンポーネントの種類	新規開発数	今回開発流用数	既存流用数	総数	流用率(%)
ビジネスコンポーネント	4	0	0	4	0
業務機能コンポーネント	12	8	0	20	40
システム機能コンポーネント	9	14	30	53	83

表５．１に示したように、実際の分析工程では、業務機能コンポーネントの流用率は 40%、システム機能コンポーネントの流用率は 80%を超える結果を得ることができた。また、本論文のマッピング方法を適用することで、この分析結果を設計・開発まで連携させることができ、コンポーネントベース開発の実現からシステム開発の効率化を計ることができる。

マッピングの際には、アーキテクトがアナリストの意向を理解した上でパターンを作成・適用し、開発・デバッグ以降へとデザイナへ引き継ぐ必要がある。そのプロセスを確立するためには、マッピングルールの適用規則を開発方法論として明確にする必要がある。今後も、設計部署の協力やプロジェクトで得た成果をもとに、開発方法論のブラッシュアップのために研究を進めていく所存である。

参考文献

(1) David S.Frankel、”MDA モデル駆動型アーキテクチャ”、日本アイ・ビー・エム株式会社 TEC-JMDA 分科会、(株)エスアイビー・アクセス、2003.
(2) (株)テクノロジックアート、”独習 UML”、 (株)翔永社、2005.
(3)長瀬嘉秀、橋本大輔、”UML システム設計実践技大全”、(株)ナツメ社、2004.
(4)湯浦克彦、大坪稔房、団野博文、石井義明、古澤憲一、桐越信一、鈴木文音、”EJB コンポーネントによる Web システム構築技法”、(株)ソフト・リサーチ・センター、2002.
(5)Martin Fowler、”UML モデリングのエッセンス第 2 版”、 (株)翔永社、2000.
(6)渡辺政彦、飯田周作、石田哲史、山本修二、浅利康二、”UML 動的モデルによる組み込み開発”、(株)オーム社、2003.
(7)ObjectManagementGroup、“UML 仕様書”、(株)アスキー、2001.
(8)Anneke Kleppe、Jos Warmer、Wim Bast、”MDA モデル駆動型アーキテクチャ導入ガイド”、(株)インプレス、2003.
(9) Craig Larman、”実践 UML パターンによるオブジェクト指向開発ガイド、(株)プレンティスホール、1998.
(10)(株)テクノロジックアート、”パターンモデリングガイド”、(株)ピアソン・エデュケーション、2002.
(11)Ivar Jacobson、Grady Booch、James Rumbaugh、”UML による統一ソフトウェア開発プロセス”、 (株)翔永社、2000.

商標