2004－IS－89　(4)
2004／8／26

アプリケーション開発におけるコンポーネントベースモデリングの適用

安部麻衣　桐越信一　浜口弘志　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よく開発するために、業務モデリングの分野に注目が集まっている。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と繋げていくことで、経営と情報システムを結びつけた管理を行っていくことができるためである。このためには、業務モデリング部分における開発方法論の策定とプロセスおよびそのプロセスの関連を確立することが重要となる。開発方法論の策定においては、汎用的なモデルを導出するため、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)の考え方を採用し、異なるアーキテクチャやプラットフォーム間でのモデルの再利用を可能とした。さらに、業務レベルのコンポーネントを導出できるよう、UML（Unified Modeling Language)を使用したモデリングプロセスを定義し、プロセス間の関連と導出方法を明確にした。これらの結果について報告する。

Component base modeling in application development

Mai Abe　Shinichi Kirikoshi　Hiroshi Hamaguchi　Michiko Oba
Software Division, Hitachi, Ltd.

In order to develop an information system efficiently, attention has gathered for the field of business modeling. It is because management connected with the information system can be performed by arranging the business process of the whole organization, data and the structure of an information system according to the structure of EA (Enterprise Architecture) using modeling. It is important to establish the development methodology itself in a business modeling, and the relation between the processes of the methodology. When we developed the methodology, we adopted the view of MDA (Model Driven Architecture), which is the reference architecture the international standardization organization OMG (Object Management Group) promotes, since a general-purpose model was derived and reuse of a model was enabled between the different architecture and the different platform. Furthermore, we defined the modeling process using UML (Unified Modeling Language), to clarify the relation and the derivation method between processes so that the component of a business level could be derived. These results are reported.

1.ＭＤＡ概略

従来からの情報システム化を振り返ると、EA のような情報システム方針に対する体系の共通化はなかったものの、各企業ではそれぞれ方針を立てて業務を分析し、情報システムの構築が行われてきた。ただ、従来の開発手法では、業務の分析結果とプラットフォームやアーキテクチャが密接に結びついているため、OS や言語などの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る。

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、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で、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できることを目指している。業務分析モデルを見出すためのモデリング工程、あるいはそれらを情報システム化するための設計工程を分け、それぞれの工程で OMG が制定した UML を共通言語として使用する。分析・設計工程は次の３つのフェーズに分けられる。

(1) CIM（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（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

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。

図 1 に EA と MDA の概要を示す。

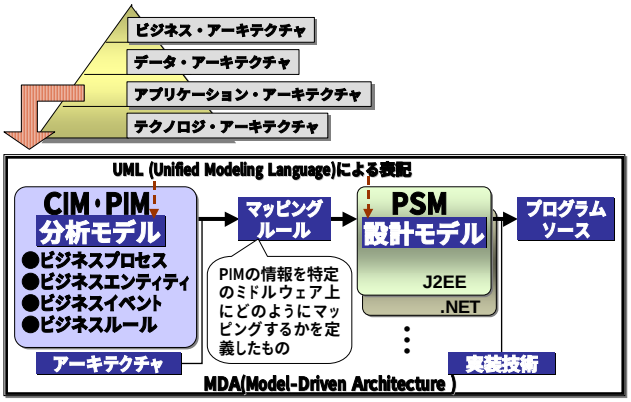

図 1　EA と MDA の概要

本報告では CIM、PIM におけるモデリングの開発プロセスと方法論策定について述べる。

２．コンポーネントベースモデリングの目的と定義

2.1　目的

開発プロセスと方法論の策定にあたり、以下に示す事項を目的とした。

(1) 分析工程でのコンポーネントの導出が可能なこと

業務をモデリングし、そのモデル自体を流用できることはもちろんのこと、実装時の工数短縮のため、業務分析工程でのコンポーネントの見出しを可能とする。モデルはコンポーネントの組立型で実装できることを目指す。

(2) 各開発プロセスの関連が明確なこと

上位プロセスの成果物が次のプロセスの入力となる部分を明確にし、プロセス間の関係を導き出すことで、ある程度機械的な判断による次工程のドキュメント作成を可能とする。

2.2　コンポーネントの定義

ある業務を考えた場合、それは小さな単位の業務の組立で表現でき、さらにその業務はより小さな業務機能の組み合わせで表現できる。その階層は業務機能階層となり、実装の際にコンポーネントにマッピングできると考えた。図２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。

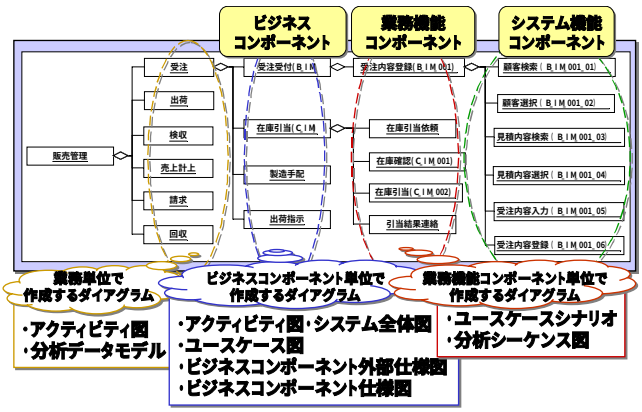

図２　コンポーネントの定義

図２は販売管理の例であるが、販売管理は受注や出荷などの業務で構成される。さらに受注は、受注受付、在庫引当、出

荷指示などの業務で構成される。かつ、受注受付は顧客検索、見積検索などの機能で組み立てられている。ここで図２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。

このような構成を見出してモデリングすれば、ビジネスコンポーネントは業務機能コンポーネントの組立であり、その業務機能コンポーネントはシステム機能コンポーネントの組立となる。また、下位のコンポーネントは上位コンポーネントでの共有が可能となり、実装時にのみ共有コンポーネントを見出す場合に比べ、開発工数が短縮できる。

３．開発プロセスの策定

3.1 開発プロセスの概略

開発プロセスを検討するに当たり、MDA の CIM、PIM 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フェースおよび必要となる内部の振舞いを開発プロセス概略として図３に示す。

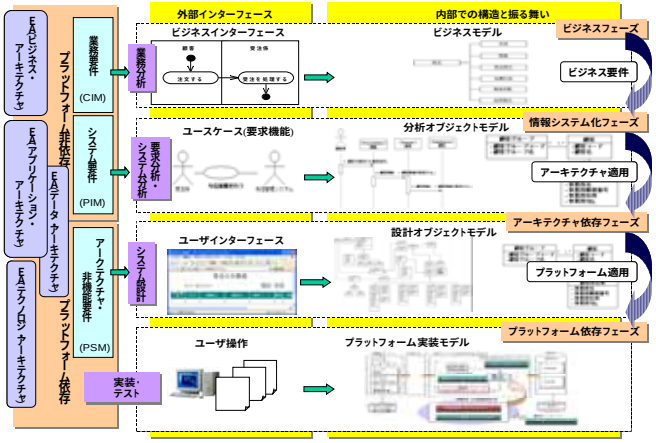

図３　開発プロセス概略

3.2　開発プロセスの詳細

次にそれぞれの詳細な工程を見出し、開発プロセスを策定した。開発プロセスの策定においては、MDA 工程にそれぞれ UML での作成物を当てはめ、かつ、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 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、何を見出すのかの目的を明確にした上で必要なものだけを採用した。また、モデリングの目的によっては 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。本報告においても同様、UML だけでは不足する図や資料は別途作成し、CIM・PIM 作成のための工程と作成モデル、またその関連を図４のように開発プロセスとして定義した。

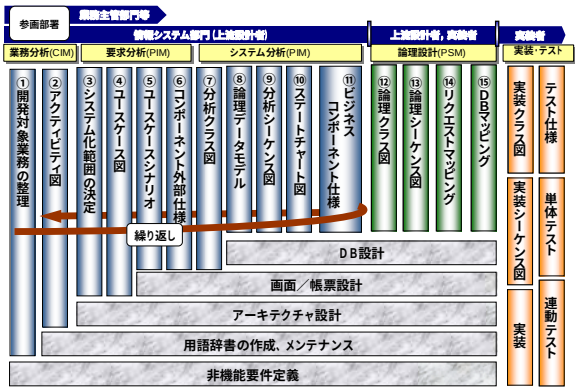

図４　開発プロセス

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５にそれぞれの工程のプロセス関連図を示す。

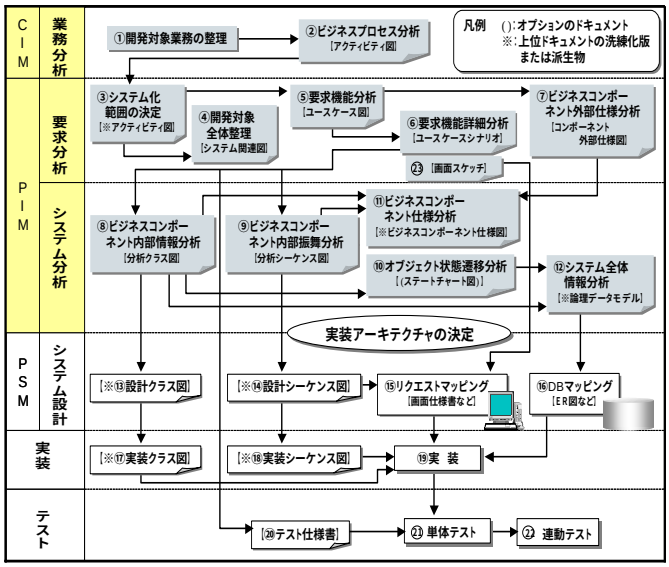

図５　開発プロセス関連図

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工程

で、コンポーネントを導出していくための方法について考えた。

4.1　業務分析工程(CIM)

業務分析では、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。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】

既に準備されている資料の利用や、業務主管部門へのヒアリングにより業務を細分化し、業務機能階層図として整理する。なお、この工程は次のビジネスプロセス分析の結果として導出するケースもある。

(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、アクティビティ図で業務を分割し、分割した業務の構造を(1)の「開発対象業務の整理」の業務機能階層図へ反映する。これにより、業務機能コンポーネント、システム機能コンポーネントの対象を導出する。図６に業務機能階層図への反映を示す。

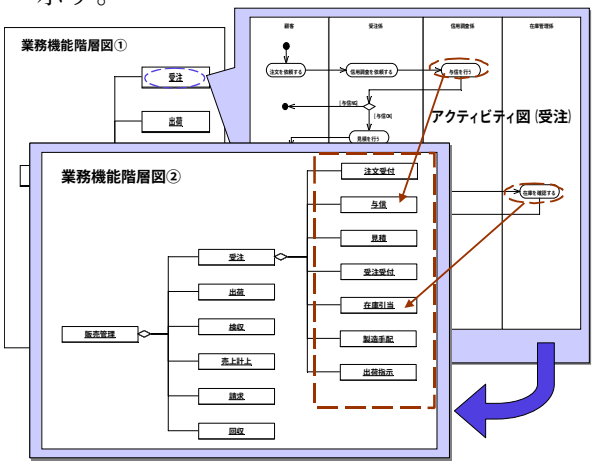

図６　業務機能階層図への反映

4.2　要求分析工程(PIM)

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、システム化範囲の機能を明確にするために、ユーザーや外部システムとの関係を定義した後、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

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絞り込む。

(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、そこでやりとりされる情報の相関関係をシステム全体図として表す。図７にシステム全体図を示す。

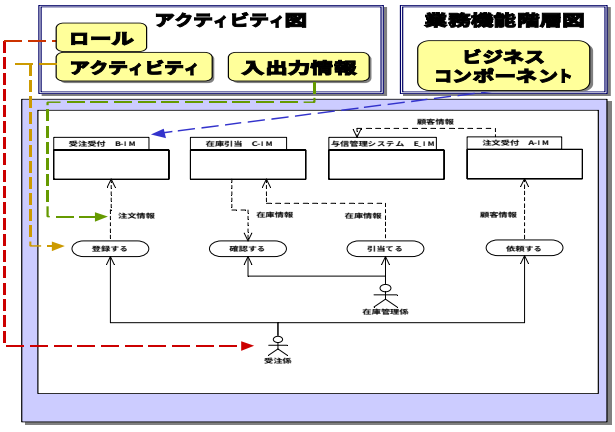

図７　システム全体図

(3) 要求機能の分析【ユースケース図】

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。ユースケース図は下記の観点で作成する。

(a)ユースケース図はシステム化範囲のアクティビティをユースケースとして定義する。

(b)システム化範囲のアクティビティを担当するロールをアクターとして定義する。

(c)ビジネスコンポーネントごとにユースケース図を作成する。

(d)ビジネスプロセスに登場する外部システムや他コンポーネントはアクターとして定義する。

(e)外部システムや他コンポーネントを利用するユースケースはそのアクターと関連付ける。

この工程は次の工程である「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、現在使用している紙の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、処理の内容を詳細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)の「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 8 にユースケース図への反映例を示す。

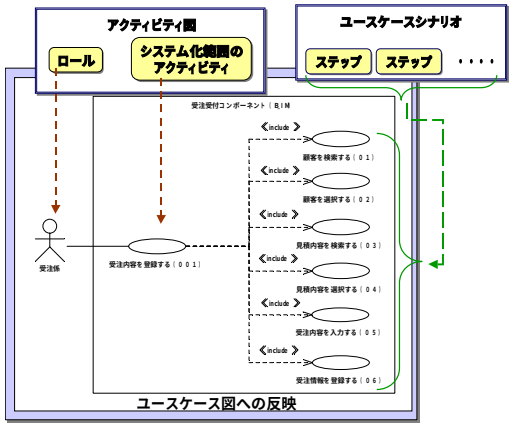

図 8　ユースケース図への反映

(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。ビジネスコンポーネント外部仕様図はインターフェースとオペレーション（メソッド）、入出力される情報（引数と戻り値）を記述する。図 9 に外部仕様図の例を示す。

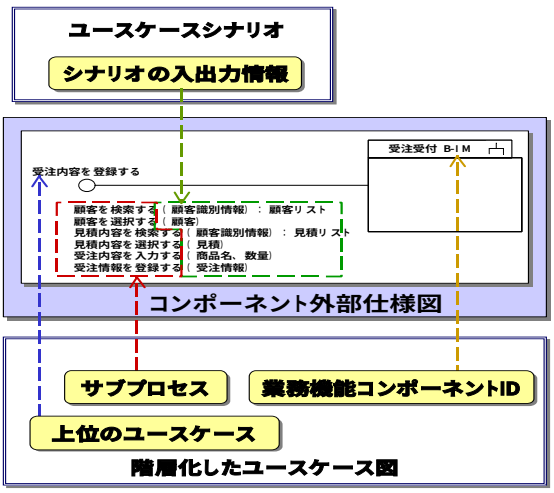

図 9　ﾋﾞｼﾞﾈｽｺﾝﾎﾟｰﾈﾝﾄ外部仕様図

4.3　システム分析工程(PIM)

システム分析では、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。また、そのビジネスコンポーネントのインターフェース、システム化範囲の機能、入出力情報の関係から、シス

テムの内部処理手順を明確にする。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する。最後の工程として、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 10 に分析クラス図を示す。

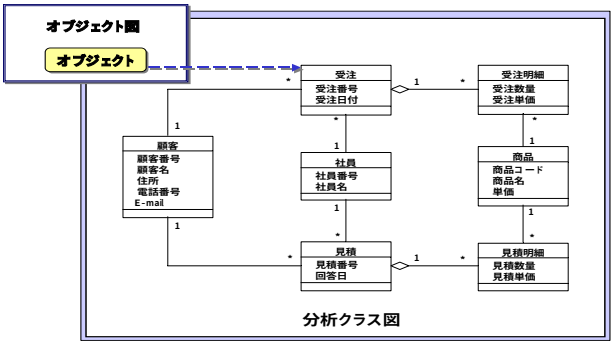

図 10　分析クラス図

(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、ある処理を実現するために行われる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 11 にビジネスコンポーネント仕様図を示す。

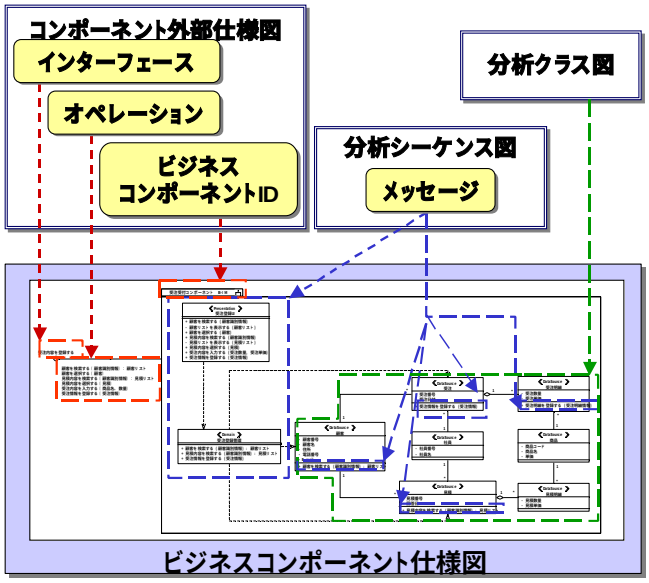

図 11　ビジネスコンポーネント仕様図

(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。ステートチャート図とは、あるオブジェクトが生成され、消滅するまでの状態遷移を表現したものである。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。

以上のように開発プロセスおよびプロセス間の関連、プロセス内でのコンポーネントベースモデリングの開発方法論を策定した。過去に経験したプロジェクトではオブ

ジェクト指向開発や UML モデリングと言ってもクラス図中心で分析した結果を実装し、大規模なシステムでは特に実装者間の意思の疎通が難しいために、コンポーネントの流用がなかなかうまくいかなかった経験を持つ。また、出来上がったコンポーネントはカレンダーや認証など、比較的システム機能に近いユーティリティコンポーネントが中心で、業務寄りのコンポーネントの導出・流用がうまく行かなかった。その要因として業務分析などの上位フェーズからの落とし込みをしなかったことが反省点であった。

今回の方法論のように業務分析の上流でコンポーネントの導出を行えば、それを設計、実装へと繋げていくことは比較的楽にでき開発工数も短縮できる。さらに、そのモデル自身も業務が変わらない限り、流用が可能であり、企業のビジネスモデルとして活用できるはずである。

５．まとめ

今回報告したコンポーネントベースでの開発プロセスと開発方法論の策定について、すでに数社の実際のプロジェクトでこのモデリング手法を適用中であり、コンポーネントベースでの開発が行われている。実際の分析工程では、新規開発システムの中で導出したコンポーネントの 70%以上がプロジェクト内で流用できる結果を見出しているところもある。

現在は、この結果を実装へ持っていくまでの PSM(Platform-Specific Model)工程での方法論を新たな研究内容として纏めており、そちらが完成すると上流から実装までの一貫したプロセスと各工程での開発方法論が出来上がり、より一層、コンポーネントベース開発の効率化を狙えると考える。一方、MDA に基づく CIM・PIM から PSM への自動切り出し、さらにプログラムソースやＤＢ定義パラメタの自動生成など、今後は機械化できる部分がかなりの割合を占めることが予想される。そのためには上流での分析方法論はより一層重要となり、今後とも開発方法論についてのブラッシュアップを続けていく所存である。

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)近藤博次、”よくわかるオブジェクト指向の基本と仕組み”、(株)秀和システム、2003.

(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) Doug Rosenberg、Kendall Scott、”UML オブジェクトモデリング”、(株)ソフトバンクパブリッシング、2002.

(10)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.

(11)(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.

(12)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.

商標および登録商標