社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2006－DD－53 (4)
2006／1／27

コンポーネントベースモデリングにおけるドキュメント体系

浜口弘志 藤井啓詞 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。我々のプロジェクトでは、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)を採用した開発方法論の策定を行ってきた。さらに、システムの汎用性、保守性を考慮し、コンポーネントベースモデリングを実現する開発プロセスの改定を行った。その改定後の開発プロセス詳細と共に、UMLをベースとする各種ドキュメントについて報告する。また、改定後の開発方法論について、その有用性について検討する。

Structure of documents for component base modeling

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

In order to develop information systems 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), and to settle on the development methodology that achieves cooperation from the business model to the application model. Our development methodology was based on reference architecture MDA(Model Driven Architecture) and UML(Unified Modeling Language) those are promoted by international standardization group OMG(Object Management Group). Furthermore, in view of the generality and maintainability of an information system, we revised the development processes that realize a component base modeling. We report various documents based on the UML, with detail of the revised process. Moreover, the utility is examined about the revised development methodology.

1 はじめに

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[1]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。我々のプロジェクトでは、国際標準化団体OMGの推し進める参照アーキテクチャ MDA [2]と共通記述言語 UML1.4[3][4]を採用した開発方法論の策定を行ってきた[6][7]。さらに、システムの再利用性、汎用性、保守性を考慮し、コンポーネントベースモデリングを実現する開発プロセスの改定を行った。これまでの開発方法論では、システム全体のコンポーネント構造を明確にするダイアグラムは規定しておらず、コンポーネントにダイアグラムをUMLで、もしくは独自に規定していた。今回、開発方法論の改定にあたり、UML2.0[3][5]のパッケージ図を成果物として新たに定義した。また、MDA ツールを適用することで、開発工程のシームレスな連携を実現することを目指した。改定後の開発プロセス詳細とドキュメント、そしてその有用性について報告する。

2 開発方法論の前提知識

ここでは、前提知識となるMDAとUMLについて記述する。

2.1 MDAの概略

情報システムを効率良く開発するためには開発方法論の策定が必要である。我々はこれまで開発方法論MDAを採用し、要件定義を行ってきた。

MDAでは図 1に示すように、分析モデル、構造モデル、実装モデルを順に作成してゆく。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はマッピングという作業により変換を行う。マッピングとは、前工程のモデルを特定のプラットフォームや技術に即したモデルに変換することである。これらの変換ルールをマッピングルールと言う。この際、UMLで成果物を作成することにより、MDAツールでの変換が可能となる。

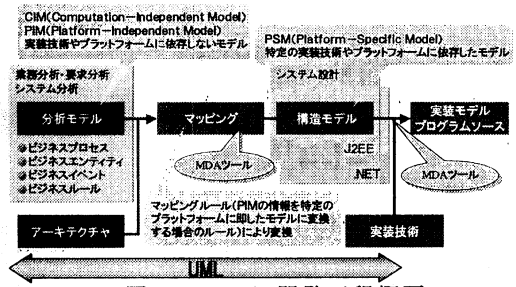

図 1 MDA開発工程概要

なお、分析モデルは業務分析工程、要求分析工程、システム分析工程を経て作成してゆく。構造モデルはシステム設計工程で作成する。各工程の概要を以下に示す。本論文では分析モデルを作成する3つの工程を中心に報告する。

(1)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工程であり、作成するモデルはCIM(Computation-Independent Model)である。

(2)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工程である。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工程であり、作成するモデルはPIM(Platform-Independent Model)である。

(3)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工程であり、作成するモデルはPSM(Platform-Specific Model)である。

2.2 UMLの概略

UMLでは表記法が統一されているため、モデルの翻訳が不要である。また、UMLは純粋な表記法として開発方法論と分離されており、開発規模や内容にあった方法論と自由に組み合わせることができる[8][9]。

UMLを導入することで、以下のような効果が期待できる。

(1)意思疎通が容易になる

発注者と受注者間の情報共有を強化し、システムの品質を高めることができる。

(2)成果物の可読性が高まる

過去のシステムの成果物（ドキュメント）を再利用できる。

(3)表現力が高まる

静的な側面や動的な側面など、多様な視点からモデルを表現できる。

(4)一貫した表現法を利用できる

プロセス間で成果物を追跡でき、各プロセスの開発者間のコミュニケーションが円滑になる。

(5)MDAツールを使用できる

MDAツールの規格統一がなされれば、ツールを使ってマッピングを自動に行うことができる。現在これらのツールはUML1.xベースからUML2.0ベースにシフトしてきている。

3 改定前の開発方法論

ここでは、改定前の開発方法論について、その概要と問題点を述べる。

3.1 開発方法論詳細化の目的

MDAは参照アーキテクチャであり、図 1に示したような概念部分しか定義されていない。我々のプロジェクトでは、コンポーネントベースモデリングを目的として開発方法論の詳細化を行った。そのコンポーネントベースモデリングの実現のために必要な事項を以下に示す。

(1)分析工程での共有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、実装時の工数短縮を実現する。また、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す。

(2)開発プロセス間関連の明確化

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成果物作成を可能とする。

(3)マッピングルールの確立

マッピングルールを定義し、MDA ツールを用いて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[10]。

3.2 コンポーネントの定義

開発方法論の策定にあたり、コンポーネントの定義を行った。定義対象例として、図 2 に製品資料配布管理システムを示す。

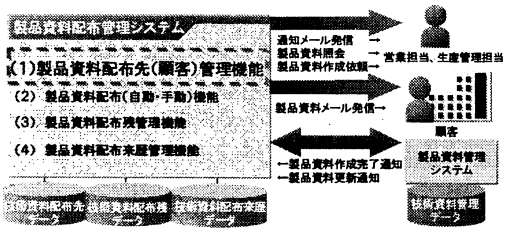

図 2 製品資料配布管理システム

製品資料配布管理システムは、以下に示す 4 つの機能を持つシステムである。

(1)製品資料配布先管理

配布先を管理する機能。

(2)製品資料配布

資料の配布機能。

(3)製品資料配布残管理

配布依頼の状態を管理する機能。

(4)製品資料配布来歴管理

配布来歴を管理する機能。

さらに、(1) の管理機能は以下の 6 つの機能で構成される。

(a) 製品資料配布先登録・更新

(b) 製品資料配布先取得

(c) 連結情報取得

(d) ロール情報取

(e) 顧客情報取得

(f) ユーザー認証情報取得

これらの機能をコンポーネントとして定義し分析を行うことで、システムをコンポーネント階層構造で表現することが可能となる。なお、各コンポーネントレイヤーにおいては、作成する成果物が定義されており、それらの関係と共にコンポーネントの定義を図 3 に示す。

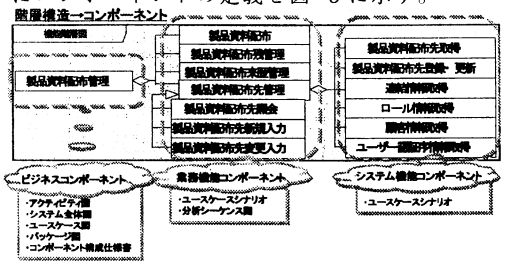

図 3 コンポーネントの定義

図 3 に示すように、それぞれのレイヤを上から順に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。

このように導出したコンポーネントを該当業務及び他業務で共有することで、開発工数の短縮が可能となる。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。

3.3 問題点

これまでに我々が詳細化した MDA の開発プロセスを図 4 に示す。UML1.4 をベースとした成果物の作成プロセスをこのように詳細化した。

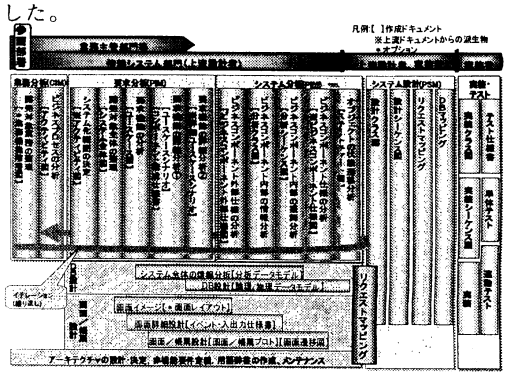

図 4 改定前の開発プロセス

これまでの開発プロセスは UML1.4 ベースであったが、UML1.4 では分析工程でのコンポーネントを表現するのに適した成果物が定義されていなかった。そのため、オプションとして図 4 の破線部に示すビジネスコンポーネント仕様図を定義し、コンポーネントの仕様を記述していた。図 5 にビジネスコンポーネント仕様図を示す。ただし、この成果物には以下のような問題点があった。

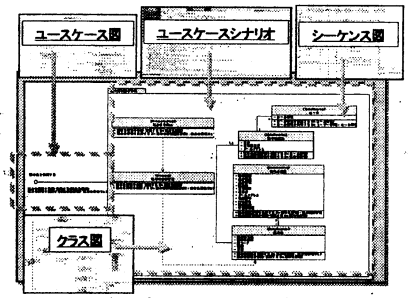

図 5　ビジネスコンポーネント仕様図

(1)コンポーネント仕様の表現方法

コンポーネント仕様図はビジネスコンポーネントのみをMVCモデルで表現する成果物である。そのため、全体のコンポーネント構造を把握するためには、コンポーネント仕様図と図 6に示すようなユースケース図、クラス図、業務機能階層図を参照する必要があった。

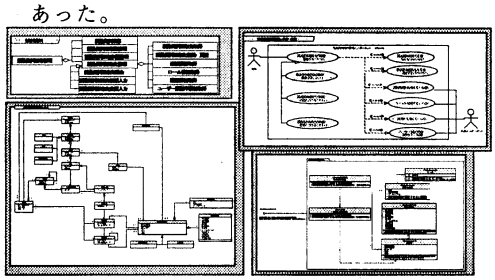
図 6　コンポーネント構造を
定義する成果物（改定前）

(2)静的構造（クラス図）の表現方法

システム全体のクラス図は定義されていたが、コンポーネント単位でのクラス図が定義されていなかった。そのため、設計・実装担当者（デザイナ）の能力によって、設計・実装時にコンポーネント構造の実現が困難な場合もあった。

4　開発方法論の改定

ここでは改定後の開発方法論について、変更点と概要を述べる。

4.1　開発プロセスと成果物の追加

これまでの開発方法論の策定においては、UML1.4をベースとして成果物の定義を行ってきたが、UML2.0において、システムのコンポーネント構造を表現するのに適したドキュメント「パッケージ図」が提供された。

パッケージ図とは、システムを構成する要素をパッケージという単位にまとめ(パッケージング)、その内部構造とパッケージ同士の関係を表現するダイアグラムである。パッケージ図を図 7に示す。

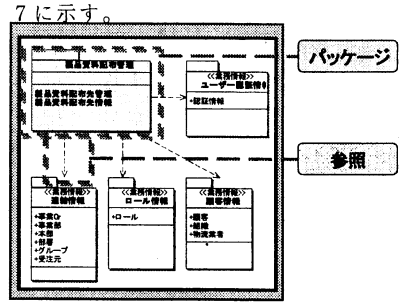

図 7　パッケージ図

そこで、このドキュメントを採用し、3.3節で提示した問題点の解決を図った。

改定後の開発方法論においては、図 6で表現していたコンポーネントの仕様を図 8のパッケージ図で表現する。そのプロセスの詳細化にあたり、パッケージングの方針を検討した。パッケージングの方針は各プロジェクトで分析工程担当者（アーキテクト）を中心に定義する必要がある。その例を以下に示す。

(1)業務機能コンポーネントをパッケージとして定義する。

(2)業務機能コンポーネントが呼び出すクラス群をパッケージとして定義する。

(3)システム機能コンポーネントは(2)のオペレーションとして定義する。

(4)共通機能はパッケージとして定義する。

(5)各パッケージは内部にクラス構造を持つ。

これらの条件からも判断できるように、パッケージ図は階層構造で表される。これらの条件に沿って作成したパッケージ図を図 8のパッケージ図の例に示す。

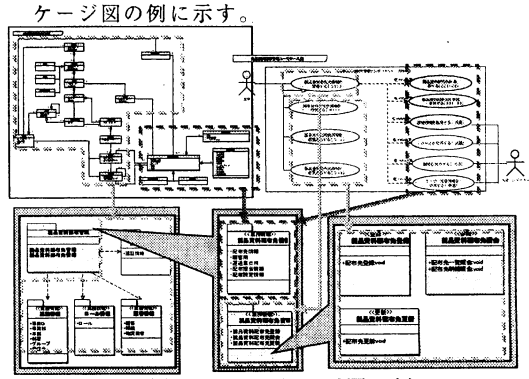
図 8　パッケージ図の例

4.2 開発方法論の改定

図 9に開発方法論の改定(開発工程全体図)を示す。

コンポーネント構造をパッケージ図で表現することで、図 4のようにビジネスコンポーネントの仕様作成プロセスは不要となった。また、クラス図をコンポーネント単位で分析することを明確にした。

改定後の開発プロセスで追加もしくは修正されたプロセスについては破線で示す。

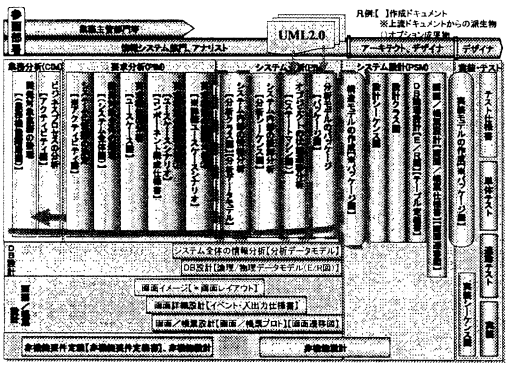

図 9 開発方法論の改定(開発工程全体図)

開発工程全体図の改定に合わせ、開発工程関連図の改定も行った。パッケージ図の作成においては、ユースケース図と分析クラス図を参照することを明らかにし、図 10のように改定を行った。

パッケージ図の作成プロセスを破線で示す。

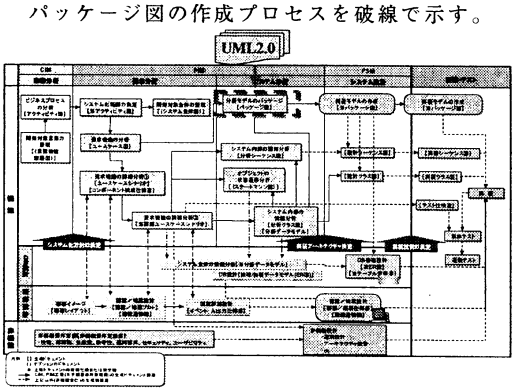

図 10 開発方法論の改定(開発工程関連図)

4.3 開発工程の関与者

弊社では J2EE™ に対応した Java™ アプリケーションの開発・デバッグ環境 uCosminexus Developer において MDA ツール [11] (以降はモデル変換プラグインと呼ぶ)が提供されている。このツールの入力情報はパッケージ図であり、改定後の開発プロセスで採用することで、PIM から PSM、さらに設計・実装へと統一されたマッピングが実現できる。また、MDA では 4.2節で示したように多数の成果物が定義されている。そこで、各工程の関与者を明確にするために図 11のように定義した。

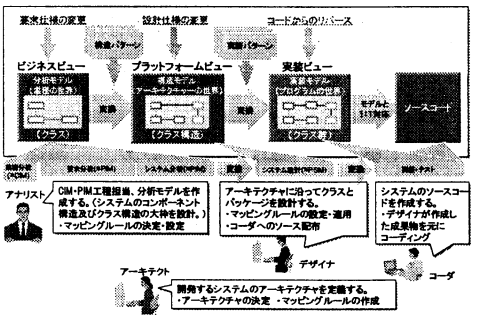

図 11 各工程と関与者

(1)アナリスト

CIM・PIM 作成を担当し、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(構造パターン、実装パターン)を決定・設定する。

(2)デザイナ

システム設計と開発・設計工程を担当し、実装パターンを設定・適用する。ただし、DB 周りの設計等も担当するが、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、構造パターン、実装パターン作成を担当する。

5 MDA ツールの適用

図 2システムの成果物に対して、モデル変換プラグインの適用例をここに示す。

5.1 モデル変換プラグインの適用

図 2のシステムのパッケージ図をここに示す。この成果物は図 9に示したプロセスで作成したものである。

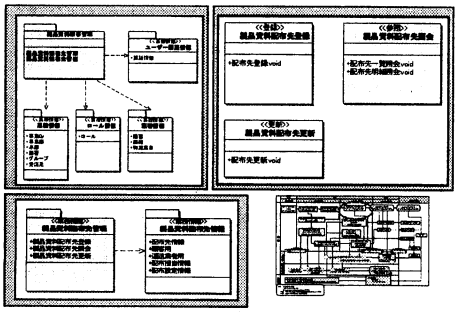

図 12　パッケージ図の例

モデル変換プラグインでは、デザインパターンの組み合わせで構成されるパターンでマッピングルールを定義する。モデル変換プラグインでサンプルとして提供されるパターンを表1に示す。このようなパターン作成をアーキテクトが担当する。

表1　サンプルパターン

#	適用	対象	提供パターン	説明	パターン内容
1	分析モデル(PSM)→構造モデル(PSM)	パッケージ	業務機能	プレゼンテーション層からの共通の処理サービスを提供するパッケージ	ファサード
2			業務情報	永続化情報との入出力を行い、情報管理に責任を持つパッケージ	ファサード
3			外部連携	他の既存システムとのサービス連携に責任を持つパッケージ	ファサード、アダプタ
4			業務方式	業務上のまとまった方式を提供する	ファサード、ステラテジ
5			―	整理	パッケージ内のクラスの属性化を行いクラスの整理を行う
6		クラス	業務機能	参照	キーに従い業務情報への検索指示	キーチェック、検索指示
7				登録	キーに従い業務情報への登録指示	データチェック、登録指示
8				更新	キーに従い業務情報への更新指示	データ取得、チェック
9				削除	キーに従い業務情報への削除指示	キーチェック、削除指示
10			業務情報	組織階層	木構造(連続する一対多関係)を表現	コンポジット
11				コード	コードマスタを初期入力し、保存	シングルトン、プロキシ

変換例として、図 12の製品資料配布先管理パッケージに業務機能パターンを適用した結果を図 13のモデル変換プラグインの適用例（分析モデル→構造モデル）に示す。

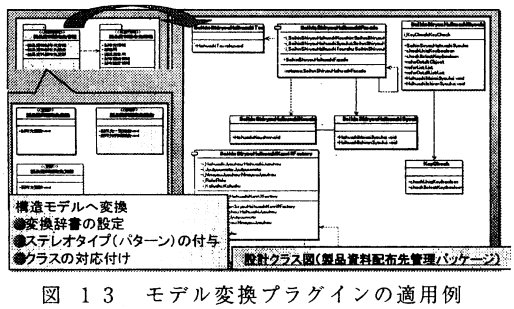

図 13　モデル変換プラグインの適用例
（分析モデル→構造モデル）

同様に構造モデルにもパターンを決定・適用することで、実装モデルを作成することができる。このように、モデル変換プラグインを適用することで、コンポーネント単位の開発が可能となる。また、プロジェクト内で統一されたマッピングを行うことができる。

6 評価

図 2のシステムにおいて、改定後の開発方法論で見出した分析工程のコンポーネント総数と流用率を表2に示す。

表2　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	既存流用数	総数	流用率(%)
ビジネス	4	0	0	4	0
業務機能	12	8	0	20	40
システム機能	9	14	30	53	83

ここに示したように、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。

この分析結果を元に、製品資料配布先管理機能部分において、改定後の開発方法論により再度分析を行った。その結果、改定後の開発方法論でも表2の分析結果に変化はなかったが、成果物の作成数に変化があった。表3に成果物の比較に示す。

表3　成果物の比較

ダイアグラム名称	業務機能階層図	アクティビティ図	ユースケース図	シナリオ	クラス図	コンポーネント構成仕様書	パッケージ図	総数
改定前	1	1	1	7	1	1	0	13
改定後	0	1	1	7	1	1	4	16

ここに示すように、改定後では分析工程において、成果物の数は増え、改定前に比べてより多くの時間がかかってしまうことが判断できる。しかし、以下のようなメリットを得ることができる。

(1)コンポーネントベース開発の実現

MDA ツールを適用することができるため、分析工程のコンポーネント構造を設計・開発工程まで引き継ぐことができる。その結果、コンポーネント単位での開発が可能となる。

(2)マッピングの平準化

MDA ツールを適用することで、プロジェクトで統一されたマッピングを行うことができる。

この結果、システムの保守性や汎用性の向上が実現できる。

このように、開発方法論の改定を行い新たな成果物を定義した結果、コンポーネントベース開発の実現が可能となった。

モデル変換の試行を繰り返したところ、アナリスト、アーキテクト、デザイナの解釈によって変換結果に個人差が出ることが分かった。マッピング作業の平準化を図るためにも、プロジェクト内において分析結果を有効に活用できるようなルールの策定と、その適用指針を明確にすることが今後の課題である。

7 おわりに

UML1.4をベースとしていたこれまでの開発方法論に、UML2.0のドキュメントを組み込んだことで、システム全体のコンポーネント構造を表現することができ、さらにMDAツールの適用が可能となった。この結果、コンポーネントベースモデリングにおいて、分析工程から設計・開発工程までの連携が実現可能であると共に、その有用性を示した。

参考文献

(1) 湯浦克彦、"実践！！エンタープライズ・アーキテクチャ"（株）ソフト・リサーチ・センター、2005.

(2) 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.

(3) Object Management Group、"UML™ Resource Page"、http://www.uml.org/

(4) (株)テクノロジックアート、"独習UML改訂版"、(株)翔永社、2005.

(5) (株)テクノロジックアート、"独習UML第3版"、(株)翔永社、2005.

(6) 安部麻衣、桐越信一、浜口弘志、大場みち子、"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"、情報処理学会研究報告、2004-IS-89、Vol.2004、No.88、pp.1-7 (2004).

(7) 浜口弘志、原口拓也、桐越信一、大場みち子、"MDAによるコンポーネントベースモデリングの実例"、情報処理学会研究報告、2005-IS-93、Vol.2005、No.86、pp.1-8 (2005) .

(8) 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.

(9) 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.

(10) 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.

(11) （株）日立製作所、"ミドルウェア・プラットフォームソフトウェア総合サイト"製品紹介「uCosminexus Developer」：http://www.hitachi.co.jp/Prod/comp/soft1/cosminexus/index.html

商標