ソフトウェア工学 140－15
（２００３．３．７）

MDA に基づくソフトウェア開発と従来手法の比較, 及び実適用へ向けての考察

安東孝信† 山城明宏† 神谷慎吾†† 山本修一郎†† 峰岸巧††† 永田守男†††

OMG は MDA と呼ばれる技術を発表し，またそれに用いる表記として UML Profile for EDOC をまとめている．これらは仕様としてまとめられているが，効果的な利用方法はまだ明らかにされていない．そこで我々は，簡単な業務システムを対象に， 実際に設計モデルまでを構築する試作を通して，MDA や UML Profile for EDOC に基づくシステム開発のノウハウの収集や実現可能性の検討を行い，実適用への足がかりを得ることができた．本論文では，この試作を通して得られた知見から，ソフトウェア生産技術に関する従来手法との関係/比較や，実適用に向けて今後解決せねばならない課題等について考察する．

Feasibility Study for MDA Based Software Development

TAKANOBU ANDO†, AKIHIRO YAMASHIRO†, SHUICHI YAMAMOTO††, SHINGO KAMIYA††,
SATOSHI MINEGISHI††† and MORIO NAGATA†††

OMG published MDA as a new software development technology. OMG is specifying UML Profile for EDOC as a tool for the MDA. However, effective use of them has been unclear yet. So we have made a feasibility study using these new technologies through actual development of a simple business application. We got some clues for practice using these new technologies through the study. Based on the result, this paper describes relations and comparisons between these new technologies and some typical existing software development technologies. In addition, some left unsolved problems for practice picked up from the study are described.

1. はじめに

OMG(Object Management Group)が提唱する MDA(Model Driven Architecture)[1] は, プラットフォームの変化に強く, モデルの実行可能性検証なども含まれる, 新しいソフトウェアの開発の手段として，現在注目を集めている．

ソフトウェアが利用される様々な分野において，近年の技術進化や実行環境の進歩/変遷が極めて速いことを考えると，MDA の有効性は十分に期待できるものであると考えられる. しかしながら現時点での MDA の普及は, 組込みドメインや監視/制御ドメインなどの限られたドメインへの適用[2]に留まっている感がある．これらのドメインの特徴としては, 実行可能性検証を可能にするモデル表記や CASE ツールの充実などが挙げられる．これらの点が，MDA をこのドメインへ最初に普及させた理由であると考えられる．

一方でソフトウェア開発の一つの大きなドメインが業務アプリケーションドメイン(情報系システムや基幹業務系システムなど)である．昨今の業務アプリケーションドメインのシステムは大規模化の傾向にあり, たいていは様々なプラットフォームが混在する大規模分散システムにな

† 株式会社 東芝
TOSHIBA Corporation
†† 株式会社 NTT データ
NTT DATA Corporation
††† 慶應義塾大学
KEIO University

る．従って，このドメインへも MDA を適用することが望まれている．

分散システムのモデル化を行う場合，現在の UML(Unified Modeling Language)[3]だけでは足りない側面もある．この理由から現在，分散業務システムをモデル化するための表記として OMG では UML Profile for EDOC [4]が仕様化されようとしている．

UML Profile for EDOC を用いることで分散業務システム開発への MDA 適用が促進することが期待できる．しかしながら，UML Profile for EDOC は非常に新しい技術であり，その具体的かつ効果的な適用方法に関してはまだ不明な点が多い．いくつかの日本語のガイド[5][6]も公開されてはいるが実適用を考慮すると，完全なものはない．また，MDA 自身もソフトウェア開発方法論の側面では今だ十分に成熟しているとは言い難い状況である．

このような現状を鑑み，我々は MDA/UML Profile for EDOC を用い，簡単な分散業務システムの試作を通し，ノウハウの収集や実適用に向けての検討を行う研究を行った．本論文ではこの研究活動の成果の中から特に，この MDA/UML Profile for EDOC を用いた新しい開発手法と，これまでに存在した様々なソフトウェア生産技術との関係や比較について述べ，また MDA/UML Profile for EDOC を実際の開発へ適用するにあたってまだ残されていると考えられる課題について検討/考察する．

本論文の構成は次の通り．2 章では MDA と UML Profile for EDOC について概説し，我々の解釈におけるこれらに基づく開発プロセスを示す．3 章では今回の研究にて行った試作の内容について簡単に紹介する．4 章では MDA/UML Profile for EDOC に基づくソフトウェア開発と，従来のソフトウェア生産技術の関係/比較について述べる．5 章では MDA/UML Profile for EDOC を実適用する際に起こりうる問題点について，検討/考察する．最後に 6 章でまとめる．

2. MDA と UML Profile for EDOC

MDA は OMG が進める「新しいアーキテクチャの標準化」であり，UML Profile for EDOC は MDA による開発をサポートする UML Profile の一つである．本章ではこれらについて概説する．また UML Profile for EDOC を用いた開発プロセスについても述べる．

2.1. MDA

MDA は，ソフトウェア/システムの全ライフサイクル(設計から配備，管理，他との統合)をサポートし，プラットフォーム技術の変化に対応することのできる堅牢なフレームワークを提供することを目指している．これを達成するために MDA では，各種の仕様をプラットフォームに依存しないモデル(PIM : Platform Independent Model)から，各種プラットフォーム技術に特化したモデル(PSM : Platform Specific Model)へとマッピングすることを通して，ソフトウェア開発を行う．つまり，モデルベースのシステム開発手法である．

2.2. UML Profile for EDOC

UML Profile for EDOC は，RM-ODP[7]のフレームワークに基づく 5 つの Viewpoint(図 1)でターゲットシステムをモデル化することを可能にする UML Profile である．この各 Viewpoint のモデルを構築するために，UML Profile for EDOC は特定の技術に非依存な ECA (Enter-

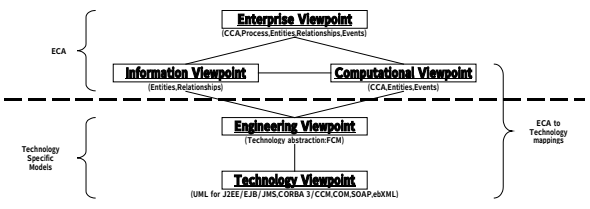

図 1 RM-ODP の 5 つの Viewpoint

prise Collaboration Architecture)と呼ばれるプロファイルと, パターンプロファイル, 特定の技術に特化するモデルセット, から構成されている. また ECA はさらに CCA (Component Collaboration Architecture), エンティティプロファイル, イベントプロファイル, ビジネスプロファイル, リレーションシッププロファイルの 5 つのサブプロファイルで構成されている.

以下では, 各 Viewpoint について概説する.

Enterprise Viewpoint

この Viewpoint ではビジネスドメイン, ビジネスプロセス, システム設計の間の本質的なトレーサビリティを提供するために, 下記の概念を用いてビジネス組織のコンテキストにおけるシステムの構造や振舞をモデル化する.

- システムによって支援されるビジネスプロセス
- プロセスにおけるステップと, ステップ間の関係
- ステップに適用されるビジネスルール(ポリシー)
- ステップごとの生成物(artifact)
- ビジネスエンティティを表すエンタープライズオブジェクト
- ビジネスプロセスに関係する役割(role)
- ビジネスプロセスの各ステップの責任を明らかにするための役割間の関係

UML Profile for EDOC では, ECA の全てのプロファイルを用いてこれらをモデル化する.

Computational Viewpoint

この Viewpoint では Enterprise Viewpoint で定められたシステムの役割を果たすための実装を, 下記を定義し, システムの機能をコンピューテーショナルオブジェクトに分解することで記述する.

- システムのある機能的役割を果たすインタフェースを持ったコンピューテーショナルオブジェクト
- コンピューテーショナルオブジェクトのインタフェース
- コンピューテーショナルオブジェクト間の協調関係

EDOC では CCA を用いてこれらをモデル化する. コンピューテーショナルオブジェクトは ProcessComponent と呼ばれ, その表現する抽象度のレベルに応じて E-Business, Application, Distributed, Program の 4 種類の抽象レベルのコンポーネントに分類されている.

Informational Viewpoint

この Viewpoint では情報のセマンティクスとビジネスプロセスにおける情報の処理を, 下記によって表現する.

- 情報オブジェクトの構成(静的スキーマ)
- 情報オブジェクトの振舞(動的スキーマ)
- 上記に適用される制約(不変スキーマ)

UML Profile for EDOC では, エンティティプロファイルやリレーションシッププロファイルを用いてこれらをモデル化する.

Engineering Viewpoint

この Viewpoint ではシステムに必要な分散透過性やそれを実現するために必要なサービスを定義する. また下記の件などを決める.

- ネットワークアクセスされるオブジェクトとそうでないオブジェクトの分離
- トランザクションや非同期メッセージングの考慮
- 永続性とその実現手段

UML Profile for EDOC では, Computational Viewpoint のモデルからテクノロジに非依存なモデルへのマッピングによって導き出される.

Technology Viewpoint

この Viewpoint では, システム実現のためのソフトウェアやハードウェアの選択/配置や, 特定のテクノロジへのマッピングに基づいてモデル化が行われる.

2.3. UML Profile for EDOC を用いた開発プロセス

UML Profile for EDOC を用いた開発プロセスは明らかにされていない. そこで我々は, Viewpoint の定義や各 Viewpoint 間の関係に着目し, また試作を通して, 図 2に示すような開発

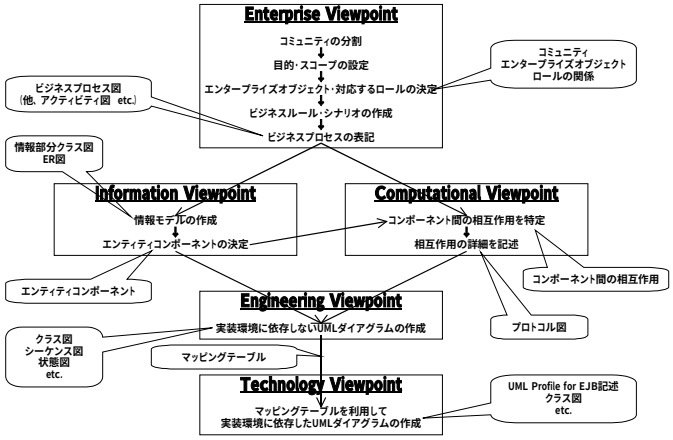

図 2 開発プロセスと成果物のまとめ

プロセスをまとめた[8]．図中の矩形が各 Viewpoint を表し, その内部のフローは開発のステップを示している. また吹き出しは成果物を表している．詳細は別途報告[8)22)]する.

3. 適用事例

MDA に基づき UML Profile for EDOC を用いて,「在庫管理における照会業務」という具体的事例を対象に，各 Viewpoint のモデル化を行った[8]．その一例として，Enterprise Viewpoint のモデルとして作成したビジネスプロセス図の例を図 3に示す．詳細は別途報告[8)22)]する.

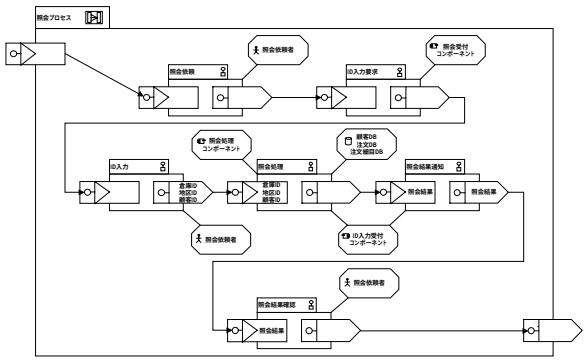

図 3 ビジネスプロセス図の例

4. 従来手法との関係/比較

3章で示した実適用における結果明らかになった課題の一つとして，各 Viewpoint のモデル化をいかに行うかが不明瞭である点が挙げられた. これを解決するための手段の一つは他のソフトウェア生産技術との連携であると考えられる. そこで本章では, 既存の代表的なソフトウェア生産技術と，2.3節で述べた開発プロセスも含めた MDA/UML Profile for EDOC に基づく開発手法(以後 MDA/EDOC 手法と呼ぶことにする)との関係について比較/検討を行う.

4.1. 構造化手法との関係/比較

構造化手法[9)]は，1970 年台後半から，大規模化するソフトウェアの「ソフトウェアの複雑さ」を扱うために生まれたソフトウェア開発手法である. 機能中心にシステムを分割し，トップダウンに分析作業を進め, システムをいくつかの視点からモデル化する，という特徴を持っている. 後にリアルタイムシステム向け[10)]，など様々な発展形が開発された.

構造化手法ではソフトウェアを構造モデル, 機能モデル，動的モデルの 3 つの視点で表現する. 複数の視点からシステムを捉えるという点で, MDA/EDOC 手法と似ているが, 各視点の意義がそれぞれで異なっている．MDA/EDOC 手法では，システム開発のプロセスに従って視点(Viewpoint)が変遷する(2.3節)と考えられる．これに対し，構造化手法の各視点は，開発プロセス

の観点からは基本的には同一のプロセス内に存在し，開発を進める上で，それぞれが詳細化/細分化されていく. 従って視点の観点では直接比較することはできない.

ところで, 構造化手法の最も大きな特徴は機能分割である．MDA/EDOC 手法では，Enterprise Viewpoint におけるビジネスプロセスモデルの構築や，Computational Viewpoint のコンピューテーショナルオブジェクトの抽出は, 実は機能分割的に行われる．つまりこれらの Viewpoint におけるモデル化では, 構造化手法の考え方が効果的であると考えられる．また，このことを逆に捉えると, 構造化手法はソフトウェアの細部を分析/設計するという観点では，仕様変更に対する柔軟性や保守性, 開発時の後戻り容易性などの点からオブジェクト指向開発手法に取って替わられた感が強いが，より抽象度の高い Enterprise Viewpoint や Computational Viewpoint でのモデル化においては，むしろ構造化(機能分割)の考え方が重要になるのではないかと考えられる．

4.2. オブジェクト指向開発手法との関係/比較

オブジェクト指向開発手法[12)13)14)](以後 OO 手法と呼ぶ)は，1980 年台後半から，構造化手法の問題点を克服するために生まれてきたソフトウェア開発手法である. ソフトウェアシステムをオブジェクト(基本的にはデータと振舞を併せ持つもの)の構成で表現し，ラウンドトリップ型の開発をサポートし, システムをいくつかの視点でモデル化する，という特徴をもっている.

OO 手法でも, 構造化手法と同じくソフトウェアを構造/機能/振舞の 3 つの視点で表現する．こちらも4.1節と同じく，MDA/EDOC 手法の各 Viewpoint と直接比較することはできない.

OO 手法は構造化手法に比べ, 基本的にはシステム内部の記述に様々な点で優れているといわれている[15)16)]．MDA/ EDOC 手法における Viewpoint のうち, システムの内部構成をモデル化する視点は Information，Engineering，Technology の 3 つの Viewpoint である．これらの視点のモデル化では OO 手法が有効であると考えられる．また Enterprise Viewpoint や Information Viewpoint でもビジネスオブジェクトや概念オブジェクトの抽出にも OO 手法の考え方は有効であると考えられる．従って，OO 手法は全ての視点で関係が深いと考えられる.

4.3. データ中心アプローチと関係/比較

データ中心アプローチ(DOA)[17)]は，変更が頻繁な業務プロセスに対して比較的安定しているデータ設計を中心にシステム設計する手法である．DOA では基本的に ER 図によってデータ設計を行う．MDA/EDOC 手法の Information Viewpoint で表現するものと同じと考えられるので，この Viewpoint のモデル化においては DOA の考え方が有効ではないかと考えられる.

4.4. RUP 4+1View との関係/比較

RUP(Rational Unified Process)では，4+1 View と呼ばれる 5 つの視点(ユースケース, 配置, 論理，プロセス，実装の 5 つのビュー)を用いてシステムのアーキテクチャを定義する.

これらの視点はシステム開発のプロセスに従って視点が変遷する点は, MDA/EDOC 手法と同様である．ユースケースビューは Enterprise Viewpoint や Computational Viewpoint に，配置ビューは Enterprise Viewpoint に，論理ビューは Information や Engineering Viewpoint に，プロセスビューは Engineering Viewpoint に，実装ビューは Technology Viewpoint に，それぞれ関係していると考えられ, これらの点については, 両者は似ている. これに対し, Computational Viewpoint におけるコンポーネント間のプロトコルのモデル化など, 分散システムを強く意識したモデル化の視点は RUP 4+1 View には含まれ

ておらず, MDA/EDOC 手法の大きな特徴であると考えられる.

4.5. ソフトウェアパターンとの関係/比較

ソフトウェア開発において, 現在最も注目されている技術要素の 1 つがソフトウェアパターン[18]である. 様々な分類方式があるが, 本論文では主なものを表 1に示すように分類する.

表 1 ソフトウェアパターンの分類

パターン分類	概説
システムアーキテクチャ	システム構成を表す
アナリシスパターン[19]	業務の概念構造を表す
アーキテクチャパターン[20]	システムアーキテクチャを表す
デザインパターン[21]	設計の特定の問題を解決
フレームワーク	実装を含む
イディオム	実装のノウハウ

MDA/EDOC 手法の各 Viewpoint に照らし合わせていくと, システムアーキテクチャは Enterprise や Computational の Viewpoint へ, アナリシスパターンは Enterprise や Information の Viewpoint へ, デザインパターンは Engineering Viewpoint へ, フレームワークやイディオムは Technology Viewpoint へ, それぞれ適用可能ではないかと考えられる. アーキテクチャパターンについては基本的にはオブジェクトの抽出の指針を与えるものであることから Engineering Viewpoint へ適用可能と考えられる. しかし, 例えば代表的な MVC アーキテクチャを考えると, Model は Information Viewpoint に, Controller は Computational Viewpoint に関係する. View については直接関係する Viewpoint は存在しない. このことから, アーキテクチャパターンはその種類/性質に応じて各 Viewpoint に適用できるものと, そうでないものがあると考えられる.

4.6. 比較/検討のまとめ

前節までに述べた内容をまとめると図 4に示すようになる. 3章で示した実適用を通して明らかになったが, MDA/EDOC 手法では, 各 Viewpoint の定義や各 Viewpoint 間の関係については明示的に示されているが, 各 Viewpoint で開発/洗練するモデルの構築手法が不明瞭であった. この図に示すように他の開発手法との対応関係を明示することによって, 各 Viewpoint でのモデル構築手法を既存の手法でカバーすることができるようになり, 実適用の際の一つの指針として役立つのではないかと考えている. ただし具体的にどのように適用すべきかは今後の課題である.

5. 実適用へ向けての考察

MDA や UML Profile for EDOC の仕様だけでは, 実適用へのイメージを掴むことが難しい. 今

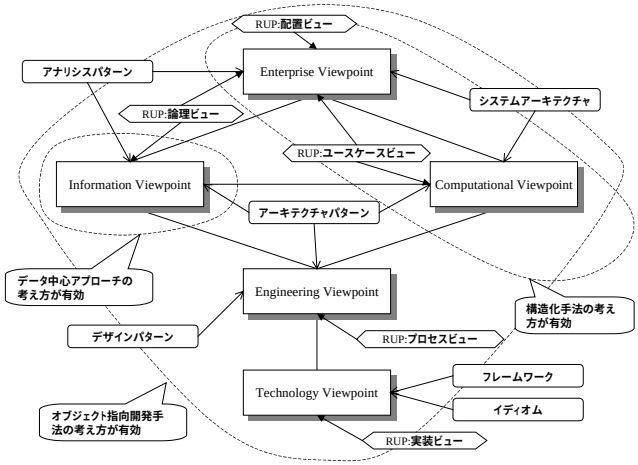

図 4 MDA/EDOC 手法の各 Viewpoint と他のソフトウェア生産技術の関係

回の研究でも3章で示した試作を通し，MDA / EDOC 手法を適用する上で解決せねばならない多くの課題が残されていることが分かった. いくつかは議論を通して我々なりの解釈を得ることができ, 別途具体的に報告された開発プロセスとしてまとめた[8]が, 十分な解決策を得られなかった課題もある. 本章ではそれらの中から主なものについて挙げ，検討/考察する．なお，今回の試作では露見しなかった課題も多く残されていると考えられるが, それらの発掘については今後の研究課題としたい.

マッピング

MDA の効果を現実化するための一つの重要な技術はモデル間のマッピング技術である．UML Profile for EDOC では，Computational Viewpoint から Engineering Viewpoint への技術に非依存なマッピングと，Engineering Viewpoint から Technology Viewpoint への特定の技術に特化したマッピングの 2 種類があることが明記されている．しかしながら, 技術に非依存なマッピングは具体的には十分に述べられておらず，また技術特化のマッピングも現状では J2EE へのマッピングがまとめられつつある程度である．そのため，MDA/EDOC 手法の恩恵はまだ十分には引き出せない現状である. この課題を解決するためには仕様の充実を待つだけでなく，MDA/EDOC 手法のユーザ(ソフトウェア開発者)が，自分の必要とするマッピング技術を開発していかねばならない. さらに抽象化するとマッピング技術を作るための手法も必要である.

CASE ツールによるサポート

MDA はモデルの変換によって開発の効率化やモデルの再利用ができることが一つの特徴である．これを実現するには前述のマッピングに加え，それを機械的に支援する CASE ツールが必要である．現在，MDA 対応を謳う CASE ツールは 50 近く存在しているが，ツールの機能性や安定性, サポートの有無, 将来性などに関する不安や, そもそもそれらの情報量が少ないことからツールの選定が困難であること, などから実際に利用するにはまだ壁が高いと思われる. この問題を解決するには, これらの点の改善が求められると共に，ユーザも積極的な CASE ツール調査や導入を意識しなければならない.

プラットフォームの定義

MDA では，プラットフォームは主として分散通信環境(CORBA や EJB, DCOM など)のことを指しているように思われる. しかしながら実際の開発現場においてはその他の技術要素がプラットフォームになることもありうる. 例えばデータベースなどのミドルウェアや GUI などのライブラリ，監視/制御ドメインではデバイスなどもプラットフォームとして捉えることができる. 従って, プラットフォームをより広義に捉え, 整理し, それぞれのマッピング技術を用意することで, MDA に期待されている効果をより広い範囲で得ることができるようになるのではないかと思われる.

ユーザインタフェースのモデリング

UML Profile for EDOC の Viewpoint に基づいて開発する場合, ユーザインタフェースのモデル化に関する言及がない．例えば，Enterprise Viewpoint におけるビジネスプロセスに ResponsibleParty が接続されている場合，そこにユーザインタフェースが存在することになる. 従って, その段階でセマンティックなレベルのモデル化は出来そうだが，物理的配置/構成をモデル化することなどは考慮されていない. この点についての詳細は別途報告[22]される.

その他

ここまでに挙げた内容に加え, Activity 抽出の指針, 既存のデータベーススキーマを利用した場合のそれ以外の Entity との抽出/関連付け，プロトコル定義における方向性，ProcessComponent

の粒度(抽象度)と各 Viewpoint の関係，Engineering 以降の Viewpoint での設計単位，今回の試作では利用しなかった UML Profile for EDOC 内のその他のプロファイルの有効利用方法，など検討しなければならない細かな内容は多々残されている．これらについては今後の研究課題である．

6. おわりに

今回の研究では，MDA/UML Profile for EDOC を実際に用いたシステム設計を通し，その効果を確認するとともに，これらを用いた開発手法がどのようなものであるか，ということを我々なりに解釈し，開発プロセスなどをまとめることができた[8)22)]．また，本論文ではこれらの成果に基づき，既存のソフトウェア生産技術との関係や，残された課題について述べた．これらの成果から，MDA/UML Profile for EDOC の実適用への足がかりが得ることができたのではないかと考えている．

MDA が目標としていることは真新しいことではなく，オブジェクト指向技術者であれば，おそらく普通に行ってきたことである．しかしながら，MDA の考え方，およびそれを支援する仕様/ツール群が提供されたことで，より広い開発者層で，再利用性の高い，プラットフォームへの依存性の低い，モデルベースの開発が行われるようになると考えられる．

今後は5章で示した課題の解決や，MDA を支援するツール群の有効利用，実行可能性検証などを考慮にいれながら，実際の大規模分散システム開発への適用を視野に入れたモデルベースの開発手法について研究開発を行っていきたい．

参考文献

1) OMG Architecture Board ORMSC : Model Driven Architecture(MDA) (2001)
2) Anirudda Gokhale, and et al : CoSMIC: An MDA Generative Tool for Distributed Real-time and Embedded Component Middleware and Applications, OOPSLA 2002 Workshop (2002)
3) OMG : OMG Unified Modeling Language Specification (2001)
4) OMG : UML Profile for Enterprise Distributed Object Computing Specification (2002)
5) INTAP オープン分散処理委員会：平成 13 年度 RM-ODP と UML Profile for EDOC の適用ガイド～Enterprise Modeling を中心に～，INTAP (2002)
6) 橋本大輔：UML Profile for EDOC チュートリアル，UML PRESS vol.1(p158-167)，技術評論社，(2001)
7) ISO/IEC : Information technology - Open Distributed Processing - Reference Model : Architecture, ISO/IEC 10746-3 (1996)
8) 峰岸巧 他：MDA に基づくソフトウェア開発の事例と開発プロセス，第 140 回ソフトウェア工学研究会 (2003)
9) DeMarco, D. : Structured Analysis and System Specifications, Prentice Hall (1979)
10) P.T.Ward, S.J.Mellor : Structured Develop-ment for Real-Time Systems, Prentice Hall (1985)
11) Derek J. Hatley, Imtiaz A. Pirbhai : Strategies for Real-Time System Specification, Dorset House Publishing (1987)
12) J.Rumbaugh and et al. : Object-Oriented Modeling and Design, Prentice Hall (1990)
13) Booch, G. : Object-Oriented Analysis and Design with Applications. Second Edition, Benjamin Cummings Publishing Company (1994)
14) Jacobson, I. and et al. : Object-Oriented Software Engineering : A Use Case Driven Approach, The ACM press, A Division of the Association for Computing Machinery (1992)
15) 本位田真一，山城明宏：オブジェクト指向システム開発，日経 BP 社 (1993)
16) A. Yamashiro and et al : Comparison of OOA and Real-Time SA - From the Experiment of Analyzing an Image Filing System, OOPSLA 1993 Experience Report (1993)
17) 斎藤孝：リレーショナルデータベース教科書，SRC (1999)
18) 鈴木純一 他：ソフトウェアパターン再考，日科技連 (2000)
19) Fowler M. : Analysis Patterns:Reusable Object Models, Addison-Wesley (1997)
20) F. Buschmann and et al : Pattern-Oriented Software Architecture - A System of Patterns, WILEY (1996)
21) Gamma, E. and et al : Design Patterns, Addison-Wesley (1994)
22) 神谷慎吾 他：業務アプリケーション開発への UML Profile for EDOC の適用法の提案，電子情報通信学会 知能ソフトウェア工学研究会 (2003)