ソフトウェア工学 103−22
（1995．3．10）

オブジェクト指向開発におけるドキュメント量の問題と解決

入内島 裕子*，吉田 和樹*，山城 明宏*，佐藤 実**
*(株)東芝 研究開発センター **(株)東芝 青梅工場

本報告は，オブジェクト指向手法の実用化に関する改良提案である。改良のポイントは，開発を通して表記を一貫させる，手順を明確にする，作業量を削減するという三点である。我々は，それら三点の改良のため，複数の既存オブジェクト指向手法の表記を組み合わせ，簡略化するとともに，個々の仕様書の開発上の位置付けを明らかにした。また，仕様書全体を開発用と保守用に分離し，各開発工程で作成すべき仕様書を特定することで，一度に作成するドキュメント量を削減した。本報告では，我々が改善したオブジェクト指向手法の運用方法を示し，その適用によって既存のオブジェクト指向手法と比較してドキュメント量が半減した効果を示す。

Notation and Process of Object-Oriented development method for saving volume of specifications

Hiroko Iriuchijima　Kazuki Yoshida　Akihiro Yamashiro　Minoru Satoh

Research & Development Center, Ome Works,TOSHIBA Corporation
70, Yanagi-cho, Saiwai-ku, Kawasaki-shi 210, JAPAN

This report describes practical notation and process of Object-Oriented development method. Our aims are as follows: To describe specifications throughout development with the same notation, to show every process of development apparently, and to save volume of specifications properly. In order to obtain these three aims, first of all we expanded the notation and process of the existed method. Then we refined them later. As a result, we got the practical OO-mehod, with which we are able to describe specifications of target system fully and smoothly, in spite of the half volume of specifications compared with the existed method.

1 はじめに

ＧＵＩの採用などシステムの高機能化に伴い，ソフトウェアが複雑，巨大化するとともに，長期間保守する必要が生じている。またC++などの新しい実装言語の登場に伴い，それを反映させる表記の必要性が高まっている。これに対して，我々は実用的な手法の確立を目指している。具体的には手法の適用によってＧＵＩやC++を前提とする開発が円滑に進められ，しかも開発作業は妥当な量であり，さらに保守しやすくするような支援を備えた実用的な手法を確立したいと考えている。

これまでに我々は開発や保守に有効だと思われる手法を探して適用し，一般に主張されている効果が実際に得られるかどうかを調べた。

具体的には，まず記述力豊富な表記と明確な手順を備えることを基準として手法を探した。その結果，既存のオブジェクト指向手法の中でＯＭＴ法を選択した。それはＯＭＴ法が，従来の開発方法と同様にシステムを多角的に記述できる表記と，仕様化方法の見本となる実践的な手順を備えているからである。

次に保守しやすさを調べるため，実際にＯＭＴ法の適用結果を機能拡張した。すると拡張前後で修正箇所が局所化されることが分かった[4]。

さらに開発のしやすさを調べるため，ＯＭＴ法を以下三つの基準で評価した。

・開発を通して表記は一貫しているか
・開発を通して手順は明確か
・作業量は妥当か

その結果，ＯＭＴ法で分析すれば，関係者全員が共通の充実した表記を用いて決定事項や検討過程を記録でき，情報の共有や交換(レビュー)を実施できることが分かった[3]。ところが(基本／詳細)設計の手順は曖昧だったため，開発者自身が何をどの順番で仕様化すれば良いかを決めなければならなかった。また開発中に作成すべき仕様書が種類，量とも多いため，情報収集よりもむしろ書き留めるために多大な労力を必要とした[2]。

以上より，我々は既存のオブジェクト指向手法を，分析支援は充分だが，分析結果から実装への移行性がさほど良好でなく，作業量に問題がある手法であると評価した。

本報告の構成は以下の通りである。第二章では適用対象について述べる。第三章では既存オブジェクト指向手法の課題を述べ，第四章ではその課題を解決するという本報告の目的を明らかにする。第五章では方針を示す。第六章では経過と結果を示す。最後に第七章でまとめる。

2 対象:画像ファイリング装置

我々はオブジェクト指向手法の効果を顕著にするようなシステムを対象として探した。ここでは開発規模が大きく，マルチメディアに通じるような複雑なデータを扱うシステムが対象として適切だと考えた。そして画像ファイリング装置を選んだ。この装置は数枚の画像から成る書類を管理する装置であり，登録，検索，表示という三つの業務をこなす。登録とは一冊の書類にタイトルとして複数のキーを付けて保存する業務である。この装置は検索の効率を上げるため，書類はバインダという単位で，バインダはキャビネットという単位でグループ化して管理する。検索とはシステムの利用者から与えられた検索条件と書類のタイトルとを比べて，同一タイトルが付けられた書類(該当書類)を書類箱に集める業務である。表示とは既に登録された書類に関してシステムの利用者から指定された頁を抜き出し，対応する画像を画面表示する業務である(Fig 2.1参照)。

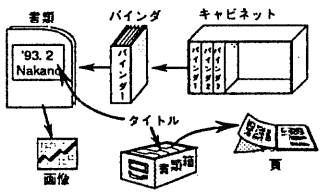

Fig 2.1 対象: 画像ファイリング装置

3 オブジェクト指向手法の課題

当初ＯＭＴ法に着目した頃，我々はＯＭＴ法が開発に有効だと思っていた。それはＯＭＴ法の表記や手順が開発を通して十分であるように見えたからである。また実際に適用してみると，システムへの要求を十分に分析できることが分かった。但し，分析結果に基づく実装を行うため，基本／詳細設計の仕様を決めようとした時，以下三つの問題が明らかになった。

一つは表記についてである。基本設計に関しては，専用の表記はなく，記述例も示されていなかった。詳細設計に関しては，型や情報隠蔽など実装言語の機構を直接反映するような表記要素が不足していた。

もう一つは手順についてである。基本／詳細設計の手順については，分析手順のきめ細かさとは対照的にほとんど具体性がなかった。

さらに作業量についても問題があった。分析においては開発者が作業を円滑に進められ，仕様書の作り直しは少なかったが，出来上がったドキュメントの総枚数はかなり多かった。このドキュメント量の問題は，分析に限らず設計でも起こると考えられる。それはＯＭＴ法に従って分析と設計で同じ種類の仕様書を作成すると仮定すれば，設計でも分析と同程度の枚数が追加されることが予測されるからである。また仮に設計では実装言語を反映させる仕様書を使うとしても，枚数がさらに増すことはあっても減る可能性は少ないからである。

4 本報告の目的

このように，既存オブジェクト指向手法ＯＭＴ法は分析の表記と手順は充分に提供する[5]が，(基本／詳細)設計の表記や手順は不明である。またドキュメント量が多いという問題もある。そこで本報告では，Ｃ++やＧＵＩという実装環境を前提として，一貫した表記で充分に記述でき，分析から実装までの手順が明らかで，かつ妥当な作業量となるようなオブジェクト指向手法を確立することを目的とする。

5 手法改良の方針

我々はこの目的に対し，分析支援が充実しているＯＭＴ法(Fig 5.1を参照)をベースとして用い，この手法を改良することで問題を解決することにした。

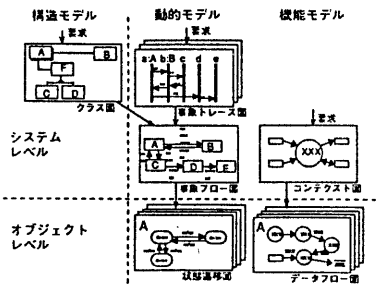

Fig 5.1 ＯＭＴ法の表記

手法改良の方針は以下に示すように決めた。つまり，

1)記述力を拡充する(表記第一版作成)
2)第一版適用により記述力を確認する
3)手法をリファインする(表記第二版作成)
4)第二版適用の効果を確認する

1)・3)では問題を絞って手法の改良案を考え，2)・4)では改良した手法を適用し，洗練するという方針である。ここで適用対象はFig 5.2に示すような画像ファイリング装置の業務アプリ(一部)である。第六章の６．１節から６．４節では，この方針に従って手法を改良した作業経過と結果を示す。

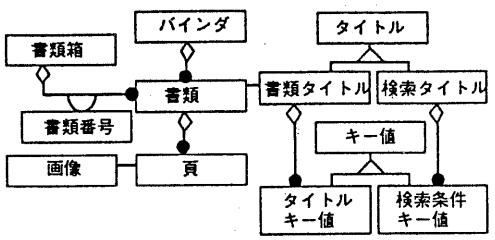

Fig 5.2 対象への手法の適用範囲

6 結果

6．１ 記述力を拡充する(表記第一版作成)

ここでは開発の前提，つまりＣ++／ＧＵＩを反映するための表記改良と，その他実装に必要な表記を追加するという改良について述べる。

●Ｃ++を反映する

Ｃ++との対応を改善するため，我々はＯＭＴ法の表記要素と実装の構成要素との差分を基本／詳細設計における表記要素として追加しようと考えた。そこでＯＭＴ法の表記をTable 6.1のようにモデル，仕様書，表記要素という三段階に分類した。つまり，

Table 6.1 ＯＭＴ法の表記(モデル，仕様書，表記要素の対応)

表記 / 範囲	構造モデル	動的モデル			機能モデル	
	クラス図	事象トレース図	事象フロー図	状態遷移図	コンテキスト図	データフロー図
システム単位	・クラス クラスの種類 ・関係 関係の種類 ・関係の特性 多重度 役割	(シナリオ) 文章 ・メッセージの送信順序 送受信の対象 制御	・メッセージの一覧 送受信の対象 制御		・外部仕様 操作装置 属性 他のクラス	
クラス単位				・実行順序 事象 動作 遷移条件		・外部仕様 操作装置 属性 他のクラス

モデルには構造／動的／機能モデルの三つがあり，各モデルは仕様書を用いて記述される。仕様書の記述範囲はシステム／クラス単位という二種類に分けられる。各仕様書の表記要素は仕様化項目，選択項目という二段階に分類できる。例えば，構造モデルの表記要素はクラス，関係，関係の特性という三つの仕様化項目に対応する。そしてクラスという仕様化項目はクラスの種類という選択項目を持ち，この選択項目に対して幾つかの表記要素が選択肢として提供される。これらの表記要素から一つを選ぶこと

で仕様の決定を表現できる。

次に我々は分析の表記に対して，(基本／詳細)設計で追加すべき仕様化項目，選択項目を検討した。ここではFig 5.2に示す範囲を対象にＧＵＩを用いずに小規模プロトタイピングを行い，ソースコードから逆に仕様化項目を抽出した。その結果，機能モデルでは関数の内部仕様という仕様化項目の追加が必要であった。一方，構造モデルや動的モデルでは選択項目や選択肢を増すという小さな追加で対応できる見通しを得た。そのため我々は実装言語と対応する構成要素を備えているＢｏｏｃｈ法[6]を参考に，各モデルへ必要な表記要素を追加することにした。

機能モデルの内部仕様という仕様化項目に対しては，Ｂｏｏｃｈ法にも正式な表記がなく，必要に応じて流れ図を用いることになっている。我々は，複雑な関数の実装時には内部仕様を記述すべきだと考え，流れ図を採用することにした(Table 6.2を参照)。

Table 6.2 機能モデルへの表記要素の追加

仕様書	仕様化項目	選択内容	表記要素	
			分析で用いる表記要素 (OMT法，Ｂｏｏｃｈ法に共通)	設計で追加する表記要素 (仕様化項目の選択から決定)
コンテキスト図／データフロー図	・データの入出力	属性 メンバ関数 データフロー 他のクラス	データストア プロセス メンバ関数の起 ターミネータ	
流れ図 (当社標準)	・関数の内部仕様	関数名 所属クラス 可視性 処理の種類		メンバ関数名 ユーザ定義型 公開，保護，私的 順次，分岐，繰り返し

構造モデルの関係の特性という仕様化項目に対しては，分析ではＯＭＴ法に示されている通りに役割，多重度を用い，設計でＢｏｏｃｈ法を参考に可視性，方向，持ち方という選択項目を追加した。同様にクラスの種類という選択項目に関しては，分析ではクラス，抽象クラス，静的クラスを用い，設計でフレンドクラス，テンプレートクラス，クラスユーティリティという選択肢を追加した。その他クラスの種類，関係の種類という選択項目に関しても，実装環境に合わせて追加する言語の構成要素を表すため，Ｂｏｏｃｈ法を参考に設計の選択肢を増やした(Table 6.3を参照)。

Table 6.3 構造モデルへの表記要素の追加

仕様書	仕様化項目	選択内容	表記要素		
			分析で用いる表記要素 (OMT法，Ｂｏｏｃｈ法に共通)	設計で追加する表記要素 (Ｂｏｏｃｈ法から追加)	使わない表記要素
クラス図	・クラス	クラスの種類	クラス，抽象クラス，静的クラス	フレンドクラス，テンプレート，メタクラスユーティリティ	メタクラス
	・関係	関係の種類 可視性	集約，(単一／多重)継承，関連	利用，テンプレート具象化 公開，保護，私的	メタ関連
	・関係の特性	多重度 制約 方向 持ち方	1，0/1，0～多，1～多，n，m～n ロール，限定子	双方向，片方向 実体，参照	

構造モデルと同様に構造モデルについても，設計の表記要素を追加した(Table 6.4を参照)。

Table 6.4 動的モデルへの表記要素の追加

仕様書	仕様化項目	選択内容	表記要素		
			分析で用いる表記要素 (OMT法，Ｂｏｏｃｈ法に共通)	設計で追加する表記要素 (Ｂｏｏｃｈ法から追加)	使わない表記要素
事象トレース図／事象フロー図	・送受信の対象	オブジェクト	オブジェクト		
	・制御	方向 制御の中心 メッセージの特性	片方向 順次的	リターン値の追加 実行中，休止中	メタ関連 同期，ポータ，タイムアウト，非同期
状態遷移図	・関数の実行順序	状態 遷移	状態 事象，動作，活動，条件		

●ＧＵＩを反映する

ＧＵＩの構築を支援するため，我々はモデルの階層化や画面仕様の記述を導入しようと考えた。

階層化については，ＯＭＴ法には複数のクラスをグループとしてまとめたり構造を階層化したりする手段がなかった。しかし我々は実際にＧＵＩを構築してみて，問題領域のクラス群とＧＵＩに関するクラス群とは独立性が高いことを実感し，仕様書を分割すべきだ考えた。そのため工程ごとに異なる抽象度で記述するという方策を立てた。つまり分析と同一表記で，基本設計ではシステム全体のサブシステム構成を示し，詳細設計ではサブシステムごとに内部のオブジェクト構成を記述する方法を採用した。

画面仕様については，ＯＭＴ法には画面のレイアウトや画面間の遷移を示す表記がなかった。しかしＧＵＩを構築するには画面仕様は不可欠であることを実感し，画面仕様に関する仕様化項目を明確にする必要があると考えた。そのため画面仕様を正式な仕様書として導入し，入出力データや受信可能なメッセージなどを仕様化するという方針を立てた。

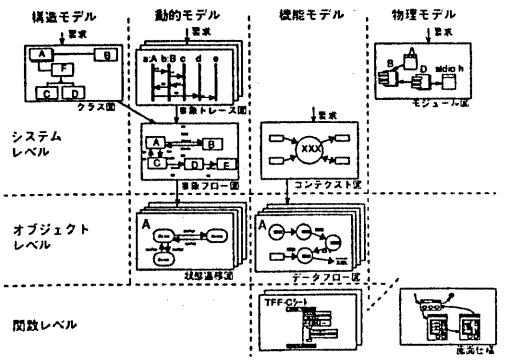

Fig 6.1 表記第一版

●その他

その他では，主に実装完了後にクラスをどのファイルへ割り付けたかという情報が必要であると考え

て，物理モデルを追加した。その際，Ｂｏｏｃｈ法のモジュール図という仕様書を参考に，物理構造の表記要素を導入した。

記述力拡充後の表記(表記第一版)をFig 6.1に示す。

６．２ 第一版適用により記述力を確認する

我々は記述力が拡充されたことを確認するため，対象に表記第一版を適用して仕様化しながらＧＵＩを用いるプロトタイプを作成した。そして構造／動的／機能モデルについて記述力を評価した。但し，ここでは機能モデルの説明は省略し，構造／動的モデルのみを示す。それは機能モデルが，個々のオブジェクトに属す各メンバ関数の内部仕様というシステム全体から見れば局所的な範囲の記述であり，開発上の重要性が低いからである。たとえ内部仕様を変更しても，各オブジェクトの外部仕様としてメンバ関数の入出力が確定されている以上，その影響が他の仕様書に及ぶことはほとんどない。

●構造モデルの記述力の確認

構造モデルではクラスの種類，関係の種類，関係の可視性，関係の方向，関係の持ち方という五つの選択項目に対して選択肢を増やした。このうちクラスの種類，関係の可視性，関係の持ち方の区別はFig 6.2 で確認できる。例えば，クラスの種類にフレンドクラスが追加されているが，Fig 6.2では「F」が付いた逆三角形でそれを示している。

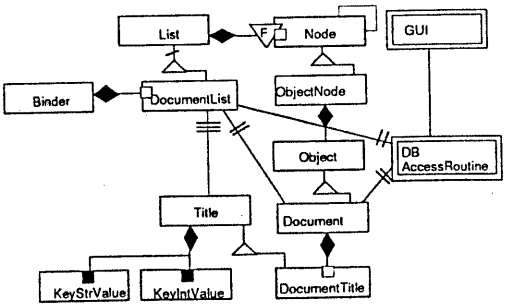

Fig 6.2 設計のクラス図(問題領域の部分)

●動的モデルの記述力の確認

動的モデルでは，リターン値の返却・オブジェクトの実行／停止という二つの選択項目について選択肢を増やした。これら二つをFig 6.3 において確認できる。例えば，リターン値の返却を表すために点線の矢印を導入したが，Fig 6.3の一番上(ProtoManagerからUserへの表示データを表す)の矢印がそれに対応する。

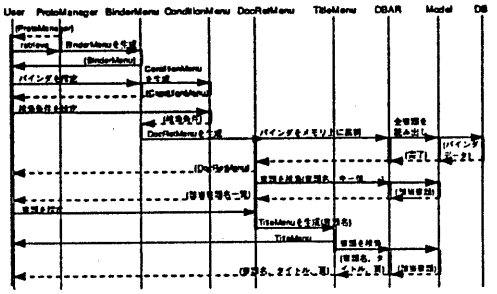

Fig 6.3 設計の事象トレース図(初期)

６．３ 手法をリファインする(表記第二版作成)

ここでは開発中に作成するドキュメント量を減らすため，表記の簡略化により仕様書を洗練し，手順の改良により一度に作成するドキュメント量を減らすという二つの改良を行った。

●表記の簡略化

表記の簡略化においては，仕様書の種類を減らすことを目的として，個々の仕様書に対して開発上の意味付けを明らかにし，同一情報を示す仕様書を共通化しようと考えた。そして我々は機能モデルのデータフロー図と動的モデルの事象フロー図に着目した。データフロー図は各クラスのメンバ関数一覧とその入出力データを記述する。事象フロー図はクラス間のメッセージ一覧を示す。そこで我々は以下二つの理由からデータフロー図を使わずに事象フロー図にまとめて記述する方策を立てた。理由の一つはデータの流れはメッセージの流れに追記できるからである。もう一つは事象フロー図が事象トレース図から機械的作業によって作成できるからである。

また各仕様書の開発上の位置付けを次のように明らかにした。クラス図，事象トレース図，事象フロー図はシステム全体をクラス群に分けるための仕様書で，各々クラス関の構造，振る舞い，機能(外部仕様)を示す。流れ図は各操作関数の内部を作成するための仕様書である。画面仕様はＧＵＩ作成に必要な情報を収集するための仕様書で，画面構成，レイアウト，状態などを示す。一方，状態遷移図は各クラスの利用方法を示す仕様書で，呼び出し可能な操作関数や実行順序を示す。モジュール図はシステム全体の物理構造を表す仕様書で各クラスの物理ファイルへの割り付けやコンパイル依存関係を示す。

●手順の改良

手順の改良においては，開発者が一度に記述する量を減らすことを目的として，仕様書を開発用／保守用に分け，保守用の仕様書は作成する時期を開発後まで遅らせるとともに，仕様書ごとにどの工程でどの範囲を記述するかを明らかにした。その結果，仕様書の役割は情報収集と情報伝達に大別できた。また状態遷移図とモジュール図は情報伝達の役割しか果たさないことが分かった。そこで状態遷移図とモジュール図は保守用の仕様書という位置付けとして，開発後に作成することにし，他の仕様書と作成時期をずらした(Fig 6.4，6.5参照)。

以上により，簡略化した表記を表記第二版としてFig 6.4，6.5に示す。

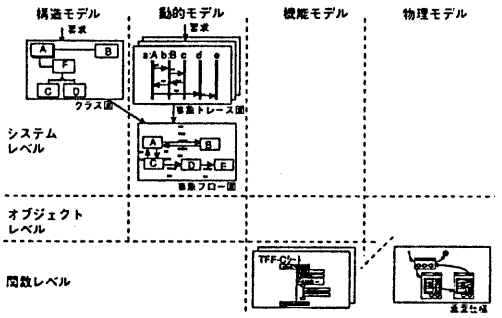

Fig 6.4 表記第二版(開発用)

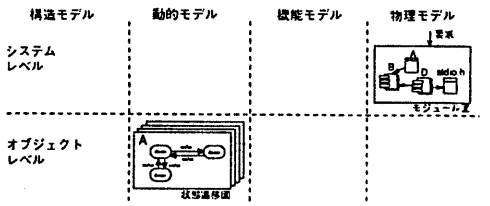

Fig 6.5 表記第二版(保守用)

各仕様書の作成時期と範囲については以下の通りである。分析ではシステムの問題領域クラス群を示すクラス図，事象トレース図，事象フロー図を作成し，基本設計ではシステム全体におけるサブシステム群を示すクラス図，事象トレース図，事象フロー図を作成し，詳細設計ではサブシステムごとのクラス群を示すクラス図，事象トレース図，事象フロー図を作成するとともに，個々のメンバ関数の内部仕様を表す流れ図を作成する。開発完了後には詳細設計で作成したクラス図とソースコードを参照しながら，クラスごとの状態遷移図，システム全体のモジュール図を作成する。

複数の担当者で開発を分担する時には，以下の運用方法で第二版を用いることができる。それはクラス図，事象トレース図，事象フロー図といったクラス間の役割分担を決定する仕様書は全員で作成してレビューを行い，その後に各担当者がクラス単位で分担し，流れ図を作成しながら内部を作るという運用方法である。この運用方法を採用すれば，開発に必要十分な仕様化が行えるだけでなく，開発者が一度に作成するドキュメント量を減らすことができる。

6．4 第二版適用により効果を確認する

我々は，ドキュメント量が削減されることを確認するため，対象にＤＦ表記第一版と第二版の適用により，どの程度ドキュメント量に違いがあるかを調べた。結果はFig 6.6 に示す通りで，表記第二版で開発すると，ドキュメント量はベースとなるＯＭＴ法を分析・設計で通して使う場合の半分程度になることが明らかになった。

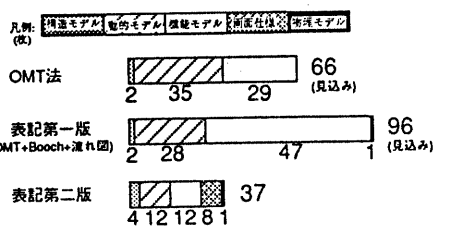

Fig 6.6 ドキュメント量の比較

また，C++で開発するために必要なシステムのオブジェクト構造，オブジェクト間の振る舞い，個々のオブジェクトの外部仕様，画面仕様に関する情報をもれなく記述することができた。

7．まとめ

我々はオブジェクト指向手法の表記改良を提案した。具体的には，ＯＭＴ法をベースに，C++やＧＵＩという実装環境を前提とした実装への移行性を高めるため，記述力を拡充した。また同一情報を含む仕様書を共通化することで仕様書の種類を減らし，仕様書の用途に応じて作成時期を遅らせることで一度に作成する仕様書を減らした。

また以下のように各開発工程の位置付けを明らか

にした。分析ではシステム内に関して問題領域の切り分けを行い，基本設計では周辺機器を考慮してサブシステムレベルの分割を行い，設計では個々のサブシステムをクラスに切り分ける。開発完了後には保守に備えて利用上必要な情報をまとめる。

改良した手法によって，既存のオブジェクト指向手法と比べて約半分のドキュメント量で，開発に必要な情報をもれなく記述することができた。

参考文献

[1]飯島 正，中野 裕子，吉田 和樹，本位田 真一:
"画像ファイリングシステム開発へのオブジェクト指向開発方法論の適用事例"
情報処理学会 ソフトウェア工学研究会，Vol. 85，
pp 15-22，1992.5

[2]山城 明宏，中野 裕子，本位田 真一:
"ＯＯＡとリアルタイムＳＡの変更容易性に対する考察"
電子情報通信学会 信学技報，SS93-22，pp 17-24，
1993.9

[3]吉田 和樹，山城 明宏，齋藤 悦生:
"画像ファイリングシステムへのＯＭＴの適用"
1993年情報学シンポジウム予稿集，日本学術会議 他，
1993

4]中里 竜，吉田 和樹，入内島 裕子，山城 明宏:
"ＯＭＴ法における画像ファイリングシステムの分析
-拡張容易性と分析指針-"
電子情報通信学会 信学技報，SS93-22，pp 17-24，
1993.9

[5]J.Rumbaugh，M.Blaha，W.Premerlani，F.Eddy，
and W.Lorensen:
"オブジェクト指向方法論 ＯＭＴ -モデル化と設計-"
トッパン，1992，羽生田 栄一監訳

[6]G.Booch:
"Object Oriented Analysis and Design with Applications
2nd ed."，The Benjamin/Cummings Publishing Co.，
1993