ソフトウェア工学　84－3
（１９９２．３．３）

画像検索システムへのＯＭＴ手法の適用

山城明宏*　手島文彰*　井上勝博*　前田伊知朗**
*　(株) 東芝　システム・ソフトウェア技術研究所
**　(株) 東芝　医用機器事業部

オブジェクト指向分析／設計方法論の１つであるＯＭＴ手法を、医用画像処理システムのサブシステムである画像検索装置の分析と設計に適用した経験をもとに、オブジェクト指向パラダイムの有効性について検討した結果を述べる。導入容易性、記述性、支援ツールの３つの観点からＯＭＴ手法の利点／欠点の検討を行った結果、本手法は特に分析フェーズの方法論として完成度が高いものであること、および、その有効性は記述の目的によって異なることを述べる。また、利点と欠点を明確にすることによって、本手法を支援する環境の提案も行う。

The Application of OMT to the Medical Image Retrieval System and its Evaluation

Akihiro Yamashiro*　Fumiaki Teshima*　Katsuhiro Inoue*　Ichiro Maeda**

*　Systems & Software Engineering Laboratory, Toshiba Corporation
**　Medical Systems Division , Toshiba Corporation

*　70, Yanagi-cho, Saiwai-ku, Kawasaki-shi 210, JAPAN
**　Toshiba Nasu Works, 1385, Shimoishigami, Otawara-shi, Tochigi 329-26, JAPAN

This paper describes the result of discussion on the effectiveness of the Object-Oriented Analysis and Design methodology, which is one of the Object-Oriented paradigms. Especially, we considered the application of the OMT to the medical image retrieval system. The pros and cons of the OMT methodology are discussed from three points of view, namely, ease of introduction, descriptiveness, and the necessity of the supporting environments. Concluding from our consideration, it clarified that the methodology is well-organized and practical enough to be applied to the development of commercial products and that the effectiveness will differ based on the purpose of the application. Furthermore, in this paper, the supporting environment for the methodology is discussed, based on our experience.

1．はじめに

プログラミングフェーズへのオブジェクト指向言語の適用に続き、その上流の分析／設計のフェーズにもオブジェクト指向のパラダイムを適用するための手法が提案されつつある[Booch90] [Coad91a] [Shlaer88]。これらの背景として、従来の手法ではフェーズ間にギャップが存在し[Pressman 87]、その改善を図る手法が待たれていたことが挙げられる。オブジェクト指向パラダイムはこの問題に対して分析／設計／実装においてモデリング観点を統一し、モデルの拡張／進化を支援することで解決を図るアプローチであるといわれている。

本報告では、ＯＭＴ(Object Modeling Technique)手法[Rumbaugh91]を医用画像処理システムの分析と設計に適用した経験をもとに、オブジェクト指向パラダイムの有効性について検討する。

本稿の構成は以下の通りである。２章で適用技術であるＯＭＴを紹介し、３章においてその適用対象である医用画像検索システムの概要を述べる。４章、５章は適用の経緯とその結果である。６章では、ＯＭＴ手法の利点／欠点について考察する。

2．OMT

オブジェクト指向分析／設計方法論の１つであるＯＭＴは、開発対象のシステムに３つの観点からのモデリングを行なうことで、ユーザの要求より得られる仕様の分析から、実装に結びつく設計仕様の作成までを支援することを特徴とする。

モデリングにはオブジェクトモデリング、ダイナミックモデリング、ファンクショナルモデリングの３つの観点がある。オブジェクトモデルは、オブジェクトとクラスの静的なデータ構造及びそれらお互いのリレーションをERモデル[Chen76]の拡張によって記述する。ダイナミックモデルは、システムにおけるオブジェクトがイベントやオブジェクト同士のメッセージのやりとりに対してどう反応するかをステートチャート[Harel88]で記述する。ファンクショナルモデルは、オブジェクト内部に保持される値の変化とその変化の際の制約をデータフロー図[Hatley87]で記述する。

方法論の適用工程は、図１に示す通り、分析、システム設計、オブジェクト設計の３フェーズと、その後の実装に分けられる。分析のフェーズでは、上記３つのモデリングを用いて要求の獲得とリファインを続け、「システムが何をするか」を表現するための実世界指向モデルを開発する。システム設計フェーズは、数種類の標準アーキテクチャをもとに上位レベルのシステムの構造を選択する。オブジェクト設計フェーズでは、分析フェーズで開発されたモデルを詳細化し、実装の詳細な基盤を提供する計算機指向モデルを開発する。

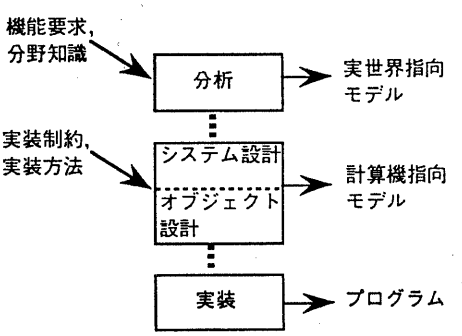

図１　ＯＭＴによるソフトウェア開発プロセス

3．対象

ＯＭＴ手法の適用対象は、医用画像処理システムの中のディレクトリマネージャと呼ばれる画像検索サブシステムの開発である。本サブシステムは、(1) 異なるアプリケーション間で共通に利用し、アプリケーション共通の画像選択操作を提供する、(2) 画像管理装置／ファイルシステムの構造の違いを隠蔽し、アプリケーションに操作の違いを意識させない、(3) ユーザインタフェースの標準化／向上を図ることの３点を目的とする。

ディレクトリマネージャに相当する機能は既にに存在しており、今回の開発は、機能の改良とプラットフォームの変更に伴うサブシステムの作り直しである。実装はGUI環境としてMotifが動作するUNIXマシン上で行なわれ、C言語の利用を前提としている。予想される開発の規模は15Kステップ程度である。

図２にディレクトリマネージャの動作環境を示す。ディレクトリマネージャは最初に、ファイル管理システムより得られるディレクトリ情報をもとに、仮想ファイルシステムを構築する。この情報は管理画像の変更に伴い、常時更新されなくてはならない。ユーザまたは何らかのプロセスがアプリケーションを起動すると、アプリケーションは必要に応じてディレクトリマネージャに画像選択要求を発行するので、ディレクトリマネージャはユーザインタフェースを通してユーザから画像選択に必要な情報を受取り、画像検索を開始する。検索された画像は起動元アプリケーションに渡される。

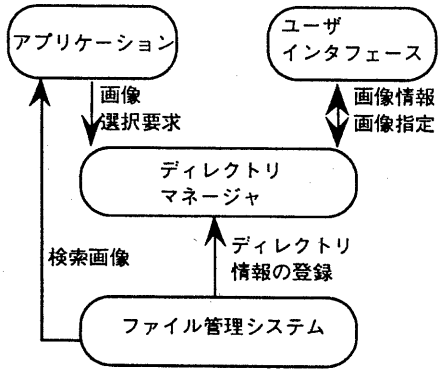

図2　ディレクトリマネージャの動作環境

4．OMTの適用

医用画像検索システムに対するOMTの適用を述べる。OMTでは、各フェーズの必要な作業項目が順序付けられている。我々の試行は基本的にこの手順に従うが、前提条件の異なる部分、及び不自然と思われる部分は独自の作業項目に変更を行なった。また各作業項目の実施において必要と考えられるドキュメント類は我々の都合に合わせて体系化／関連付けをし直した。

作業は4人で行なった。問題領域の知識を持つものは2人、[Booch90][Coad91a][Shlaer88]の他のオブジェクト指向分析／設計について見識のある者が2人である。4人ともHatleyのリアルタイムSA [Hatley87]に関しては適用経験を持つが、OMTについては試行前に知る者がなく、従って作業は原書の読み合わせから始めた。

4．1　分析フェーズ

●問題の獲得：
要求として獲得した情報は、システムに必要な機能をまとめた「要求仕様」、主要な対話シーケンスをまとめた「シナリオ」、実装環境における制約をまとめた「システム実現制約」、旧システムの「データフォーマット仕様」である。前2者が本来の要求、後2者が実装に関連する制約である。実装に関する制約やデータフォーマットは、今回の開発が旧システムの再開発であることと開発対象がシステム全体の中の1サブシステムであることから要求時点で課された制約である。

●オブジェクトモデルの構築：
要求仕様から約50名詞を抽出した。このなかで30名詞がリファインの対象となり、アソシエーション、クラス階層を考慮した結果、お互い関連付けられた約20のオブジェクト群が得られた。得られたオブジェクトは、(1)ディレクトリマネージャ及びそれと直接通信を行うアプリケーション、UIマネージャ、ストレージマネージャ、(2)オペレータ、(3)ディレクトリマネージャが管理する情報、(4)UIマネージャが管理するウィンドゥ群、(5)ストレージマネージャが管理するストレージ群、に分類される。

●ダイナミックモデルの構築：
シナリオから9種類のイベントトレース図を得た。イベントトレース図において通信を行うオブジェクトは、オブジェクトモデルから得られたディレクトリマネージャ、アプリケーション、UIマネージャ、ストレージマネージャ、オペレータである。オペレータは直接ディレクトリマネージャとは通信しないが、イベントトレースを理解しやすいものとするために必要であった。次に9種類のイベントトレース図を1枚のイベントフロー図にまとめ、個々のオブジェクトが送受信するイベントに着目したオブジェクト毎の状態図を作成した。

●ファンクショナルモデルの構築：
ディレクトリマネージャとUIマネージャを対象にデータフロー図を作成した。データフロー図は情報の流れそのものよりも、状態図で記述したイベントフローとそのイベントの受信／発信時に得ていなくてはいけない情報はなにか、というダイナミックモデルとの関わりを記述するために用いた。

4．2　システム設計フェーズ

●構造に関する上位レベル設計戦略の作成：
分析の結果得られた主要オブジェクト群をサブシステムに分割し、その依存関係をブロック図に明記した。ブロック図では、ディレクトリマネージャはUIマネージャが提供する機能に依存しており、これら両者は、依存度の低いストレージマネージャとともに、アプリケーションを実現するための機能を提供すること、及びUIマネージャはMotif上に、ストレージマネージャはOSの機能上に構築される、という構造を明らかにした。

●その他の上位レベル設計戦略の作成：
本フェーズにおいてブロック図以外に明記した設計戦略は、(1)サブシステムの並列性の識別とタスク割り付け方針、(2)サブシステムの境界条件（初期化時、終了時、異常時）、(3)データストアの実装と共有資源のアクセス制御の方針、(4)制御の実装方針の決定、(5)設計上のトレードオフ項目の優先度の決定、である。この中で(1), (3), (4)がアーキテクチャ構築のベースとなった。

●アーキテクチャ図の構築：

得られた上位レベルの設計戦略をもとにアーキテクチャ図を構築した。これは実装環境と実現可能性を考慮して、分析フェーズで得られた主要オブジェクト群の構造と互いの関係を記述し直したものである。今回のアプリケーションはデータ管理とユーザ／プロセス会話処理を含むので、対話インタフェースとトランザクションマネージャの標準アーキテクチャを参考にした。

実際には、設計戦略の多くは環境との兼ね合いで要求の段階で明らかであった。設計戦略の文書化はオブジェクト設計の前提条件を明文化する目的で行った。

4．3　オブジェクト設計フェーズ

正規のオブジェクト設計は、(1) 3モデルの統合によるオペレーションの抽出、(2) オペレーションを実現するアルゴリズムの設計、(3) データアクセスパスの最適化、(4) 制御の実装、(5) クラス構造の調節による継承の増加、(6) アソシエーションの実装方法の決定、(7) 実装を考慮したクラス属性の表現方法の決定、(8) クラス群のパッケージング、の8作業項目からなる。我々のアプリケーションはC言語による実装を想定しているので(5)はスキップした。また、実装上の制約として満たすべきデータ構造とそのアクセスルーチンの設計以前に(2), (7), (8)を行なうことは不自然と考え、これらもスキップした。このフェーズでの実作業は(1), (3), (4), (6)である。

●Booch図の作成：
オブジェクト群に対して、オブジェクト図の関連、状態図のイベント送受信、データフロー図の機能をオペレーションとして追加し、オペレーション間の方向付けられたアクセスパスを記入したダイアグラム(Booch図 [Booch87])を作成した。ここでは分析モデルにおいてUIマネージャとの通信として仕様化されていたディレクトリマネージャ及びオペレータとの関係を、UIマネージャが管理するウィンドゥオブジェクトとの関係として表現することで、実装ベースのモデルに構築し直した。

●詳細オブジェクト図の作成：
画像データ構造を表現するオブジェクト図を構築した。

5．結果

分析／設計に要した作業期間は全員による集中作業2週間を含む1．5ケ月である。作業時間の割合は、分析3：システム設計1：オブジェクト設計2程度である。分析／設計の結果として図3に示す17種類のドキュメントを得た。各工程で記述するドキュメントは図4に示す依存関係を持つ。矢印は記述の順序を示すとともに、結ばれたドキュメント間では一貫性が保持されるべきことを示す。矢印の方向は基本的な作業手順であり、実際の工程で発生した後戻りは書かれていない。

工程	ドキュメント分類	ドキュメント（概要）	種類	個々の対象
全	データ辞書	データ辞書	1	全体
分析	問題の記述	機能要求仕様（システムの機能）	1	全体
		シナリオ（システムに対する典型的な操作）	9	典型的操作
		システム実現制約（ＨＷ等の環境の制約）	1	全体
		データフォーマット仕様（旧データ構造）	1	全体
	オブジェクトモデル	抽出名詞一覧（機能要求仕様の中の名詞）	1	全体
		オブジェクト図（オブジェクトとその相互関連）	1	全体
	ダイナミックモデル	イベントトレース図（イベントの流れの順序）	9	典型的操作
		イベントフロー図（オブジェクト間のイベント）	1	全体
		状態図（オブジェクト毎の状態遷移）	5	オブジェクト
	ファンクショナルモデル	データフロー図（コンテクスト）	1	全体
		（オブジェクト毎の機能）	2	オブジェクト
システム設計	上位レベル設計戦略	ブロック図（システムの環境／構造）	1	全体
		その他（並列性、設計トレードオフ、データ構造等に関する設計戦略）	4	観点毎
	アーキテクチャ	アーキテクチャ図（システムの環境／構造）	1	全体
設計	データ構造	詳細オブジェクト図（画像データの構造）	1	画像データ
	システム構造	Ｂｏｏｃｈ図（オブジェクトとその相互関連）	1	全体

図3　ドキュメント一覧

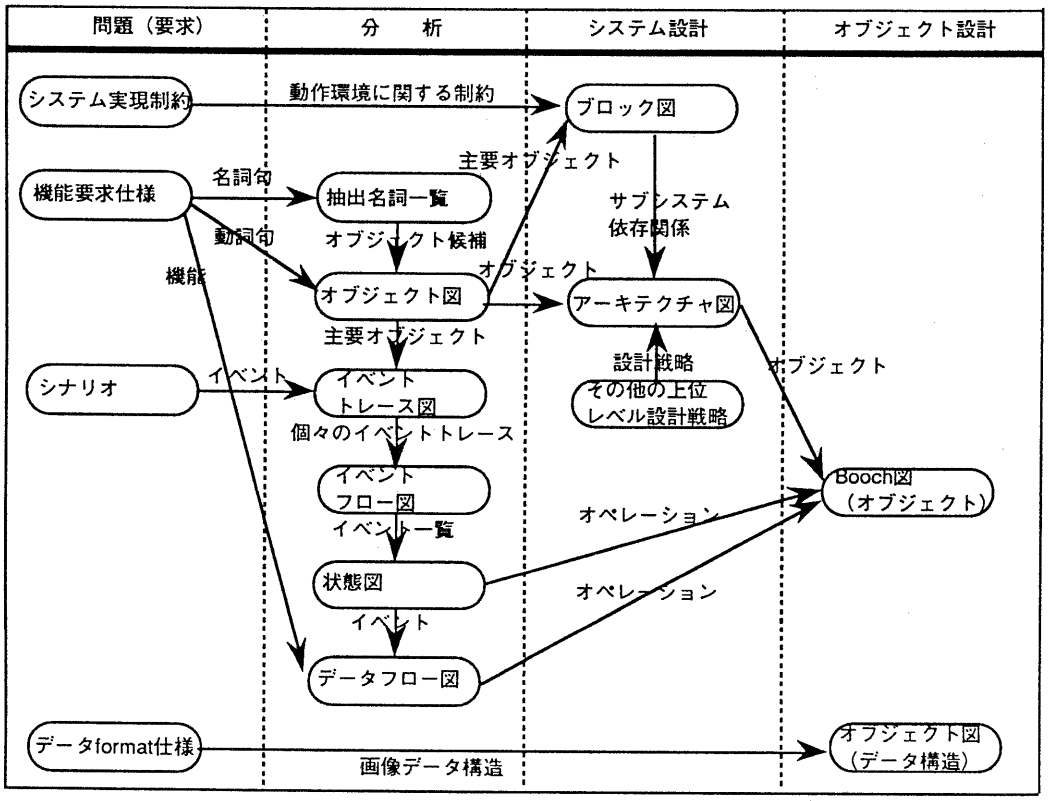

図4　ドキュメントの関連

6．OMTの評価

本章では、以下の3つの観点からOMTの利点／欠点の考察を行ない、これらを通してオブジェクト指向パラダイムの分析／設計工程適用の効果をまとめる。

[導入容易性]
(1) 方法論に示される作業は本当に実施可能か？
(2) 既存の開発スタイルとの親和性は良いか？
(3) 大規模システムの開発にも使えそうか？

[記述性]
(4) 分析／設計における思考の流れはスムーズか？
(5) OMTモデルは作成／理解しやすいか？
(6) 何が表現しやすくて、何が表現しにくいのか？

[支援ツール]
(7) どのような支援機能が必要か？
(8) どのようなツールによる支援が可能か？

6．1　導入容易性の観点からの評価

(1) 方法論に示される作業は本当に実施可能か？

Hatleyのリアルタイム SA手法などに比べると、工程毎の作業手順と各作業で作成すべき成果物が十分明確になっており、実際の製品開発にも取り入れた方が良さそうなノウハウが豊富に盛り込まれている。特に、分析フェーズの作業手順が良く考えられており、具体例をもとに要求を抽象化／一般化していくことで、抜けの少ない分析結果が得られる感触を得た。またOMTモデルをプログラムに展開していくまでのガイドは、非常に参考になった。この意味で、方法論の完成度は高いといえる。あえて問題点を指摘すれば、オブジェクト設計フェーズで、分析フェーズで得られたオブジェクト群に対して実装を考慮したトップダウンな詳細化を行う作業手順を推奨している点である。実際の設計ではソフトウェアアーキテクチャに対してボトムアップで設計する部分がかなりの比重を占める。

(2)既存の開発スタイルとの親和性は良いか？

(2.1) ＯＭＴモデルと既存成果物との対応関係：

これまで提案された分析／設計手法の多くがそうであるように、ＯＭＴ手法も既存の成果物が持つ情報を全て表現できるわけではない。これは、情報を形式的に表現する場合の宿命である。情報の漏れのない記述を目的に、当社ではソフトウェア品質保証体系であるＳＱＡＰ(Software Quality Assurance Program)を用いてソフトウェア開発の標準工程、成果物を定めている。我々はＳＱＡＰに既存の分析／設計手法の仕様書を取り込むための基本的な検討を行っており、Hatleyのリアルタイム SA手法の成果物の対応付けは既に終えた。ＯＭＴ手法は、仕様書の種類はリアルタイムSA手法とさほど変わらないので、既存成果物との対応付けは可能であり、ＳＱＡＰに取り込めると考えられる。

(2.2) 開発手順の対応関係：

ＯＭＴで定めている開発フェーズは、ＳＱＡＰの中で工程と作業手順を規定しているＱＣＰ（Quality Control Process Chart）に出てくる工程とうまく対応しており、特に問題はない。但し、個々の工程の作業手順および作業内容はかなり食い違っている。ＯＭＴの場合、基本的には分析フェーズで獲得したオブジェクトを拡張／進化させていくことによって、最終的なプログラムを作成するのであるが、その過程でオブジェクトは１つのタスクであったり、１つのパッケージ（データ抽象化設計に基づくデータとオペレーション群）であったりする。そして、それらのオブジェクトは、タスクとなるべきものであったり、タスクを構成するパッケージとなるものの部分／全体、スーパークラス／サブクラスであったりして、非常に混沌とした世界を作り出している。「もの」を中心に分析／設計を行うことで、分析フェーズと設計フェーズとのギャップを少なくするという利点は、オブジェクトの意味が二転三転するという欠点と裏表の関係にある。

(3) 大規模システムの開発にも使えそうか？

分析フェーズや設計フェーズにおけるオブジェクトの抽出は恣意的である。そして、大量の問題記述の中から効果的なオブジェクトを見つけることは慣れぬ技術者にとって至難の技である。以降の分析／設計作業はオブジェクトを中心に行われるため、選択したオブジェクトがあまり良くなければ、オブジェクトの取捨選択からＯＭＴモデルの構築をやり直さなければならない。このため、途中で排除したオブジェクト候補は、理由と共に残しておく必要がある。以上より、ＯＭＴの適用は試行錯誤的であり、大規模システム開発への非常に大きなリスクを伴うといえる。但し、オブジェクト抽出作業に優秀な技術者を投入することで以下の作業をスムーズに進めることができると考えるならば、良いシステムを作るための作業の負荷がオブジェクト抽出の一点に集中することは利点であるとも言える。

６．２　記述性の観点からの評価

(4) 分析／設計における思考の流れはスムーズか？

以前から指摘されているように、要求分析フェーズと設計フェーズとの間に大きなギャップが存在している。このギャップとは、例えば構造化分析手法に従ってデータフロー図を作成しても、それからシステムアーキテクチャやソフトウェアアーキテクチャが出てくるわけではない、ということを指している。確かに、出来上がったアーキテクチャとデータフロー図との対応は付けられるが、アーキテクチャの設計は技術者の経験に大きく依存することが多い。

オブジェクト指向分析／設計手法は、分析から設計までオブジェクトモデルという統一した視点を提供することで、このようなギャップを少なくするといわれている。今回、その有効性を確認するために、分析フェーズから設計フェーズまでのオブジェクトの追跡性に着目して評価を行う。表５に、その対応表を示す。

オブジェクト	要　求	分　析	設　計
システムマネージャ	×	×	○
アプリケーション	○	○	○
オペレータ	○	○	○
マネージャ	×	○	×
ＵＩマネージャ	○	○	○
ウィンドゥ	×	○	×
メニュー	○	○	○
ソートアイテム	○	○	×
サーチアイテム	○	○	×
一覧表	○	○	○
ＯＫボタン	○	○	×
確認ウィンドゥ	○	○	○
ヘルプウィンドゥ	○	○	○
ソートウィンドゥ	○	○	○
サーチウィンドゥ	○	○	○
イメージウィンドゥ	×	×	○
ディレクトリマネージャ	○	○	○
ディレクトリ情報	○	○	×
ディスプレイ	×	○	×
ストレージマネージャ	○	○	○
ファイルディレクトリ情報	○	○	×
（その他２５アイテム）	○	×	×

表５　オブジェクトの追跡性

表5を見ると分かるように、アーキテクチャを決定したことにより新たに付け加えられたオブジェクトは少なく（2オブジェクト）、また減ったもの（8オブジェクト）は他のオブジェクトに併合されたことを示しており、工程間のギャップを埋めている。これは、オブジェクトという概念がもともと並列性をもっていることと無関係ではない。また、今回の適用がシステム全体の中の1つのプロセスに過ぎず、全体のアーキテクチャは決まっていたことも影響している。

(5) OMTモデルは作成／理解しやすいか？

OMTでは、異なるモデル間の関係がダイアグラム上に明示的に現れてこない。例えば、Hatleyのリアルタイム SA手法では、制御フロー図上の「コントロールバー」記号が制御フロー図と制御仕様書（状態遷移図）を結び付ける役割を果たし、データフロー図上の「データストア」記号がデータフロー図とデータ構造図（ER図）とを結び付けている。これに対して、OMTではラベル（オブジェクトやアソシエーションの名前）でしか異なるダイアグラム間の対応関係を記述することができない。つまり、Hatleyの手法では、全ての分析／設計モデルを比較的強く結合しているのに対して、OMTにおけるモデルは弱い結合となっているのである。このことは、あるOMTモデルを構築するときに、他のモデルとの関連をあまり意識することなく、比較的自由に記述することを可能にする。逆に、この利点はモデル間の追跡を面倒にするだけでなく、モデル間の矛盾が生じ易くなるという欠点でもある。また、モデルの保守という観点からもあまり薦められない。

(6) 何が表現しやすくて、何が表現しにくいのか？

今回の適用対象では、GUIとしてMotifを採用しているわけであるが、Motifと画像検索システムとの関係が、OMTのアーキテクチャモデルではうまく表現できなかった。また、タスク間の通信方式や同期関係なども実際のソフトウェア開発では重要な情報であるが、それらを記述できないという問題もある。更に、今回はBooch図を導入することで解決を図ったが、モジュールの呼び出し関係なども直接的に表せない。

これに対して、今回の適用対象の最大の技術的課題であった、複数のファイルシステム間で異なるデータ構造を吸収するの仕組みを検討するために、オブジェクトモデルを使った際、3種類のアソシエーションによって、何が同じで何が違うのかをきれいに整理することができた。従って、OMTモデルは問題の整理やデータ構造の表現には非常に適しているといえる。

6．3　支援ツールの観点からの評価

(7) どのような支援機能が必要か？

(7.1) 作業を効率化するための機能：

●ダイアグラムの知的編集機能
成果物の評価で述べたように、オブジェクト指向分析の過程で得られるモデルは分析の進行に伴い、詳細化と抽象概念の導入が行われる。オブジェクトモデルを例にすると、例えば最初はアソシエーションとして記述されたオブジェクト間の関係を、分析の進行に伴いアグリゲーションとして記述し直したり、あるいは1つのオブジェクトを異なる複数のオブジェクトに具象化すること（あるいはその逆）に伴い、アソシエーションの張り替えを行うことが頻繁に発生する。このようなモデルの構成要素、あるいはモデルそのものの「進化」はオブジェクト指向パラダイムの特徴の1つである。このため、ダイアグラムを編集するエディタは、ダイアグラムを構成する図形要素の編集に加えて、変更に伴うリンクの張り替え（変更の派生）がモデルの意味を考慮して自動的に行えることが望ましい。

(7.2) 成果物などを管理するための機能：

●用語の管理機能
OMTでは、分析の開始時に自然言語で書かれた仕様書から名詞、動詞を抽出する作業から開始するが、これは(3)で述べた通り、仕様書が詳細であるほど困難な作業である。また、分析／設計の過程で多くのドキュメントを記述するため、仕様書間の用語の整合とその意味の統一、関連の保持をしておかないと、作業の経過に従って作業者間での用語の統一が困難となる。例えば、字づらは同じだが、性質の異なるオブジェクトの存在は、モデルの解釈を曖昧にし作業者の誤解を招く可能性がある。従って、分析／設計の作業を進める上では、用語の抽出／管理支援が必要となる。

●ダイアグラムの管理機能
設計モデルは、分析モデルの詳細化によって得られるため、分析モデルへの上書きが繰り返されて、分析モデルの原形がドキュメントとして残らない。このため、版管理を含めたダイアグラムの管理機能が必要となる。

●関連管理機能
フェーズ内、フェーズ間を問わず、多くのドキュメントは用語をキーとして互いに関連し合っており、それぞれ矛盾しないことが必要である。そのためには用語の管理とともに、ある語の定義が変更になった場合にはその語を用いる全てのドキュメントにおいて語の変

更が反映されなくてはならない。このために、仕様書に現われる用語をキーとして、同一フェーズにおける種類の異なる仕様書間の関係、同一モデルを記述したフェーズの異なる仕様書間の関係、の双方を辿れるようなリンクトデータベースのようなアプローチ（仕様書自身の構造／関連関係を保持し、仕様書間を任意に行き来できる環境の構築）が有効であろう。このとき、なぜ詳細化／抽象化を行なったかの理由を書いておくために、リンクには注釈付けできることが望ましい。

（8）どのようなツールによる支援が可能か？

●ＪＯＫＥＲ９１

現在、我々はソフトウェアの上流工程支援を目的として、JOKER91という用語知識の再利用ツールを開発している。このツールは自然言語の要求文の各々から名詞、動詞とそれらの修飾語、及び文形を切り出してそれらの関係を辞書に蓄積するとともに、名詞、動詞のいずれかをキーにしてブラウザに表示することを特徴としている。蓄積においては名詞群はis-a, has-aのリンクで結ばれる意味ネット形式で保持される。JOKER91では、文の中の特定の用語の意味的な類似性を文中の他の用語の類似度とその文型シグナチャの比較によって判断するアプローチを採用している。このようなツールを仕様書間の用語の意味及び関連を保持する用語辞書として利用することによって、矛盾の除去、参照の容易化といった改善を図ることが可能と思われる。JOKER91の例は文献[井上92]を参照。

●ＯＯＡＴｏｏｌ／ＯＭＴｏｏｌ

今回、OOATool[Coad91b]の試用も行ったが、本ツールがオブジェクトモデル作成に課すシンタクスと記述結果に対する文法チェックは不十分であった。OMToolは入手できず、試用していない。

結論としては、形式的な記述に何を求めるかによって、ＯＭＴの有効性は変わってくるだろう、という点が挙げられる。少なくともモデルの記述法の統一によって形式上のギャップは埋まったが、分析と設計との本質的なギャップは、それほど埋まっていない。しかしながら、オブジェクト指向アプローチは、再利用し易いソフトウェア部品を提供し、保守性や理解性を向上させるといった副次的な効果を考慮にいれるならば、従来の方法や構造化分析／設計手法に比べて、より良い手法であるといえる。

７．おわりに

本報告ではオブジェクト指向分析／設計方法論の１つであるＯＭＴ手法を、医用画像処理システムの分析と設計に適用した経験をもとに、オブジェクト指向パラダイムの有効性について検討した結果を述べた。導入容易性、記述性、支援ツールの３つの観点からＯＭＴ手法の利点／欠点の検討を行った結果、本手法は特に分析フェーズの方法論として完成度が高いものであること、および、その有効性は記述の目的によって異なることを述べた。また、利点と欠点を明確にすることによって、本手法を支援する環境の提案も行った。

本システムは、現在オブジェクト設計フェーズに続くプログラム設計を終了し、C言語による実装を行なっている。実装の終了後、設計と実装の間の追跡性とギャップについても調査することで、オブジェクト指向パラダイムの効果の検証については実装工程を含めて検討を続けていく予定である。

参考文献

[Booch 87] Grady Booch. "Software Engineering with Ada Second ed.", Benjamin/Cummings, 1987

[Booch 90] Grady Booch. "Object-Oriented Design with Applications", Benjamin/Cummings, 1990

[Chen 76] P.P.S.Chen. "The Entity-Relationship Model - toward a unified View of Data", ACM Transactions on Database Systems 1, 1, 1976

[Coad 91a] Peter Coad, Edward Yourdon. "Object-Oriented Analysis Second ed.", Prentice Hall International, 1991

[Coad 91b] Peter Coad, Jeff McKenna. "OOATool An Object-Oriented Analysis Tool for the Macintosh V 1.1 First ed." Object International, Inc., 1991

[Harel 88] David Harel. "On Visual Formalisms", Communications of the ACM 31,5, 1988, pp.514-530

[Hatley 87] Derek J. Hatley, Imtiaz A. Pirbhai. "Strategies for Real-Time System Specification", Dorset House Publishing, 1987

[Pressman 87] R.S.Pressman, "Software Engineering : A Practitioner's Approach Second ed.", McGraw-Hill Book Company, 1987

[Rumbaugh 91] J.Rumbaugh, M.Blaha, W.Premerlani, F.Eddy, and W.Lorensen. "Object Oriented Modeling and Design", Prentice Hall, 1991

[Shlaer 88] Sally Shlaer, Stephen Mellor. 本位田真一, 山口亨訳, "上流ＣＡＳＥのためのモデル化手法", 啓学出版, 1990

[井上 92] 井上秀行, 山田淳, 山城明宏, 井上勝博. "要求分析支援ツールJOKER'91", 第４４回情報処理学会全国大会論文集, 1992