ソフトウェア工学　89－6
（１９９２．１２．１４）

ＯＯＡとリアルタイムＳＡの変更容易性に対する考察

山城明宏　中野裕子　本位田真一
（株）東芝　システム・ソフトウェア生産技術研究所

オブジェクト指向分析手法の１つであるRumbaughのＯＭＴと構造化分析手法の１つであるHatleyのリアルタイムＳＡを、画像ファイリングシステムの検索機能の分析に適用した経験をもとに、オブジェクト指向パラダイムと構造化パラダイムの違い、及びその違いが分析作業の容易性と分析結果の理解し易さに与える影響について検討した結果を述べる。まず最初に、それぞれの手法の共通点と相違点を述べ、次に分析／設計の役割分担、手順の実施容易性、表現の一意性、トレーサビリティの４つの観点から両手法の利点／欠点の検討を行った結果より、構造化分析手法に対するオブジェクト指向分析手法の優位性を示す。

Consideration on ease of modification by comparing OOA with Real-Time SA

Akihiro Yamashiro　Hiroko Nakano　Shinichi Honiden

Systems & Software Engineering Laboratory, Toshiba Corporation

70, Yanagi-cho, Saiwai-ku, Kawasaki-shi 210, JAPAN

This paper describes the result of discussion on the difference between Object-Oriented paradigm and Structured paradigm from our experience of applying Rumbaugh's OMT and Hatley's Real-Time SA to the analysis of image filing system. We firstly discuss the similarities and differences of these methodologies. Then we discuss advantages and disadvantages of both methodologies from four points of view, namely separation of analysis and design, ease of following process, consistency of description, and the tracability of documents. Concluding from our consideration, it clarified that the Object-Oriented Analysis methodology is superior to Structured Analysis methodology concerning ease of modification.

1．はじめに

近年注目されている「オブジェクト指向方法論」は、実世界を反映したモデルの採用によって、安定した骨格と機能的な変更に対処できる耐久性を備えており、特に保守面で優位であると主張されている。

我々はいくつかの試行を通してオブジェクト指向分析手法の実務適用性を確認し、現在はその有用性を模索している段階である。検討に当たって我々が本報告で採用したアプローチは、構造化分析手法を比較の対象として取り上げ、オブジェクト指向分析手法と構造化分析手法の両者を用いてシステムの同一部分の記述実験を行い、そこから得られた評価結果をまとめるという方法である。

本稿の構成は以下の通りである。2章は我々が採用した手法を紹介し、3章ではその適用対象の概要を述べる。4章では、両手法の適用結果を評価するための評価項目について述べ、5章で評価結果をまとめる。最後に6章では変更容易性についての考察を述べる。

2．RSAとOOA

今回の記述実験で、我々は構造化分析手法としてHatley/Pirbhaiのリアルタイム SA（以下、RSAと略）を、またオブジェクト指向分析手法としてRumbaughのOMT（以下、OOAと略）を利用した。

2．1　RSAの概要

（1）モデル

RSAでは機能、振る舞い、構造の3つの観点よりモデル化が行われ、その各々は図1に示すような関係で結びついている。中心となるのは機能モデルであり、段階的に詳細化される機能モデルの各々に対して動的モデルとデータモデルが作成される。

機能モデルは、処理と、システム内部のデータ格納庫あるいはシステム外部との間のデータの流れを表現する。機能モデルに現われる処理の実行順序を制御するためのイベントは制御仕様（縦棒）に対する制御の流れとして表現されるが、イベントに対して起動される具体的なプロセスとの対応はここでは表現されない。

動的モデルは、機能モデルで表現されるプロセス群の実行順序を表現する。機能モデルにおいて制御仕様に向かうイベントは動的モデルにおいて状態を遷移させるイベントに対応し、また、機能モデルにおけるプロセス群は動的モデルにおけるアクションに対応する。

データモデルでは、機能モデルのデータ格納庫に格納されるデータの構造が仕様化される。

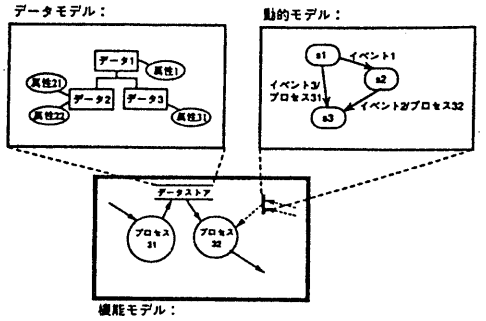

図1　RSAのモデリング

（2）仕様書

RSAでは5種類の仕様書が用いられる。データフロー図とプロセス仕様書は機能を、制御フロー図と制御仕様書は振る舞いを、また実体関連図は構造を記述する。仕様書群の関連を図2に示す。

まず、データ／制御フロー図では、システムが提供する全てのデータの変換処理をプロセスとして表現し、それをシステムの構成単位とする。このプロセスはデータ／制御フローを入出力とする。また、プロセスはより小さな機能のサブプロセスを組み合わせたものとして階層的に詳細化される。次に制御仕様書では、データ／制御フロー図内に存在するプロセス群について、起動条件や実行順序などの振る舞いを定義する。プロセス仕様書では、これ以上分解できないプロセスの仕様を疑似コードで示す。一方、データ／制御フロー図で出てくるデータ／制御フローの一時的な保管場所はデータ／制御ストアとして表現される。データ／制御ストアに格納される情報に構造がある場合には、実体関連図を用いて記述する。システム内ではデータ名やプロセス名を一意に統一する必要があり、これらの名称一覧は要求辞書において定義される。

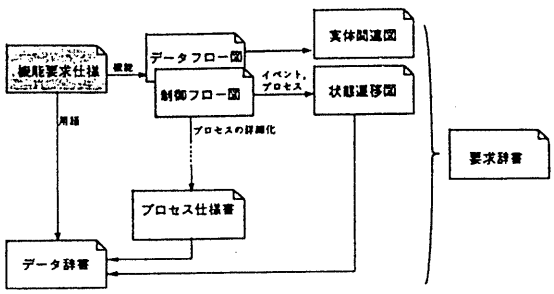

図2　RSAの仕様書一覧

2．2　OOAの概要

（1）モデル

OOAではRSAと同様、3つの観点よりモデル化が行われるが、その各々は図3に示すような関係で結びついている。中心となるのはオブジェクトモデルであり、本モデルで抽出されるオブジェクトの各々に対して動的モデルと機能モデルが作成される。

オブジェクトモデルはオブジェクト間の静的な関係を記述する。オブジェクト図において、オブジェクトはオブジェクト名と属性とメソッドからなり、結線によってオブジェクト間の関連が記述される。他のオブジェクトからこのオブジェクトへのメッセージは、この関連を示す線を通して流れ込むが、どのようなメッセージに対してそのオブジェクトがどう振る舞うかという具体的な側面はここでは仕様化されない。

動的モデルで記述されるイベントは、そのオブジェクトに対して流れ込むメッセージを示しており、またアクションにはそのオブジェクトが持つオペレーションが対応する。なお、動的モデルで表現されるオブジェクトの状態は、オブジェクトモデルで記述された属性の値によって決定されている。

機能モデルでは、メッセージの流れに伴う入出力データが仕様化される。

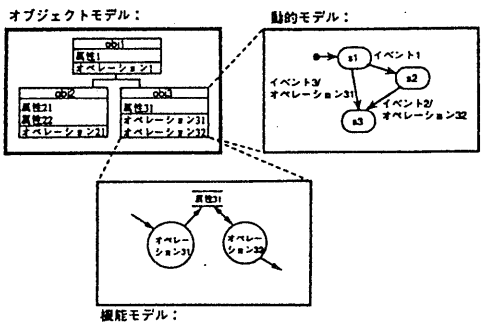

図3　OOAのモデリング

（2）仕様書

OOAでは6種類の仕様書が用いられる。オブジェクト図は構造を、イベントトレース図、イベントフロー図、状態遷移図は振る舞いを、データフロー図は機能を記述する。仕様書群の関連を図4に示す。

オブジェクト図は前記の通り、オブジェクト間の静的な関連を記述する。イベントトレース図は、システムの典型的な実行例を元に、ある機能の実現に必要なオブジェクト間のメッセージのやりとりを時系列で表したものである。イベントフロー図は機能毎に記述されたイベントトレース図を参考に、全ての機能の実現に必要なオブジェクト間のメッセージ通信を時系列を考慮せずに1枚の仕様書にまとめて書いたものである。状態遷移図は、オブジェクト毎に自分の取り得る状態を列挙し、各々の状態において受け付けることのできる外部イベントとそれに対する反応を対応付け、そのオブジェクトで定義される操作がいつ起動されるかを示す仕様書である。データフロー図は、オブジェクト内で行われるデータ変換を記述する。ここではオブジェクト図や状態遷移図で発見された操作関数の入出力データや操作関数間の依存関係を明らかにする。データ辞書は用語を統一するために用いられる。

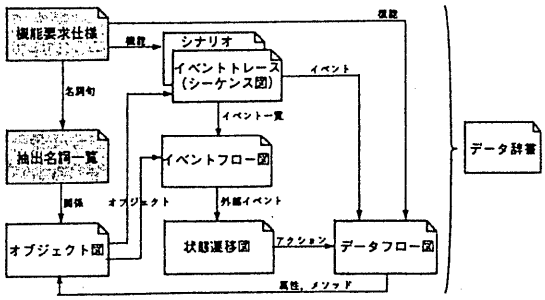

図4　OOAの仕様書一覧

2．3　共通点と相違点

分析指針と表記の2つの観点よりまとめた両者の共通点は以下の2点である：

（1）システムを構造、機能、振る舞いの3つの観点で分け、多角的に分割を行う。
（2）構造、機能、振る舞いを表現するための記述法は、基本的には同じである。

まず、分析指針上の共通点は、システムを構造、機能、振る舞いの3つの観点で分け、多角的に分析を行うという点である。例えば、構造に関しては、RSAではシステム内部に保存されるデータの構造を示し、OOAではシステム全体を構成するオブジェクトの構造を示す。機能に関しては、RSAではシステム内部で実行されるデータ変換処理を示し、OOAでは各オブジェクト内で実行されるデータ変換処理を示す。振る舞いに関しては、RSAではデータ変換処理の実行順序を示し、OOAでは属性の値により異なるオブジェクトの状態を示す。それぞれの使われ方は異なるが、どちらも構造、機能、振る舞いという3つの観点でシステムを分析しようとするアプローチは同じである。

次に、表記上の共通点は、基本的に同じ記法を用いるという点である。つまり、システムの機能を記述するためにデータフロー図を用い、振る舞いを表現する

ために状態遷移図を用い、構造を示すために実体関連図（あるいはその拡張）を利用するというアプローチが同じである。

他方、分析指針と表記の２つの観点よりまとめた両者の相違点は以下の２点である：

（１）システムの３つの観点のうち、何を中心に分析するか
（２）１枚の仕様書が対象システム全体のうち、どの範囲を表現するか

まず、分析指針上の相違点は、システムの３つの観点のうち何を中心に分析するかという点である。これはシステムの構成単位とシステムの概要を示すモデルの相違によって異なる。システムの構成単位は、RSAでは「機能」であり、OOAでは「オブジェクト」である。RSAではシステム構成とは機能の組み合わせを意味し、トップレベルのデータフロー図で示されるプロセス群とそれらの間に流れるデータと制御がシステムの概要を表す。OOAではシステム構成とはオブジェクトの静的な関連を意味し、オブジェクト図で示されるオブジェクト群とそれらの関連がシステムの概要を表す。

次に表記上の相違点は、１枚の仕様書が対象システム全体のうちどの範囲を表現するかという点である。どの範囲を表現するかは、構造、機能、振る舞いを記述する各モデルが何を記述するかによって決まる。

構造のモデルを記述する仕様書は、RSAでは実体関連図である。１枚の仕様書が記述する範囲は個々のデータストアである。一方OOAではオブジェクト図がこれに対応する。１枚のオブジェクト図が記述する範囲はシステム全体である。

振る舞いのモデルを記述する仕様書はRSAでは状態遷移図である。この図で実行順序が定義される対象の範囲は、同レベルのデータフロー図で定義されるプロセス群に限られる。従って、システム全体の大まかな実行順序は上位層に現われる抽象的なプロセスに関する状態遷移図に表現される。OOAでは、状態遷移図が表現する範囲は、各オブジェクトの内部状態であるため、システム全体の状態は全ての状態遷移図の直積となる。

機能のモデルを記述する仕様書は、RSAではデータフロー図である。１枚のデータフロー図が表現する範囲は階層毎に異なる。最上位層ではシステム全体の大まかな処理が列挙される。下位層ほど、システム中の狭い範囲について具体的で詳細な処理が現われる。OOAでもデータフロー図に機能を記述するが、各オブジェクト毎に１枚ずつデータフロー図を用意する。従って１枚のデータフロー図に記述される範囲は、オブジェクト内部の変換に限られる。

３．適用対象

今回、両手法の評価に当たって適用対象としたのは、画像ファイリングシステムの検索機能である。本システムは光磁気ディスクによって大量かつ大容量の画像記憶を行うための装置である。管理の便宜を図るため、本システムは通常のオフィス業務で扱うような書類管理方式を採用している点が特徴である。装置に対するディスクの装填はキャビネット単位で行われ、このキャビネットは複数のバインダを持つことができる。このバインダは更に複数の書類から構成され、書類は複数の頁および画像から構成される。検索のためには論理検索やフリーワード検索、しおり検索など複数の手段が用意されている。ソフトウェアは全体でMLOCの規模であり、大規模システムと呼ぶに相応しい。

４．評価項目

上記適用対象に対して、我々は約５人月でRSAとOOAによる分析を行った。両手法による分析結果を評価するにあたり、我々が採用した評価項目は、手法の枠組みの観点より、手順と表記法、また手法の利用者の観点より、ライタ（分析者）とリーダ（解釈者）を考慮し、以下の４項目に絞った。

	ライタ	リーダ
手　順	分析と設計の役割分担 手順の実施容易性	
表記法	表現の一意性	トレーサビリティ

図５　評価項目

５．評価

５．１　分析／設計の役割分担

分析はシステム外部から観察できるシステムの動作や機能を仕様化することであり、設計は計算機にのせるためにモジュール構造を確定し、時間的・空間的な制約を満たすように最適化することである。分析は外部仕様を定義し、設計は内部仕様を確定するということもできる。

本来、設計時に決定すれば良い判断は、設計フェーズまで遅延されることが望ましい。従って、分析と設計の役割分担が明確であることを評価すべき要因として、ここでは以下の２点に着目した：

１）処理の実行順序の特定が遅延できること
２）ＵＩ方式の特定が遅延できること

1）処理の実行順序の特定が遅延できること

RSAでは、処理の実行順序の特定を分析の開始時より前提としなければいけない。RSAによる本システムの検索機能の分析に関する第０レベルのデータフロー図では、システムの機能として「装填／取り外しを行う」、「ファイリング処理を行う」、「検索作業を行う」といったプロセスが存在する。実際にはこれらの実行順序は幾通りかを想定できるはずであるが、仕様化にあたってはある１通りの実行順序を選択し、データフロー図や対応するレベルの状態遷移図に記述しなければならない。分析の中心となるはずの「検索作業を行う」プロセスはその処理の前処理や後処理を含めて分析モデル全体の中の位置づけが明確になってからではないと具体的な分析を進めることができない。また、一旦この位置づけを明確にしてしまった後は、「検索作業を行う」プロセスはそこで定義される順序でしか有効なものではなくなってしまい、融通性が欠けたものとなる。更に、全体の流れを変更すると個々のプロセスにも影響が及ぶため、分析の後戻りが非常に面倒なものになる。そのため、分析者は最初に決定した処理の流れに固執する傾向があり、本来分析の結果得られる最適な流れが仕様書に反映されにくくなる、という欠点が存在する。

OOAでは、処理の実行順序を分析の初期段階には考慮する必要がない。OOAによる本システムの検索機能の分析に関するオブジェクト図では、システムを構成するオブジェクトとして「バインダ」、「書類」、「書類タイトル」などがあり、例えば「書類」は「書類タイトル」を１つ以上持つという集約化の関係で結ばれているが、この関係は静的な関係であり、「書類タイトル」をもとに「書類」を識別するための処理手順を制限するものではない。従って、オブジェクト図においては「書類タイトル」が検索する際の処理の流れは幾通りかを想定することができるが、処理の流れを特定する作業は後のフェーズに遅延されている。OOAでは、分析の初期にオブジェクト図を記述するが、ここで仕様化するのはシステムの構成要素であるオブジェクトとその間の静的な関係であり、動作や機能はまだこの段階では定義されない。従って分析者は、分析の初期段階では個々のオブジェクトに関して、それが呼ばれる順序や処理方法に捕われることなく分析を進めることができる。

2）ＵＩ方式の特定が遅延できること

RSAでは、ユーザインタフェース方式は分析の初期段階に決定される。RSAではコンテクスト図は、分析の対象範囲と外部環境を切り分けるために分析の最初の段階で記述されるが、この図では同時に外部とのデータのやりとりも記述される。今回の例では、「画像ファイリングシステムの検索機能を実施する」というシステムは「ユーザ」や「ディスプレイ」などの外部環境とデータをやりとりする。「操作コマンド」や「バインダ選択メニュー」などは外部とやりとりされるイベントやデータの例であるが、これらを仕様化するためには、かなり詳細なレベルのユーザインタフェースまでを決定しなくてはならない。

OOAでは、ユーザインタフェース方式は分析の後半まで決定が遅延される。分析の初期において問題領域のオブジェクトとユーザインタフェースとの関係をオブジェクト図に記述するが、ここで示す関連は前記同様、静的な関連であり、方式や流れるデータを規定するものではない。具体的な情報は、動的モデルの仕様化が行われるまで検討が遅延される。

本来、分析者は、このような実行順序やユーザインタフェース方式の決定を後回しにすることにより、システムの柔軟性を高める努力をすべきである。

5．2　手順の実施容易性

手順の実施容易性は、分析者とともに分析結果を解釈する者にも重要な要件である。特に大規模システムの分析には、分析手法がシステムの複雑さを解消する手段を提供することが必要となる。システムの複雑さの解消手段として、ここでは以下の３点に着目した：

1）システムを複数の観点から独立に分解できること
2）システムを段階的に分割すること
3）システムの構成要素に対する命名が容易なこと

1）システムを複数の観点から独立に分解できること

システムを分析する際、観点を分離して仕様化する側面を限定し、それぞれの観点による分析を他の観点からの分析と切り離すことが可能ならば、システムの複雑さを解消することができるはずである。

RSAでは、機能という観点ではデータフロー図、振る舞いという観点では状態遷移図、構造という観点では実体関連図というように観点毎に異なる種類の仕様書が用意されている。しかし、仕様書間の関連が密であるため、事実上、ある仕様書を記述するためには他の仕様書の記述と同期をとる必要があり、観点毎に仕様書は分かれているが、記述作業は観点毎に分離して行うことができない。例えば今回の分析においては、データフロー図にはプロセスの実行を制御する「電源ON」や「操作コマンド」といったイベントを制御バーに向かう制御フローとして記述しなければならない。実際

には、これらの制御フローは、データフロー図上ですぐに見つかるのではなく、状態遷移図でプロセス群の実行順序を考えて初めて明らかになる。従って、データフロー図に現われる制御フローを明らかにするためには状態遷移図を完成させる必要がある。逆にデータフロー図の記述中に分析者が発見できなかったプロセスを、状態遷移図の作成中に見つけだすことができる場合もある。例えば、「操作コマンドメニューを表示する」というプロセスは、状態遷移図において「装填作業」や「検索作業」などが終了した後、「コマンド待ち状態」に戻る時に起動することが必要であることが分かったために、データフロー図に追加されたプロセスである。このように、データフロー図と状態遷移図は同期をとりながら作成していく必要がある。

OOAでは、構造という観点ではオブジェクト図、振る舞いという観点では状態遷移図、機能という観点ではデータフロー図というように観点毎に異なる種類の仕様書が用意されている。これらの仕様書間は独立性が高いため、記述作業は観点毎に分離して行うことができる。例えば、オブジェクト図と「書類」の状態遷移図について考えてみる。オブジェクト図において現われる「書類」は「頁」や「書類タイトル」との関係は、「書類」がそれらのオブジェクトを構成要素として所有するという静的な集約化の関係に基づいて記述される。この関係を定義する時、「書類」が「書類タイトル」を使って、どのようなイベントを受け取り、どのオブジェクトを起動させるか、といった振る舞いを考えるわけではない。一方、状態遷移図を記述する時には、分析者は既に作成済みのオブジェクト図を参照し、オブジェクト間の関係を前提としてオブジェクト毎に動作を規定していくことができる。例えば「書類」に関しては、オブジェクト図において「書類タイトル」との集約化の関係を辿ることによって、状態遷移図では「書類タイトル」に対して「タイトル一覧表示」を行うメソッドを呼び出すことを記述することができる。このように、オブジェクト図と状態遷移図とは、比較的穏やかな関連しかない。

このように、RSAと比較してOOAでは観点毎に仕様書の独立性が高い。従って、分析者が観点を分離して仕様化していくことができるという点で、OOAの方が優れていると考えられる。

2）システムを段階的に分割すること

システムを分析する際、システムを段階的に分割して作業範囲を狭めていくことができれば、システムの複雑さを解消することができるはずである。

RSAでは機能をトップダウン方式で詳細化していく。この方法は、概要を明らかにした後、各項目別に詳細に調べていくという人間の思考過程に合致している。例えば、今回のデータフロー図の例では、「検索作業を行う」というプロセスは、システムの利用者が検索作業を行うためには、「複数の書類が該当したときの処理を行う」や「画像を表示する」といった処理が必要であることから、サブプロセスへの分解を自然に行うことができた。

ところが、上位レベルのプロセスのグルーピングの仕方を変更しようという段になって、記述の大幅な変更が必要になるとともに、対応する状態遷移図や下位の全てのプロセス群にも影響が大きく及ぶことが明らかになった。我々は第0レベルのデータフロー図の初期バージョンを作成した際に、「装填をチェックする」や「ユーザにバインダを選択させる」といった必要以上に細かいプロセスを記述したため、後にそれらを「装填／取り外しを行う」や「ファイリング処理を行う」というプロセスに組み込むような修正を加えたが、これによってコンテクスト図と実体関連図以外は全て作成し直す、という大規模な作業が発生した。変更のために大きな負担が発生するのが本手法の欠点である。

OOAでは、段階的な分割は行わず、構造をボトムアップ式に組み立てていく。オブジェクト群のグルーピングはオブジェクトモデルの作成よりもむしろ動的モデル（特にイベントトレース図）の単純化のために必要であったが、この場合もオブジェクトを抽出する前にグループが現われるのではなく、オブジェクトが抽出された後にグルーピングする方が自然であった。段階的な分割を行わないため、OOAでは各構成要素を問題領域から抽出する作業は難しい。しかし、一度構成要素を決定すると、その動作や機能を仕様化する対象は個々の構成要素毎に行えば良いため、対象範囲を狭めることができる。

3）システムの構成要素に対する命名が容易なこと

システムを分析する際、識別子としての用途を考慮しながらも、開発者に過度の負担をかけないで済むような命名法が採用されていれば、システムの複雑さを解消することができるはずである。

RSAでは、システムの各構成要素の名前を識別子として用いることができるように、各構成要素名をシステム全体において一意に定めなければならない。つまり、全てのプロセス名、データ／制御フロー名、ストア名といった情報は、種々の仕様書間や異なる階層間を含め、システム全体で一貫した名前を持つように決める必要がある。一貫するということは、同じ情報に対しては同じ名前を、異なる情報に対しては異なる名前を

与えるということである。後者の例は、例えば第０レベルのデータフロー図に現われる「検索作業を行う」というプロセスは、その下位において「検索を実行する」という本質的な機能を持つプロセスとこれを補足するプロセス群に分割される。ここで「検索作業を行う」と「検索を実行する」は、異なるプロセスなので違う名前をつける必要がある。システム全体について一意に命名しようとすると、抽象的な階層におけるプロセス名と詳細化された階層におけるプロセス名を無理に変えたり、異なるプロセスの下位同士で同一処理を行う場合にも別名にしておいたりする必要が生じる。大規模システムでは、このような名前の一貫性を保持するためには、常にシステム全体で利用される名前を個々の担当者が把握している必要がある。これは現実的には非常に難しい作業である。

OOAでは、システムの各構成要素の名前に関しては、各構成要素名がシステム全体においてではなく、オブジェクト毎に一意に定まっていれば良い。同じ情報に対しては同じ名前を持つように決める必要があるが、異なる情報に対してもそれらが概念的に同じ内容を表していれば同じ名前を与えても良い。例えば、オブジェクト図では集約関係によって結ばれている「バインダ」、「書類」、「頁」の間で流れるメッセージについて考える。「バインダ」から「書類」に流れるメッセージとして「しおり名表示」がある。これは「書類」が持つ「しおり名表示」というメソッドを起動するためのメッセージである。一方、「書類」はそのメッセージを受けて更に「頁」に対して「しおり名表示」というメッセージを送るが、これによって起動されるのは「頁」が持つ「しおり名表示」というメソッドである。この場合、書類が持つメソッドと「頁」が持つメソッドは同じ名前でも明確に区別され得る。このように概念的に同じ情報に対して同じ名前を与えることができるということは、大規模システムを開発する上で、非常に有効であると考えられる。

従って、RSAと比較して、OOAでは情報に名前を与える作業が容易であるといえる。

５．３　表現の一意性

表現の一意性に関しては、今回RSAとOOAの記述実験の主担当がそれぞれ１人だったため、分析者に依存する表現の違いに関する客観的な比較材料は得られていない。以下では、構造、振る舞い、機能の表現の一意性に関して分析者の主観を述べるにとどめる。

1）構造の表現の一意性

RSAでは、実体関連図を用いてデータ構造を記述する。実体関連図にはいろいろな拡張型が提案されているが、基本的には実体とそれらの関係を表現するための記述法であり、データ構造を表現する上では、分析者に依存して表現が異なることが少ないと思われる。

OOAでは、オブジェクト図を用いて構造を記述する。ここで記述されるのは、システムを構成するオブジェクトとその間の関連である。この表記においては、従来の実体関連図と比較して、オブジェクトの定義については、あるデータをオブジェクトとするか、他のオブジェクトの属性とするか、あるいはオブジェクト間の関連が持つ属性とするかなどを選択せねばならない。またオブジェクト間の関連に関しては、継承、集約化、あるいは単なる静的なつながりのどれでモデル化するかという選択をしなくてはならない。従って分析者によって表現が異なる可能性が高い。

2）振る舞いの表現の一意性

RSAでは仕様書の種類の上でも、同一内容を記述する手段の上でも、状態遷移図、状態遷移テーブル、状態遷移マトリクスといった複数の記述方法が用意されている。表現の選択は分析者にまかされている。

OOAではステートチャートを用いる。ステートチャートでは並列状態や入れ子の状態を記述できるため、同一内容を記述する手段については多少自由度がある。

振る舞いに関しては、両手法とも状態遷移図で動作を規定する対象が限定されている（RSAでは同レベルのデータフロー図のプロセス群、OOAでは個々のオブジェクトが持つメソッド群）ため、分析者による表現には大差がないと考えられる。

3）機能の表現の一意性

RSAでは、データフロー図を用いて機能を表現する。データフローの表記法は詳細に決められているが、同一の内容を記述するにもいろいろな表現が可能である。プロセスの粒度に関する基準、イベント駆動型かデータ駆動型か、複数プロセスで同様な処理部分がある場合に共通化するかどうか、あるいは複数のプロセスが同時に起動可能なモデル化にするかどうか、更にはあるプロセスを階層のどのレベルに置くかなど、分析者のセンスで決まる自由度が非常に大きい。

OOAでは、データフロー図の書き方は詳細に規定されているとはいえないが、記述する範囲がオブジェクト単位なので、記述すべきプロセスの粒度が細かいことが多く、階層化が不要になる分だけ分析者による表現のバリエーションが少ない。

5．4　トレーサビリティ

1）処理のトレーサビリティ

RSAでは、データフロー図と状態遷移図との対応付けを用いることにより、処理の流れを追跡しやすい。これは、データフロー図に対応する状態遷移図を参照することにより、データフロー図上で次に実行されるデータ変換が明確になるからである。しかも仕様書の枚数は十分に追跡可能な程度に少ない。

OOAでは、状態遷移図は個々のオブジェクト毎に記述されている。従って、個々のオブジェクトが特定の状態にある場合に、どのようなイベントに対してどのような処理を実行するかということは明示されている。しかし、システムがある特定の状態にある場合に、どのようなイベントを受け付け、どのような処理を行なうことができるかについて一覧表示させる手段がない。つまり、システム全体の流れを表現する手段がないので、処理の流れを追跡することは難しい。処理の流れを記述する１つの方法は、イベントトレース図で記述することであるが、これはシステムの果たす多くの機能の１つの流れに過ぎず、他の機能についてもそれぞれ、別のイベントトレース図を用意しなければならない。このようにOOAでは多くの仕様書を参照しなければシステムの処理の流れを把握することができない。

2）データの関連のトレーサビリティ

RSAでは、データ／制御フローの相関関係が曖昧になりやすい。これは上位層のプロセスに入ったフローが下位層で複数のデータフローに分岐する場合、上位層では複数のフロー群を１本のフローにまとめることができるためである。例えば、「ユーザ指定検索条件」というフローは、その下位で複数の情報にに分離する。このようなデータ間の関連は、要求辞書を参照しなければ追跡できない。RSAでは階層間を渡るデータ名に関しては一貫しているとは限らないので、データフロー図や状態遷移図だけでデータの対応を識別できない。

OOAでは、オブジェクト図がデータの関連そのものである。関連を示す結線を辿ることで、オブジェクト間のデータの流れに関し、見当付けが可能である。

6．変更容易性への影響に関する考察

RSAは分析を容易にする手法としてよりも、分析が完了した時点で仕様書の見通しが良い面において長所を強調することができる。逆にOOAは分析作業を効率的に行なえる手法としての良さが際だっているが、仕様書は見通しが悪い。これは、RSAが分析の過程で処理の内容や順序を完全に決定するようなモデル化を行なうのとは対象的に、OOAでは部分的に処理の内容や順序に柔軟性を持たせるモデル化であり、完全に一意には決定しないこととも関連があると思われる。

変更が容易であるためには、まず仕様変更箇所の特定が容易であり、次に変更作業そのものが容易であることが望ましい。両手法の評価結果より考えると、仕様変更箇所の特定は、それがシステムの環境や資源に関する変更であるならば、OOAの方が、またシステムレベルの機能に関するものである場合にはRSAの方が特定しやすいと考えられる。但し、仕様書の修正量として見た場合の変更作業そのものは、変更の種類によらずOOAの方が少ないと考えられる。

7．おわりに

本報告では、オブジェクト指向分析手法（RumbaughのＯＭＴ）と構造化分析手法（Hatleyのリアルタイム ＳＡ）を、画像ファイリングシステムの検索機能の分析に適用した経験をもとに、オブジェクト指向パラダイムと構造化パラダイムの違いを、分析／設計の役割分担、手順の実施容易性、表現の一意性、トレーサビリティの４つの観点から検討した結果を述べた。仕様変更の容易性は重要なテーマなので、今回の分析をベースに、今後とも継続して検討を続けていく予定である。

参考文献

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

[Rumbaugh 91] J.Rumbaugh, M.Blaha, et al. "Object Oriented Modeling and Design", Prentice Hall, 1991