ソフトウェア工学 83－17
（１９９２．２．７）

オブジェクト指向分析の定式化

小高 信人* 岸本 芳典† 本位田 真一‡
taka@ipa.go.jp ksmt@ipa.go.jp honiden@ipa.go.jp

情報処理振興事業協会
新ソフトウェア構造化モデル研究本部

ケーススタディに基づいたオブジェクト指向分析プロセスのモデルの概要と、その評価について述べる。ここで提案するモデルは、Peter Coad と Edward Yourdon の提唱するオブジェクト指向分析 (OOA) 方法論をベースにしている。我々は、分析プロセスの実態を明らかにするために、分析作業を段階的に詳細化し、個々の分析ステップ間におけるデータフローとコントロールフローを定義した。ここでは、コントロールフローのバックトラックを分類することにより、round-trip 型の分析プロセスを支援する。また、それらの作業ステップにおける分析者の意思決定を支援するために、ケーススタディから得られた分析ノウハウ (Practical Tip) を採り入れた。

CASE-STUDY BASED SPECIFICATION PROCESS MODELING FOR OBJECT-ORIENTED ANALYSIS

Nobuto Kotaka* Yoshinori Kishimoto† Shinichi Honiden‡

Laboratory for New Software Architectures
Information-Technology Promotion Agency, Japan
Shuwa-Shibakoen-3chome Bldg., 3-1-38 Shibakoen, Minato-ku, Tokyo 105, Japan

Our research is in formalizing a specification process for Object-Oriented Analysis(OOA). To do this, we built a basic specification process model according to case study result. The prototype we propose in this paper was made considering the OOA methodology – proposed by Peter Coad and Edward Yourdon – with the addition of our own application-specific activities. By dividing the model into two parts, Data Flows and Control Flows (including backward control flows), we could analyze the chains of dependence between data and activities in a complex round-trip process. As a result of this study, we also specified some practical tips to execute the activities effectively.

*富士通エフ・アイ・ピー (株) より出向
†(株) 日立製作所より出向
‡(株) 東芝より出向

* Also with Fujitsu Facom Information Processing Corp.
† Also with Hitachi Ltd.
‡ Also with Toshiba Corp.

1 はじめに

今日のソフトウェア産業においては、開発工程の途中における仕様変更が頻繁に行なわれることにより発生する作業の手戻りが、ソフトウェアの生産性の向上を阻害する大きな要因となっている。ところが、要求分析や設計といった、いわゆる上流工程を支援することを謳った既存の方法論で、このような作業の手戻りを積極的に容認し支援する機能を有したものは、残念ながら見受けられない。そのため、熟練分析者は方法論のごく一部分を利用するに留まるか、まったく見向きもしないかのいずれかとなる。一方、経験の浅い分析者について観察すると、一般的に、方法論は具体的な分析作業における細部のノウハウを提供していない場合が多いために、方法論をうまく活用できていないのが現状であると考えられる。

我々はこれまでに、現存する様々なオブジェクト指向方法論について調査を行なってきたが[Mayer88], [Coad91a], [Shlaer91], [Wirfs90], [Rumbaugh91] 、前述した問題点はこれらの方法論の場合においても解決されていないのが現状である。しかし一方では、オブジェクト指向がこれからのソフトウェア開発方法論の主流を占めるであろうことは、概ね了解された事項であると考えてよいであろう。そこで我々は、オブジェクト指向によるソフトウェア開発方法論において、前述の問題点を解決することを目標とした研究を行なっている。本稿では、オブジェクト指向分析 (OOA) プロセスの定式化の試みとその評価について報告する。

2 OOAプロセスの特徴

OOAプロセスにおいては、ソフトウェア仕様上に直接に現れるデータおよび一時的に生成される中間生成物間の参照関係や、それらの影響を受ける分析プロセスのコントロールフローが複雑な相互作用を及ぼしあっている。そのため分析者は、ある作業項目を実行する際に、他のデータ項目をどう決定するかといった予見を要求されることが少なくない。それゆえ、試行錯誤やデータの生成の時間的な逆転現象に起因する分析作業の手戻り (バックトラック) が発生することとなる。また、OOAにおいては、例えば、クラスの候補として考えられていた項目が、後に他のクラスの属性であることが判明する場合 (およびその逆の場合) のように、データ間の相互依存性が密接であると考えられるため、その分析プロセスは本質的に並行作業的な性格が強い。

また、クラスやその属性およびサービス (メソッド) に代表される最終成果物以外の中間生成物がかなり多くなるため、机上の分析作業は作業能率の面でかなり困難となる (これは、後述するケーススタディの経験から得られた実感でもある)。このような性質を有する分析プロセスを一人の分析者が実行する場合、仮説を含めたすべてのデータおよび意思決定の過程を記録し、状況に応じて提示することが可能な支援ツールが必要となる。

3 定式化の指針

既存のOOA方法論からヒントを得て、方法論のエンドユーザである分析者にとって役に立つ支援環境を実現するために、我々は、既存のOOAに関する方法論についての調査を行なった。その結果、我々のOOAプロセスモデルのベースとして Peter Coad と Edward Yourdon の方法論を採用することとした。次に、この方法論に則ってケーススタディを行ない、そのプロセスと結果を分析することにより、OOAプロセスモデルを作成することとした。なお、定式化に当たって以下の指針を用意した。

- OOAプロセスを Coad & Yourdon 法における5つの最終成果物レイヤ[1]を最上位とした幾つかのプロセスグループに整理し、それを段階的に詳細化すること。
- OOAプロセスにおける最下層の分析項目 (これを **activity** と呼ぶ) では、(妥当性の問題は別として) 一つの解だけが存在するように、activity を十分に詳細化すること。
- このようにして定義した多層のプロセスグループにおいて、同階層のプロセスグループ間、およびプロセスグループ内のサブプロセス間、のデータフローおよび制約を列挙すること。
- 分析作業の無駄な手戻りを防止するため、データフローを基に、データの生成順に着目して、プロセスコントロールフローを生成すること。
- 各 activity における、分析者の意志決定を支援するための (ケーススタディおよび文献調査から得られた) 分析ノウハウ (これを **practical tip** と呼ぶ) を整理すること。

[1] クラス・構造・属性・サービス・サブジェクトの5つ。

4 ケーススタディの概要

OOAプロセスモデルを作成するに当たり、既存の方法論は概略を述べているに過ぎないため、細部を補うための知見を得ることを目的として、ケーススタディを行なうこととした。まず最初に、「Automating a Helicopter Landing」[Davis90]を題材として分析を行ない、OOAプロセスのプロトタイプモデルを作成した。その後、「Automated Teller Machine Example」[Rumbaugh91]と「A Mobile Phone System」[Duke91]の2つの例題により、プロトタイプモデルの評価および改良を行なった。

我々のうちの一人が、これらすべてのケーススタディを行なった。最初のケーススタディは、途中経過の検討のための週1回程度のレビューを行ないながら、約3カ月間に渡り実施された。分析に要した実時間は約250時間であり、この中にはCoad & Yourdon法の細部についての調査の時間も含む。被験者は、Coad & Yourdon法を始めとする各種のOOAに比較的精通しているが、それでも、OOAプロセスのコントロールフローをすべて理解しているわけではなく、また、遭遇した個々の意志決定場面において、方法論が支援策を提供しているか否かを調査したため、通常の分析作業に比べて多くの時間を費やすこととなった。その結果として、方法論は概念と最終成果物を提示するが、細部の戦略については貧弱であることが確認された。

その後、1カ月程度で分析時のログに関する検討を行ない、プロトタイプモデルを作成した。分析時に被験者が残したログは以下の通りである。

- Coad & Yourdon法で定義される、クラスや構造などの最終成果物。
- Coad & Yourdon法では定義されていないが、分析プロセスにおいて頻繁に参照された中間生成物。
- 被験者が残したメモ類。
- 被験者と観察者のレビュー(インタビュー)ドキュメント。

プロトタイプモデルを作成した後の、2つのケーススタディは同じ被験者が3カ月後に行なった。各事例についての作業量は、分析プロセスがプロトタイプモデルによって示されているために、各々およそ1週間で完了した。次章以降、プロトタイプモデルの概要を述べ、それについての考察を行なう。

5 OOAプロセスモデル

OOAプロセスモデルは、段階的に詳細化された分析項目、および分析項目の分析時に用いられるノウハウ(practical tip)と、データフローモデル、コントロールフローモデルの4つの要素から構成される[Kotaka91b]。

5.1 分析項目およびpractical tip

Coad & Yourdon法に関する調査とケーススタディの結果から、我々は分析プロセスにおける作業項目を、**Step**, **Sub Step**, **Activity**, の3段階の階層に分割した。

- **Step**

OOAプロセスは、Coad & Yourdon法の5つのレイヤを基本とした7つのプロセスグループにより構成される。このプロセスグループを**Step**と呼ぶ。我々は、クラスの生成に関するStepを、クラス候補の生成とその検証を行なう2つのStepに分割し、また、実際の分析の前処理として、仕様ドキュメントからのデータの抽出を行なうステップを用意した[Wirfs90]。

- **Sub Step**

Stepはそれぞれ幾つかの**Sub Step**により構成される。Sub Stepは一個の中間データもしくは最終成果物を生成するための一連の分析項目の集合体である。

- **Activity**

Sub Stepを構成する個々の分析項目を**Activity**と呼ぶ。ActivityはOOAプロセスの最小構成単位であり、そこで分析者は、提示されたノウハウ(practical tip)に基づいて、単純化されたサブ問題を解く。

本モデルではまた、Activityにおいてより良い解を生成するために、「データのどういった側面を注視すべきか」や、「データをどのように適用すべきか」といったノウハウ(practical tip)[Rumbaugh91]を整理・活用することを重視している。このようなpractical tipの収集については、人手に頼る他ないのが現状であり、意志決定の際に分析者の採用した理由の抽象化に関する方策が必要とされる。

例として、ケーススタディから得られた、クラスの属性定義に関するプロセス構造を表1に、ノウハウの部分を表2に示す。

Sub step	Triggers	Input data	Output data	Activities
4.1	1. The completion of the Step3.2. 2. The backtrack from the Step 5.2 due to the failure. 3. The backtrack from the Step 5.4 due to the failure. 4. The completion of the Step3.5.	1. Current all classes. 2. Noun phrases remained. 3. Relations of ownership of Noun phrases.	1. Names of attributes. 2. Values of attributes. 3. Not always applicable attributes.	1. Check the remaining noun phrases to be attributes. 2. Check the relation of ownership to be attributes. 3. Fill up the attributes not written in the original specifications. 4. Check always applicable attributes.
4.2	1. The completion of the Step4.1. 2. The backtrack from the Step 5.4 due to the failure. 3. The backtrack from the Step 6.1 to verify the results. 4. The backtrack from the Step 6.2 to verify the results. 5. The backtrack from the Step 4.3 due to the failure.	1. Current all classes. 2. Names of attributes. 3. Values of attributes.	1. Owner classes of attributes. 2. Refered attributes of classes. 3. Relations of reference among classes.	1. Pick up the classes which need the same attribute. 2. Decide the owner class of the attribute. 3. Change the attribute of classes to a reference attribute except the owner class. 4. Write the relations of class in the Class Relation Diagram.
4.3	1. The completion of the Step4.2. 2. The backtrack from the Step 5.1 due to the failure.	1. Current all classes. 2. Names of attributes. 3. Values of attributes. 4. Owners of attributes.	1. Complete set of values of attributes. 2. State transitions of attributes.	1. Define the values of attributes in classes. 2. Draw the State Transition Diagram of attributes.

表 1: 属性定義プロセスのプロセス構造

(1) If there is a certain verb phrase whose direct / indirect object is not a Class and its subject is a Class, the object is likely to be an Attribute.

(2) The above-mentioned candidate of an Attribute is usually a Reference Attribute.

(3) A modifier of a certain noun phrase that means a relation of ownership between the modifier and the noun phrase is likely to be a candidate of an Attribute.

(4) A noun phrase which means constant value is likely to be an Attribute. Thus look for a noun phrase which has the relation to it. Unless the appropriate noun phrase isn't found, create a new name suitable for the meaning of the noun phrase.

(5) If there appears a Class which has no Attribute, the propriety of the Class ought to be judged after the definition of the Service has completed.

(6) The Class which has neither an Attribute nor a Service is likely to be an Attribute of another Class.

(7) When the specification process has completed, except the Attributes which are not referred to by any Services.

(8) The Attribute Connections are the candidates of Generalization-Specialization Structures.

(9) Divide the range of an Attribute value into some regions paying attention to the meaning of the Attribute in the problem domain.

表 2: 属性定義プロセスにおける分析ノウハウ (practical tip) の例

5.2 データフローモデル

データフローモデルではOOAプロセスにおける、Step, Sub Step, Activity 間の入出力データとその流れを定義する。Sub Step レベルのデータフローモデルを図 1 に示す。データフローには以下の 3 種類が存在する。

- 連続プロセス間順方向データフロー (図 1:矢印 1)

コントロールフロー上、次に実行されるプロセスにおいて、当該データを用いる場合。

- 不連続プロセス間順方向データフロー (図 1:矢印 2)

時間的に不連続ではあるが、データの生成時間とプロセスの実行時間の関係の整合性がとれている場合。

- 逆方向データフロー (図 1:矢印 3)

データの生成順とプロセスの (期待されている) 実行順が逆転している場合。この場合、データの生成をトリガーとする作業の後戻りが発生する。

本モデルにおいては、各 Activity における必要データを洗いだし、それらのデータを生成する Activity と当該 Activity 間の実行順序関係を、可能な限り自然な形で記述することに留意した。その結果、詳細なプロセスレベルでは順方向のデータフローだけでプロセスを定義することができた。つまり、個々

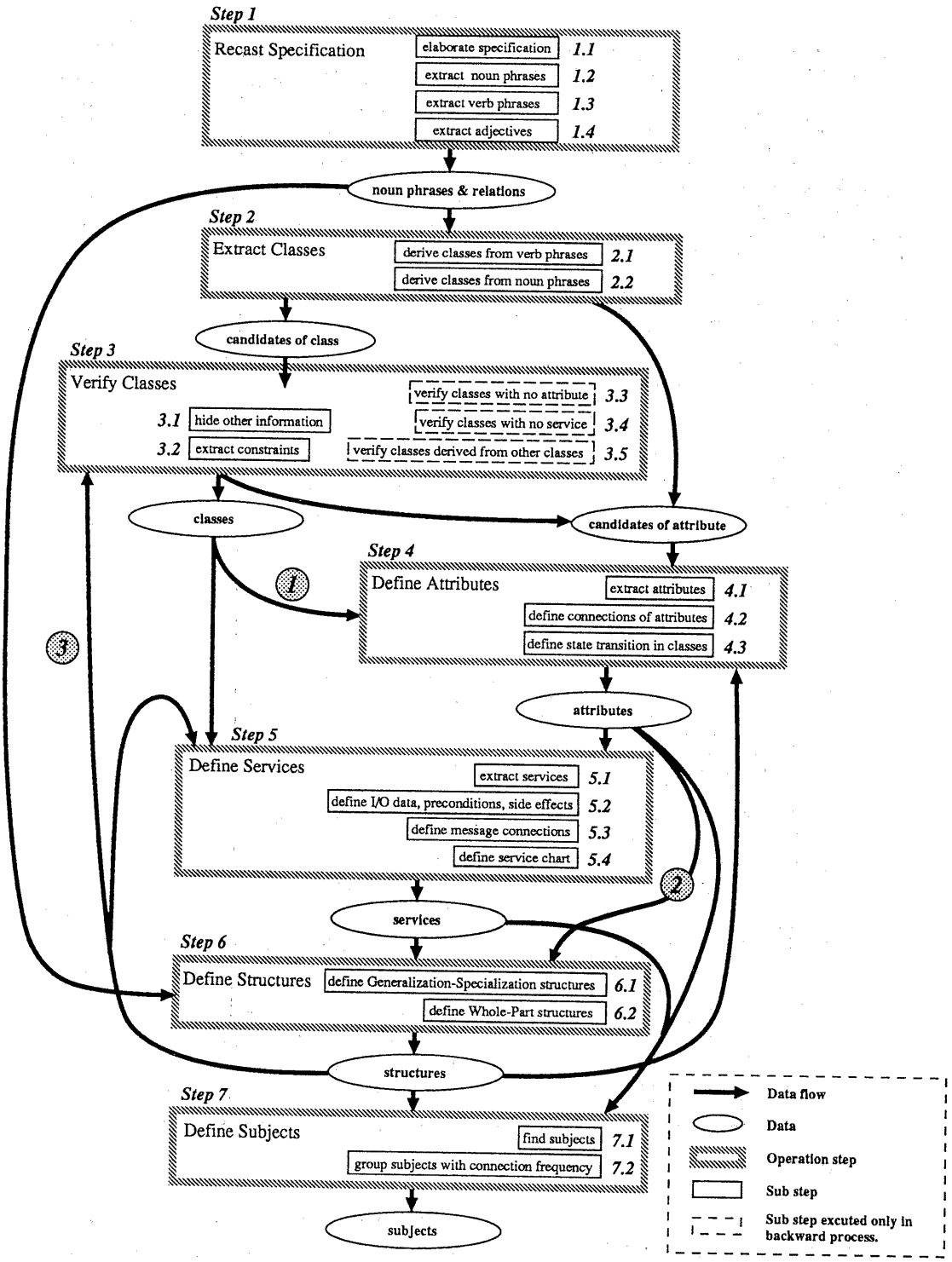

図 1: OOA プロセスにおけるデータフローモデル

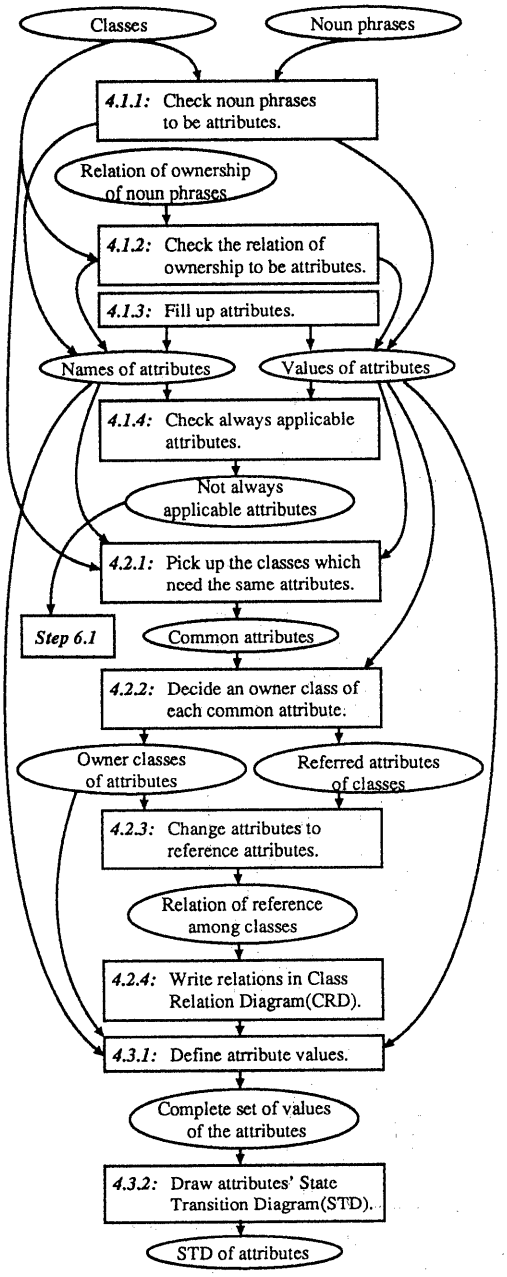

図 2: 属性定義プロセスの Step 内データフロー

の Step 内における Sub Step の最終出力データは、同一 Step 内外の Sub Step の入力データとして用いられるが、Sub Step 内で個々の Activity が生成する一時データは、その Sub Step 内でのみ参照される。例として、属性定義 Step のデータフローを図 2 に示す。図中の矩形はプロセス (Activity) を、楕円形はデータを表している。また、Activity に付与された番号は、表 1 で用いられている番号に対応している。図中の 'Not always applicable attributes' は、他の Step(図 1 参照) で使用される中間生成物である。

5.3 コントロールフローモデル

コントロールフローモデルは OOA プロセスの実行順序を定義したものである。図 3 に Sub Step レベルのコントロールフロー図を示す。

我々は、ケーススタディにおける分析者のプロセス実行過程を分析し、そこから、以下に述べる 4 種類のコントロールフローを抽出した。

- フォワードコントロールフロー

  図 3 において、灰色の矢印で示されるウォータフォール型のコントロールフローで、分析の基本的な作業順序を示す。

- データ生成遅延によるバックトラックフロー

  図 3 において、黒破線で示されるコントロールフロー。属性やサービスの定義 Step において、それらを持たないクラスが発見された時に発生する。

- Activity の失敗によるバックトラックフロー

  図 3 において、黒実線で示されるコントロールフロー。分析者のミスなどの理由により発生する作業の後戻りを表す。このコントロールフローは、各 Sub Step および Activity 間のデータの依存関係により定義される。そのため、分析対象となる問題の領域には独立した、抽象度の高いコントロールフローであると考えられる。

- 既出データの検査のためのバックトラックフロー

  図 3 において、黒一点鎖線で示されるコントロールフロー。当該 Sub Step を実行することにより得られたデータによって、以前に生成したデータの正当性の検証および補強が可能となるようなコントロールフローを表す。

我々は、これらのコントロールフローについて、トリガーとなるイベントと必要なデータを整理を行ない、バックトラックフローについてはその他に、実行後の影響範囲についても整理した [Kotaka91b] 。

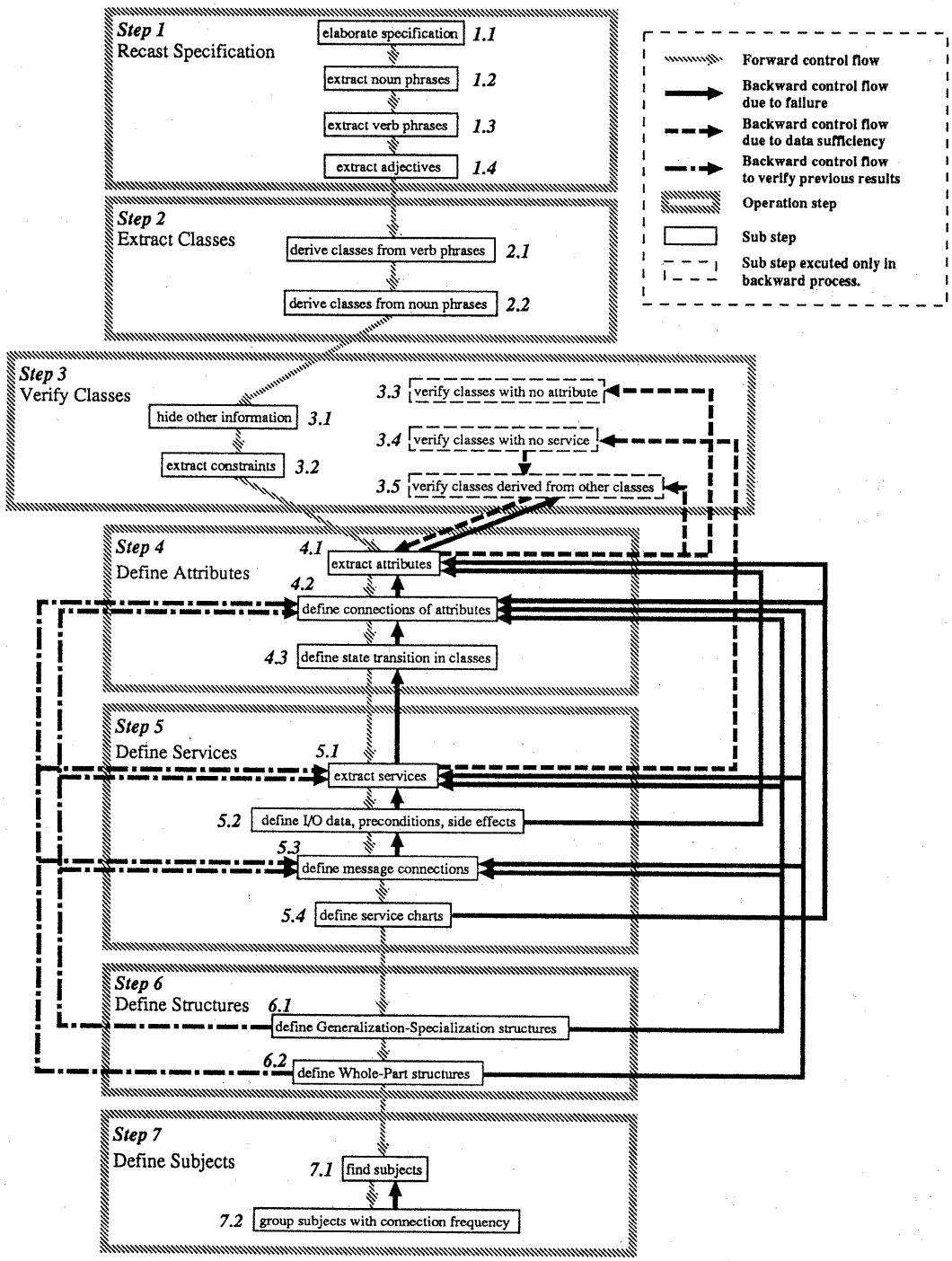

図 3: OOA プロセスにおけるコントロールフローモデル

6 モデルの評価

6.1 プロトタイプモデルの評価

本プロトタイプモデルでは、以下に述べる点において、Coad & Yourdon 法をベースとしたOOAプロセスの定式化および視覚化が達成できたと考えられる。

- OOAプロセスを3段階に詳細化した。第1レベル (Step) は Coad & Yourdon 法の最終成果物を中心とした7つであり、第2レベル (Sub Step) では中間生成物ごとにプロセスを細分化した。第3レベル (Activity) では、一つの中間生成物を生成するための一連の手順を、意志決定の単位で分割した。
- OOAプロセスにおけるデータの参照関係を検証し、Coad & Yourdon 法では明確に定義されていない、中間生成物と最終成果物の関係を明らかにした。しかる後に、中間生成物間の依存関係を、それらを生成する Activity の実行順序の決定に反映させた。これにより、無駄なコントロールフローを排した。
- OOAプロセスにおけるデータフローを検討することにより、不可避的な作業の手戻りが存在することを発見した。その上で、個々のバックトラックフローの発生原因を調査し、3つの型に分類した。また、個々のバックトラックフローについて、そのトリガーとなるイベント, 必要なデータ, 影響範囲を明らかにした。
- 個々の Activity における意志決定を支援するノウハウ (practical tip) を、ケーススタディにおける分析者の判断記録から整理した (この点に関しての考察を次節で行なう)。

6.2 複数の問題への適用評価

プロトタイプモデルを作成した後に、その有効性と汎用性の検証のために2つのケーススタディを実行した。このケーススタディの目的は、プロトタイプモデルにおいて問題領域に依存する部分とそうでない部分を明らかにすることである。我々の当初の意図としては、問題領域に依存しないコントロールフローとデータフローを定義こと、および、OOAプロセスにおいて問題領域に依存する部分を practical tip に集積することにあった。その点についての評価を中心に、問題領域, 分析者, OOA 方法論, の3つのカテゴリの知識について、ケーススタディからの考察を以下に述べる。

- プロセスの問題領域への依存性

2つのケースを通じて、プロトタイプモデル上のコントロールフローのみで分析プロセスを完了することができた。その意味では、当初の意図は達成されたことになる。つまり、ここで提案したモデルは、Coad & Yourdon 法をベースとした問題領域に依存しない OOA プロセスということができる。ただし、検証のために用いた例題がリアルタイム制御系の問題に偏ったことから、汎用性の立証については、今後、様々な問題領域における例題を用いた検証が必要であると考えられる。

- 知識表現に関する課題

本モデルにおいては、問題領域に固有の知識は practical tip として実現される。ここで言う知識とは、

1. 分析者の経験的ノウハウ
2. 方法論が提供するノウハウ
3. 問題領域の専門家の提供するノウハウ

の3種である。方法論が提供するノウハウについては practial tip として、予めモデル内に用意する。本モデルにおいては、Coad & Youdon 法をはじめとする OOA 方法論の教科書から得られた practical tip を用意した。しかし、用意された practical tip は、OOA 全般に関するものが多く、その意味ではあまり実用的とは言えなかった。これらの practical tip に依らない分析者の意志決定の理由は、全くの問題領域依存の知識である。そこから経験的ノウハウを抽出することは、問題領域に依存した命題を抽象化することであり [Matsuura92]、帰納推論的なメカニズムを必要とする。このメカニズムの計算機による実現は簡単ではないため、本研究においてはこの点についての考察を行なっていない。そのため、ノウハウの抽象化は人手に頼らざるを得ないのが現状であるが、我々は、Activity の失敗事例をモデル化し、一般的ノウハウを抽象化することを検討している。今回のケーススタディにおいて、Activity の成功事例よりも失敗事例の方が一般的なノウハウの抽出に有効であるケースが多く観察されたことは留意すべきである。

また、ノウハウの抽象化のために必要な事項として、分析時における分析者の意志決定プロセスを形式的に記述することが挙げられる。これらの機能を計算機上で実現するために、我々は、本プロトタイプモデルで明らかにされた OOA プロセスを形式的

仕様記述言語で記述する試みと、意図に関する形式的な記述法、および抽象化された仕様を類推的に適応するためのオブジェクト間のインターフェースに関するメカニズム [Kishimoto92] についての研究を併せて行なっている。

また、どこから分析を始めるかといったような、個々の分析者が持つノウハウを生かすことも重要な課題である。何らかの理由によりプロセスの実行順序を変更したい場合 (特定のクラスについてすべての分析を行なったのちに、それをベースに他のクラスを作成する場合など) については、本モデルではデータ間の関連を詳細に洗い出しているために、ある程度は柔軟に対応することが可能である。

問題領域の専門家のノウハウについては、分析対象となる問題の記述において考慮すべき事項であるため、本モデルでは特に考慮していない。ただし、入力仕様の記述形態が、イベント中心であるか、データ中心であるかにより、それに対応するための practical tip についての考察が必要であると考える。これはまた、主に自然言語で書かれた要求仕様から、どのようにして分析に有効な情報を抽出するかという問題であり [Ohnishi91] 、オブジェクト指向においては特に、クラスの候補をどのように抽出するかといった問題になる [Wirfs90] 。

• OOA プロセスの並行性の支援

OOA プロセスは通常、分析者や問題領域の専門家など、複数の人間が協調して行なう作業プロセスである。また、個々の最終成果物の間に存在する密接な相互依存性ゆえに、OOA プロセスは本質的に並行プロセスであるということができる。今回のケーススタディから作成したプロトタイプモデルにおけるプロセスでは、逐次的なコントロールフローを基本に、積極的かつ明確なバックトラックフローを定義することにより、プロセスの並行性を支援することとした。これにより、OOA プロセスの並行性を明らかにすることができた。今後は、計算機による並行性の支援を実現するために、TMS(Trues Maintenance System) や仮説推論のメカニズムを採り入れるべく検討を行なう予定である。

7 まとめ

OOA プロセスに関して、方法論を限定して詳細に調査し、ケーススタディから得られた細部のプロセスと分析ノウハウをもとに、プロトタイプモデルを作成した。その上で、プロトタイプモデルを異なる複数の問題に適用することにより、モデルの妥当性を評価した。

本モデルにおいて、分析者が行なうべきことは、practical tip に基づいた、Activity における意志決定に集約されている。すなわち、それ以外の部分については計算機による自動化が可能であり、我々はそのための研究として、モデル自体の形式的な記述およびオブジェクト間の通信方式についての検討を行なっている。そのための要素技術として、TMS や仮説推論を考えている。また、分析者の行動をモデル化し、それを形式的に記述することで人間の作業の記録を採り、それを新たな分析ノウハウとして抽象化するメカニズムについての研究を継続して予定である。

これらの研究を行なう上で、また、モデルの妥当性および汎用性を向上させるために、現状の OOA モデルを計算機上に実現することが急務であると考えている。特に、OOA の場合は中間生成物を含むデータが膨大であり、それらの間の関係も複雑であるため、手作業で中～大規模の問題でのケーススタディを行なうことは事実上不可能である。そのための実験システムについて、本年中に実現する方向で検討中である。

謝辞

本研究は、次世代産業基盤技術研究開発「新ソフトウェア構造化モデルの研究開発」の一環として情報処理振興事業協会が新エネルギー・産業技術開発機構から委託をうけて実施したものである。

参考文献

[Booch91] Booch, G. : *Object-Oriented Design: With Applications*, The Benjamin / Cummings Publishing Company, 1991.

[Coad91a] Coad, P., Yourdon, E. : *Object-Oriented Analysis: Second Edition*, Prentice-Hall, 1991.

[Coad91b] Coad, P., Yourdon, E. : *Object-Oriented Design*, Yourdon Press, 1991.

[Davis90] Davis, A. M. : *Software Requirements: Analysis & Specification*, Prentice-Hall, 1990.

[Duke91] Duke, R., et al. : *The Object-Z Specification Language Version 1*, SVRC Technical Report No. 91-1, 34-45, The Univ. of Queensland, 1991.

[Henderson91] Henderson-Sellers, B. : *A Book of Object-Oriented Knowledge*, Prentice-Hall, 1991.

[Kishimoto92] Kishimoto, Y., Kotaka, N., Honiden, S. : *Reconciliation Model of Object Interfaces*, Preprints Work Gr. for Softw. Eng., IPSJ, Vol. 92-SE-83, 1992. (In Japanese)

[Kotaka91a] Kotaka, N., Kishimoto, Y., Honiden, S. : *Formalization of Object-Oriented Analysis*, Proc. The 43rd Annual Convention IPS Japan, 3K-6, pp. 5:437-438, 1991. (In Japanese)

[Kotaka91b] Kotaka, N., Kishimoto, Y., Honiden, S. : *Specification Process Modeling in OOA*, TOOLS 6 (Proc. of TOOLS PACIFIC '91), pp. 67-81, Prentice-Hall, 1991.

[Kotaka92] Kotaka, N., Kishimoto, Y., Honiden, S. : *A Study of Process Modeling in OOA*, Preprints The 4th Software Process Workshop, Japan Society for Softw. Sci. and Technology, 1992.

[Matsuura92] Matsuura, S., Honiden, S. : *Abstract on Formal Specification Language*, Preprints Work Gr. for Softw. Eng., IPSJ, Vol. 92-SE-83, 1992. (In Japanese)

[Mayer88] Meyer, B. : *Object-Oriented Software Construction*, Prentice-Hall, 1988.

[Ohnishi91] Ohnishi, A. : *Software Preliminary Design Method for Japanese Requirements Specification based on the Requirements Frame Model*, Proc. The 8th Annual Conference of Japan Society for Softw. Sci. and Technology, C6-6, pp. 533-536, 1991. (In Japanese)

[Rumbaugh91] Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., Lorensen, W. : *Object-Oriented Modeling and Design*, Prentice-Hall, 1991.

[Shlaer88] Shlaer, S., Mellor, S. J. : *Object-Oriented System Analysis: Modeling the World in Data*, Prentice-Hall, 1988.

[Shlaer91] Shlaer, S., Mellor, S. J. : *Object Lifecycles: Modeling the World in States*, Yourdon Press, 1991.

[Wirfs90] Wirfs-Brock, R., Wilkerson, B., Wiener, L. : *Designing Object-Oriented Software*, Prentice-Hall, 1990.