ソフトウェア工学 94－8
(1993. 9. 21)

オブジェクト指向に基づく 要求記述からの形式的仕様の導出手法

滝沢陽三　上田賀一

茨城大学

〒316 茨城県日立市中成沢町4-12-1

あらまし　コンピュータによるソフトウェア開発の支援環境が整備されつつあり、要求定義のための優れた表記法が考え出された。しかしそれらの表記法を用いた定義はまだ開発者の手によるものが主流であり、要求者の要求が十分に反映されずに開発が進むことが多い。このため自然言語による要求記述から初期の形式的仕様を生成する手法が必要になる。特に開発パラダイムがオブジェクト指向である場合、実世界の対象を記述から直接抽出できれば非常に有利である。本研究は、日本語で書かれた非形式的な要求記述からオブジェクトモデル化技法(OMT)で使用されるオブジェクトモデルの情報を直接抽出する手法およびその支援システムを提案する。

和文キーワード　要求定義、オブジェクトモデル、形式的仕様、自然言語

Derivation of Formal Specifications based on Object-Oriented Approach from Requirement Description

Yozo TAKIZAWA　Yoshikazu UEDA

Ibaraki University

4-12-1, Nakanarusawa, Hitachi, Ibaraki, 316 JAPAN

Abstract

Supporting environments for the software development by computer is put in good condition, and various representations for the requirement definition are also provided. These representations, however, are usually used by developers, so customers' requirements are not fully reflected in the development. This makes us need the method for the derivation of formal specification from the requirements descirbed in natural languages. Especially, it is favorable to us that we can extract the entities in the real world from descriptions in the object-oriented approach. We propose a method and a supporting system for the derivation of the object model for the OMT from the requirement descriptions written in Japanese.

英文 key words　requirement definition, object model, formal specification, natural language

1. はじめに

近年ソフトウェア開発がますます大規模になり、コンピュータによる支援が各開発段階で導入されつつある。効率的で分業可能な手法が提案され、使われる記法も分かりやすく管理しやすいものが考え出された。

初期段階で必要とされる要求定義は従来のものより複雑になり、正確さも要求される。しかしそれにもかかわらず、多くはまだ開発者の手による定義が主流であり、時間もかかり間違いも起こりやすい。更に、これらの手法に要求者の要求を十分に反映させようとするならば、実際の開発に全く関わらないはずの要求者も各種記法を覚え、ある程度使いこなせなければならない。この溝を埋めるため、どんな人でも使いこなせる自然言語によって表現された要求記述から、各種記法で表現された形式的仕様を生成する手法が必要になってくる。

要求記述から初期構造の要求モデル図を得る手法・システムの必要性は、特にそれがオブジェクト指向パラダイム[1][2]に基づく場合、より重要になってくる[3]。「実世界の現象をそのままの形で表現する」ゆえ、人が普段使う自然言語による表現からは数多くの実世界の現象を直接取り出せるはずであり、実現されればオブジェクト指向開発法にとって非常に有利だからである。

本研究の目的は、日本語で書かれた要求記述からオブジェクト指向開発法の一つであるオブジェクトモデル化技法(OMT)[1]の3つのモデル図: オブジェクトモデル図・動的モデル図・機能モデル図を導く手法・システムを開発することである。機能モデルについては、データフロー図の生成として英語・日本語ともに各方面で研究が進んでいるが[13][14]、オブジェクト指向固有の概念を表す図の生成についてはほとんど提案されていない。よって、特にオブジェクトモデルの導出を中心に研究を進めることにした。要求記述は日本語で書かれたものとしたが、日本語を用いるシステムの開発を通して仕様変換システムの構築方針が得られれば、他の言語への変更も比較的容易であると思われる。

2. 自然言語処理とOMT記法

前章で述べたように、本研究は自然言語と表記法の間の溝を埋めることを目的としている。以下に、本研究で取り入れた自然言語処理の技法とOMTで使われる表記法についての概略を述べる。

2.1 自然言語処理

自然言語処理には大きく分けて、言語解析と言語生成がある[10]。このうち重要なのは言語解析である。様々な表現から情報を認識する必要性の方が大きいからである。言語解析には個々の文の解析と文脈の解析があるが、記述対象が単純ならば文脈の解析はほとんど必要としない。

文の解析で行われる処理にも大きく分けて2つ、構文解析と意味解析がある。構文解析とは文の構造の解析のことであり、その役割は動詞の主語や目的語が何か、修飾語が何を修飾しているか、などを決定することである。この決定には主に句構造言語を通して解析木を付与することで行われる(図1)。構文解析は他に、省略された単語を文構造から認識して復元したり、多様な入力構造を少数の構造に変換したりする文構造の調整も行うことがある。構文解析はこのように、システムが意味を理解する前段階として大きな役割を果たす。意味解析では、文の表す意味が真かどうかを決定する。この決定には主に、推論規則を与える記号論理が用いられる。

```
<文>          ::= <中心部>
<中心部>      ::= <断定文>
<断定文>      ::= <修飾部> <主部> <修飾部> <述部> <修飾部>
                  <目的部> <修飾部>
<修飾部>      ::= 副詞 | <前置詞句> | ε
<前置詞句>    ::= 前置詞 <名詞列>
<主部>        ::= <名詞列>
<名詞列>      ::= <LNR> | <what名詞句>
<LNR>         ::= <LN> 名詞 <RN>
<LN>          ::= <冠詞位置> <形容詞位置>
<冠詞位置>    ::= 冠詞 | ε
<形容詞位置>  ::= 形容詞 | ε
<RN>          ::= <前置詞句> | ε
<what名詞句>  ::= what <主部> <修飾部> <述部> <修飾部>
<述部>        ::= <LTVR>
<LTVR>        ::= <LV> 有時制動詞 <RV>
<LV>          ::= 副詞 | ε
<RV>          ::= 副詞 | <前置詞句> | ε
<目的部>      ::= <名詞列> | <to目的語> | ε
<to目的語>    ::= to <LVR> <修飾部> <目的部> <修飾部>
<LVR>         ::= <LV> 無時制動詞 <RV>
```

図1　英語の単純な構文例 (文献[10]より)

このような研究は多くが英語を対象として行われてきたが、日本語処理は、具体的な文法や単語の並びなど言語固有の処理を除いて、英語処理と基本的には同じ考え方で行われる[9]。構文解析を行って文構造を解析し、その後意味解析を行う。ただ日本語の場合、自動翻訳を中心に研究が行われており、実際の構文・意味解析では文脈把握などのため、より複雑なデータ処理が行われている。

本研究ではこれらの研究のうち主に構文解析の技法を取り入れた。本研究で開発しようとしているシステムは自然言語処理の分野から見ると極めて小規模なものであり、またこのシステムでの意味解析は自然言語処理のそれとは処理内容が違い、文章そのものが表現

する意味の真偽は問わないからである。

2.2 OMT記法

ここでは、OMTで使われる記法および概念を、以下の説明に必要最小限なものにとどめて解説する。詳細は文献[1]等を参照されたい。ただし、今回の研究ではオブジェクトモデルを対象としているため、オブジェクトモデルについては少し細かく説明する。

(1) オブジェクトモデル

オブジェクトモデル図はシステムの静的な構造を表現するための図であり、OMTのモデル図の中では最も重要なものである。表現される情報としては、クラスの認識、クラスの属性・操作、クラス間の継承・集約・関連関係などである(図2)。

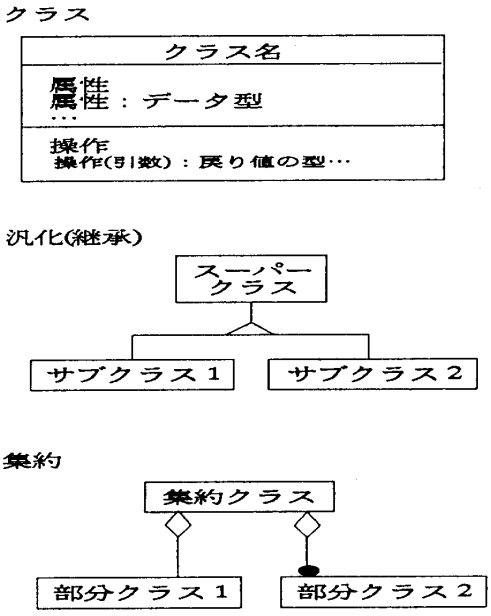

図2　オブジェクトモデル図の例

○クラス

クラスとは、実体として個々に区別されたオブジェクトを、同じ属性・同じ操作をもつものとしてグループ化したものである。このクラスを基本として、継承などの概念が表現される。クラスには、固有の特徴や振舞いを表す属性・操作が記述される。

○継承

継承とは、階層関係に基づいて上記の属性・操作をクラス間で共有することである。通常はスーパークラスとサブクラスがあり、スーパークラスでオブジェクトとしての基本的な特徴を記述し、サブクラスはそれを受け継いで更に独自の特徴を追加する。このようにして、オブジェクトの機能強化などを比較的簡単に行うことができる。

○集約

集約は、あるオブジェクトが他のオブジェクトの一部であることを表す。一部として機能するオブジェクトは普通部品のように扱われる。

オブジェクトモデルはオブジェクト指向の概念を直接表す。他のモデルが従来からの手法に使われ、図の自動生成などの研究がある程度行われているのに対し、オブジェクト指向の概念を表す図の自動生成の研究はあまり行われていない。今回考察したシステムがオブジェクトモデルの情報を導出するのも、オブジェクト指向の概念を表す図の導出手法を他の表記のものよりも優先して与えるためである。

(2) 動的モデル

いわゆる状態遷移図で表現される。時間的な変化が記述され、システムの制御の実現などに用いられる。

従来の開発手法では、例えばモジュール間の依存関係の評価などにもよく使われる。ただ、このようなモデルを記述するためには時間的な「シナリオ」が別に必要であり、要求定義の段階やオブジェクト指向パラダイムでは捉えにくい側面をもっている。

(3) 機能モデル

表記はデータフロー図と全く同じであり、入力から出力に至る情報の流れを表す。

データの流れの表記は手続き型言語とよく合うため、これまで非常によく使われてきた表記法の一つである。オブジェクト指向言語ではオブジェクト単位で物事を考えるので、データフロー図は主にオブジェクト内部のデータ処理の流れを表すのに使われる。

3. システム構成

前述したように、今回開発したシステムはオブジェクトモデル図のみを導出する。システムは次のような手順で処理を行う(図4, 図5)。

(1) 単文化

非形式的な要求記述を自然言語処理によって解析する。単語を認識し、省略された言葉や間違いなどを補正したり、複文を単文にしたりする。更にここで、後の処理で不必要な修飾語等を削除する。この段階では名詞句の形による修飾部はそのまま残るが、次の段階

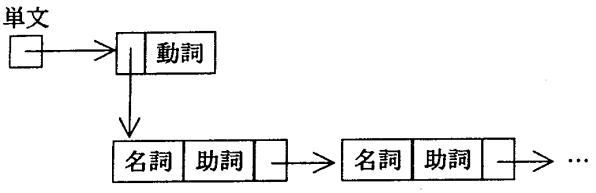

図3　単文のデータ構造

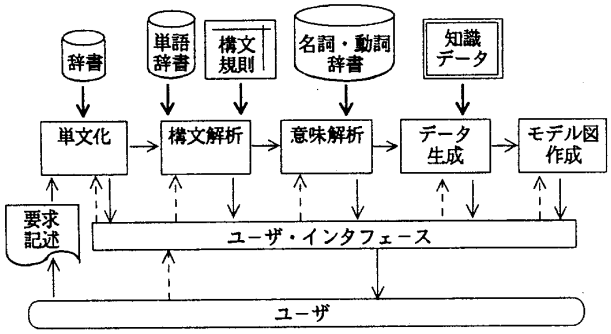

図4　導出システムの概観

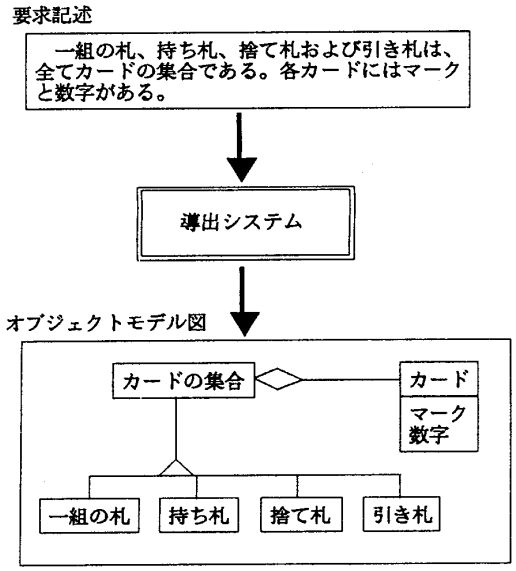

図5　導出処理の例

の意味解析で辞書を用いて修飾部であることを判断し、これを削除する。

出力される記述は最終的に、構文解析で受け入れることができる形にされる。具体的には、構文解析で参照される構文規則(後述)に従う形である。

(2) 構文解析

構文が後の意味解析で処理できる形かどうかチェックするとともに、単文を内部データ化する。あらかじめ用意した構文規則に基づき、単語の品詞を辞書で調べながら単文の構造を解析し、受け入れ可能ならば単文ごとにシステム内部のデータ構造に変換する(図3)。

(3) 意味解析

単語の意味を調べて日本語として正しいかどうかを確かめるとともに、表す概念を決定する。この段階は「単文の整形」と「概念の決定」に分かれ、単文をより処理しやすい形にしてから辞書を用いて概念を決定する。ここで使われる辞書には、名詞および動詞のもつ概念の情報が記述されている。

「単文の整形」は更に4つの段階に分かれる。

i) 動詞情報の読み込み

必要な動詞情報を読み込むとともに、ここでは特殊な動詞に対する例外処理も行う。

ii) 修飾部の削除

単文化や構造解析で削除できなかった修飾部を、動詞の情報を用いて削除する。

iii) 助詞による名詞の概念の決定

動詞による制約ではなく、助詞との関連によって導出できる概念を決定する。

iv) 複数の対象による単文の分割

1つの単文内に2つ以上のオブジェクト間関係がある場合、1つの単文で1つのオブジェクト間関係を表すように、単文を分割・整形する。

「概念の決定」は2つの段階に分かれる。

i) 主語と目的語の決定

単文の構造から主語・目的語を決定する。

ii) 概念の決定

名詞・助詞・動詞の概念の組み合わせが一致するものを辞書の情報から選び、単文全体の表す概念および名詞の概念を特定する。

意味解析によって得られる情報は単文ごとに出力される。1つの単文につき単文の表す概念、関係をもつオブジェクトなどの名前、関連名や操作名が決定される。

(4) データ生成

意味解析の結果をOMTのオブジェクトモデルを直接表すデータ(図6)にし、意味内容の整合性のチェックを行う。意味内容の整合性チェックにはあらかじめ用意された知識データが用いられる。知識データはこの段階で生成される情報と同じ書式をもつ、継承関係の上位概念などを表現したものである。

(5) モデル図作成

オブジェクトやオブジェクト関係を図にし、必要な

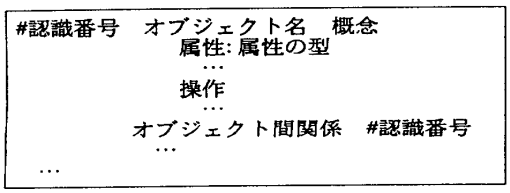

図6　出力情報の書式

レイアウトを施して出力する。

(6) インタフェースと辞書

ユーザ・インタフェースは、途中で生成される情報などを表示したり、各段階の個々の処理を促したり、また全ての段階を自動的に処理させる働きをもつ。更に、各段階で用いる辞書の指定などもこれで行う。

辞書には単文化で用いられる純粋な自然言語処理のためのものと、意味解析で用いられる各種オブジェクト指向概念と単語との対応関係を蓄積したものの2つがある。データ生成で用いられる知識データには主に「常識的な」情報が含まれ、導出されたデータと概念などの点で整合させる。

構文解析に使われる単語辞書は単に品詞の決定を行うためのものであり、意味解析で用いる辞書を流用することもできる。

このシステムのうち最初の単文化、および最後のモデル図作成は、それぞれ純粋な日本語処理・レイアウト処理であるので本研究では設計・実現は行わない。ただし、構文解析を行うのに必要なデータ構造、モデル図を描くのに必要な情報およびその書式は明確に定義した。

4. 導出手法

今回開発した手法は、オブジェクトモデル導出に関する部分のプロトタイプである。以下に、オブジェクトモデル図を導出する手法の詳細を説明する。

4.1 入力される日本語記述

「単文化」では、後に続く段階において処理がしやすいように非形式的な記述を整形する。基本的には、名詞・動詞・助詞のみからなる単文の形式にする。次の構文規則は「構文解析」で参照される構文をBNF記法で表現したものであり、単文化ではこれを満たすように記述を変形しなければならない。

```
<記述>     ::= ε | <記述> <単文> 。
<単文>     ::= <名詞句列> 動詞
<名詞句列> ::= <名詞句> | <名詞句列> <名詞句>
<名詞句>   ::= 名詞 助詞
```

このように、参照される規則は極めて単純なものである。このようにした理由は、オブジェクトモデルの表す概念は名詞と動詞に特にはっきりと表現されることが多く、修飾語などが表す意味はオブジェクトそのものの意味を限定することが少ないからである。

単文化によって、最初に入力される非形式的な要求記述は極めて形式的な形になる。表面的には名詞句と動詞のみの並びとなり、あいまいさは皆無である。手法において最初の段階にこの単文化を置いたのは、自然言語であっても最初から後者のような形式性のある記述を得ることはほとんど不可能と考えたからである。

4.2 記述に現れるオブジェクト指向の概念

「意味解析」で行う処理には大きく分けて、特定の助詞のもつ働きに従って単文を更に単純な形に整形することと、その単文の構造と単語の概念を専用の辞書と照合して具体的なオブジェクト指向の概念を決定することがある。

(1) 単文の整形

単文の整形では主に、不必要な名詞句の削除や並列関係の単語ごとに単文を分割したりするが、助詞そのものによるオブジェクト指向の概念の決定や、特殊な動詞のための例外処理を行う。

まず、修飾部の削除を行う。前述したように、名詞句の形による修飾部は単文化では削除されないので、単文に使われる動詞を動詞辞書で調べ、その動詞に使うことのできない助詞が含まれる名詞句は削除する。ただし、助詞「は」など単文の中での役割が明白であるものは、辞書に登録されていなくとも削除しない。

次に、助詞による名詞の概念の決定を行う。助詞「の」の後に続く名詞は、その「の」の前に使われている名詞をクラスとした場合の、集約されたクラスの一つ(クラス-「の」-クラス)もしくは属性(クラス-「の」-属性)とみなすことができる。名詞の種類が分かれば集約関係またはクラス-属性関係の情報として出力できる。

最後に、複数の対象による単文の分割を行う。助詞「と」は名詞同士の並列関係を表すので、各名詞は同じ概念の中で使われるものと解釈できる。したがって、最後の名詞の後に続く助詞を各名詞の後に続く助詞とみなしてその名詞の数だけ単文を作り出すことができる。ただし、助詞「と」は1つの単文の中に複数存在するのが普通であるので、作り出される単文は場合によっては非常に多くなる。更に、上記の助詞「の」

と共に使用されている場合、同じクラスの複数の属性を表すならば(例えば「ウィンドウの幅と高さは」)、最初に並列関係を判断してから名詞の概念の決定を行わなければならない。

このようにして、単文の整形によって得られる単文は1つか2つの名詞句と動詞のみ("名詞-「は」-動詞"もしくは"名詞-「は」-名詞-助詞-動詞"。名詞句が2つの場合、その順番は逆になることもある)という極めて単純な形になる。この極めて単純な単文から他のオブジェクト指向の概念を決定する。

(2) 概念の決定

オブジェクト指向の概念の決定は、動詞と名詞に関する辞書情報を取り出して、いくつかある概念の組み合わせから適切なものを選ぶことによって行う。具体的には、主語や目的語を表す名詞の概念と、動詞が取り得る文型を照合して行う。

ただし、もし単文が1つの名詞句と動詞のみからなり、名詞句の助詞が「は」(すなわち名詞は主語である)であるなら、動詞は名詞をクラスとしたときの「操作」であると判断する。例えば「ウィンドウは縮小する」という文は、「ウィンドウ」をクラスとし、「縮小する」はクラス「ウィンドウ」の操作であるとする。

文型の照合については、動詞ごとに様々な判断をする。例えば集約を表す動詞としては「成る」が代表的だが、この動詞が単文に使われる場合、まず集約するクラスと集約されるクラスが存在しなければならない。前者は普通助詞「は」によって主語を表す名詞として、後者は目的語を表す名詞として単文中に現れる。更に、目的語を表す名詞の直後に続いて名詞句を構成する助詞は、集約を表すために「から」もしくは「より」でなければならない。

重要なのは、オブジェクト指向の概念とそれを表す動詞が必ずしも1対1ではないということである。例えば上記の集約関係を表す動詞には他に「構成する」や「使用する」などがある。更に「～の一部である」という表現もある。この場合は助詞「の」の処理と連携をとって集約関係を導くのだが、「～である」という形は継承関係・属性の型・インスタンス生成を表すこともある。したがって動詞辞書には動詞1つにつきいくつもの概念や名詞の種類の組み合わせを登録できるようにしておかなければならない。

動詞辞書の情報には、使うことのできる名詞の種類および助詞が登録されている。名詞が属性を表すのかクラスを表すのかといった名詞の概念は名詞辞書に登録しておく。ただしここでの名詞の概念とはクラス・インスタンス・属性のどれかのことであり、決して継承関係による上位概念を意味するものではない。継承関係のチェックは、前述の知識データによってモデル図の情報を出力する直前に行う。

4.3 出力される情報の構造

意味解析によって得られる情報には、表す概念、概念によって結び付けられるクラスや属性などがある。「データ生成」の段階ではこれらの情報を、オブジェクトモデル図を直接表す書式にして出力する。図はオブジェクト単位で描かれるので、認識したオブジェクトにはそれぞれ番号を与え、オブジェクトごとに属性やその型、操作、オブジェクト間関係をまとめて出力する。

「モデル図作成」で出力されるモデル図は、基本的にはクラスの内部構造も表示した、OMTの記法に従ったものである。しかしオブジェクトの数が多くなるなどモデル図が大きくなる場合は、内部構造は省略したり、継承階層を一部省略したりすることになる。この場合でも、導出した内部構造や階層図は常に何らかの形で参照できるようにしておく。

4.4 辞書

意味解析で使われる名詞辞書・動詞辞書は、表す概念を決定する上でもっとも重要な辞書である。上記で示したように、名詞辞書は各名詞の概念を登録しておくもの、動詞辞書は動詞が単文中で使われる場合の名詞や助詞との関係を、概念とともに登録しておくものである。要求記述に現れる名詞の全てが概念とともに名詞辞書に登録されていなければならず、同じように動詞もまた全て概念とともに動詞辞書に登録されていなければならない(図7)。登録されていなければ解析不可能となる。

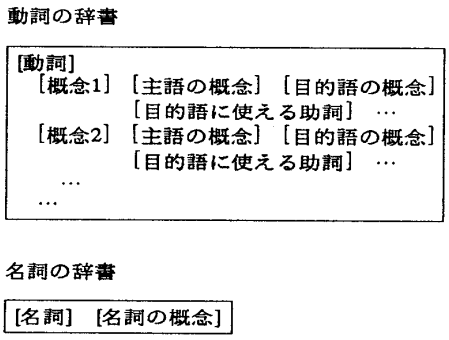

図7 名詞・動詞辞書のデータ構造

もちろん、「どのような記述が解析不可能か」は明確にされるので、手法がエキスパートシステムの中で実現される場合、何らかの形でこれらの概念を登録していくことで辞書情報の蓄積を図ることができる。

5. システムの試作

今回実現したシステムは、プロトタイプの手法そのものの実現可能性を明確にすることを目的としているため、試作システムは「構文解析・意味解析」と「データ生成」の一部分のみで構成される。従って最初に入力される要求記述はすでに「単文化」処理が終っているものでなければならず、また処理しやすいように単語ごとに区切られていなければならない。更に、最終的に出力されるデータもモデル図を描くために最小限必要な情報のみである(図8)。更に、整合性チェックも今回は行わない。

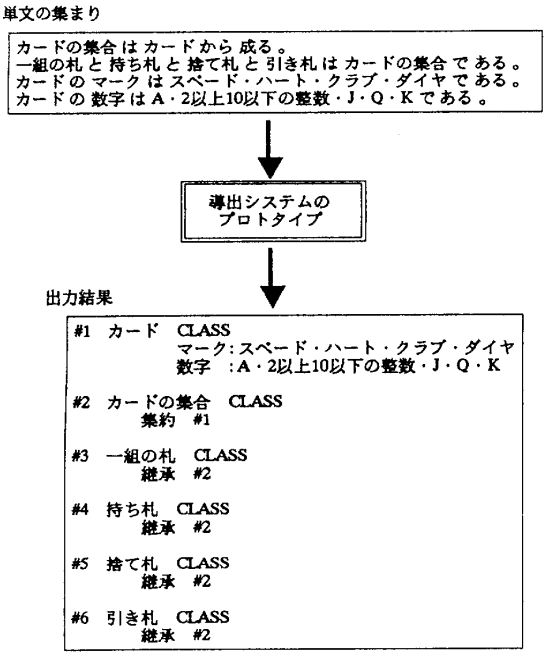

図8　プロトタイプでの入出力例

今回は実現がしやすいということで、UNIX上でコンパイラを作成するのに使用される yacc を構文解析に用いている。システム構成では構文規則が「構文解析」部分と区別されているが、yacc の性質上、構文規則はプログラム内に組み込まれている。同じように yacc の機能を用いて、構文解析そのものと内部データ化も一緒に行っている。

辞書も、テストデータで必要なものにとどめた。プログラムを単純にするため、単語辞書と名詞・動詞辞書は別に用意した。更に、今回は意味内容の整合性チェックを行わないので、データ生成で使われる知識データは作成しなかった。

実現はUNIX上のC言語で行い、単文の入力や生成データの出力は標準入出力で行われる。辞書ファイルの情報の読み込みはプログラムが直接ファイルを操作して行う。

6. まとめ

プロトタイプは、要求記述からモデル図を導出する手順の核の部分を明確にしたという点では成果が認められるが、開発した手法・システムに対する問題点や課題は非常に多い。

(1) 手法について

更に柔軟かつ充実した処理を行うためには、入出力される情報をより明確にしなければならない。どこまでを純粋な自然言語処理で処理しどこから独自に解析を行わなければならないか、また最終的に出力される情報はどこまでモデル図を詳細に表していなければならないか、などである。これにより、自然言語処理の分野での研究成果をより多く取り入れる必要が生まれてくるかもしれない。

各種概念の決定はそのほとんどが辞書に蓄積されている情報によって行われる。他にもまだカバーしなければならないオブジェクト指向の概念もあるが、辞書をベースとしたこの方法で本当に全ての概念を表現しきれるのか、他にもっと規則的で多様な処理方法があるのではないかなど、手法そのものに対する疑問にもテストや実例・改良を繰り返して解決していかなければならない。

(2) システムについて

「意味解析」で用いられる名詞・動詞辞書のデータを具体的にはどのようにして得るべきかはまだ明らかになっていない。オブジェクト指向の概念を日本語記述から抽出して辞書に登録する手法も提案されているが[8]、抽出される情報や表現形式、具体的な登録システムはまだ不十分である。対話形式で得るシステムを考えるにしても、そのために必要な処理は何か、具体的な入力・確認方法はどうするのか、などはまだほとんど考えられていない。ただ、手法は原則としてはあくまで自然言語とモデル図の対応で行うものであるから、より自然な言語表現を受け付けることを確認するため、辞書登録でも自然言語処理、特に意味情報からの自然言語生成技術を参考にする必要があるだろう。

動詞辞書等のフォーマットについては再考の余地が残っている。極めて多数の概念を表すなどの「特別扱

いしなければならない動詞」がいくつかあるが、このような動詞の数は比較的少ない。これらの情報をプログラム中に組み込むことができれば、処理そのものが効率よくなるばかりでなく、動詞辞書には概念の数が少ない動詞のみを登録すればいいことになるので、辞書のフォーマットは現在のものよりもかなり単純にすることができるだろう。

今回実現したシステムでは情報の入出力にUNIXの標準入出力を用いている。処理データが膨大になったり、将来システムにモデル図の描画システムを組み込むならば、入出力ファイルの操作や文書管理を行うのに便利な、GUIを利用した操作環境を提供するべきだろう。少なくとも、入出力や辞書ファイルの管理を簡単に行えるようにするべきである。

「データ生成」で必要とされる知識データは「常識」として登録されたものである。これについても知識の獲得方法など名詞・動詞辞書と同じ問題がある。更にこの場合、システムがより正確な情報を生成できるために、何が「常識」でそれをどのような概念で表現するかをある程度明らかにしておかなければならない。

システムの実用性や将来性も検討しなければならない。今回は手法の考察とその実現のみに重点をおいたが、考察した導出システムが実際の開発作業でどれだけの効果をもたらすかはまだ確認していない。もちろん実際の作業で用いるためには更に充実した手法・システムを実現しなければならないが、核となる手法に大きな変更は必要ないので、ある程度の確認はできるだろう。

参考文献

[1] J. ランボー 他著, 羽生田栄一 監訳: オブジェクト指向方法論OMT, トッパン(1992).

[2] I.Jacobson, et al.: Object-Oriented Software Engineering, ACM Press(1992).

[3] S.Honiden, et al.: Formalizing Specification Modeling in OOA, IEEE Software, pp.54-66(1993, Jan.).

[4] 野木兼六: 要求定義技術の最近の動向, 情報処理, Vol.27, No.1, pp.21-30(1986).

[5] 松本吉弘: ソフトウェアに対する要求の形成, 情報処理, Vol.28, No.7, 853-861(1987).

[6] 伊藤 潔, 本位田真一: プロトタイピング支援ツール, 情報処理, Vol.30, No.4, pp.387-395(1989).

[7] 山田 淳, 他: 用語知識再利用による要求分析支援手法(I)(II), 情報処理学会第42回全国大会講演論文集(5), pp.177-180(1991, Mar.).

[8] 山田 淳, 他: 要求分析支援ツール JOKER'91, 情報処理学会第44回全国大会講演論文集(5), pp.555-334 (1992, Mar.).

[9] 辻井潤一, 上原邦昭: ソフトウェア工学と自然言語処理, 情報処理, Vol.28, No.7, pp.913-921(1987).

[10] 淵 一博 監修: 自然言語の基礎理論, 知識情報処理シリーズ第4巻, 共立出版(1986).

[11] R.グリシュマン, 山梨正明・田野村忠温共訳: 計算言語学・コンピュータの自然言語理解, サイエンス社 (1989).

[12] 佐伯元司, 蓬莱尚幸, 榎本 肇: 自然言語からモジュール構造を得る手法について, 情報処理学会論文誌, Vol.30, No.11, pp.1479-1493(1989).

[13] 大西 淳: ソフトウェア要求定義のためのコミュニケーションモデル, 情報処理学会論文誌, Vol.33, No.8, pp.1064-1071(1992).

[14] 黒木宏明, 磯田定宏: 統合化CASEシステムSoftDAの機能とその評価, 情処研報83-3(1992, Feb.).

[15] 三浦 和則, 他: 日本語文からの要求分析図式作成支援機能の提案, 情報処理学会第43回全国大会講演論文集(5), pp.387-388(1992, Mar.).

[16] J.J.-P.Tsai, J.C.Ridge: Intelligent Support for Specifications Transformation, IEEE Software, pp.28-35(1988, Nov.).

[17] 滝沢陽三, 小坂和稔, 上田賀一: 要求記述からオブジェクト構造による形式的仕様への導出について, 情報処理学会第46回全国大会講演論文集(5)(1993, Mar.).