自然言語記述による要求仕様導出支援システムの提案

滝沢陽三† 上田賀一†

本論文では，自然言語記述による要求仕様導出支援システムの考察および設計/試作について述べている．この支援システムの目的は要求者が要求を自然言語で仕様化できることである．要求者が要求を自然言語で定義し確認できれば，要求者の意図が開発者に伝えやすくなる．しかし，現在の自然言語処理技術ではきわめて制限された処理しか行えない．本論文では，記述解析において構文解析による処理と単語による辞書検索を一緒に用いることで，記述から多くの情報が導出できることを示す．この考えに基づいて試作した日本語による支援システム ARDES/J について解説し，システム全体についての検討事項を報告する．

Supporting System for Acquiring Requirement and Deriving Specification from Descriptions Using Natural Language

Yozo Takizawa† and Yoshikazu Ueda†

In this paper, we propose the supporting system for deriving requirement specification from descriptions using a natural language. The goal of the supporting system is that a requirer can specify some requirements using a natural language. The requirer will convey the idea more easily if he/she can define and confirm requirements by natural language. But current processing techniques of natural language can process only the restricted sentences. We describe that many informations about the specification are derived using both the syntax analysis and the search for words in dictionaries. Under this consideration, we show the implemention of ARDES/J, which is the supporting system by Japanese, as a prototype, then we report some matters about a whole system.

1. はじめに

本研究では，要求者が要求を非形式な自然言語によって定義し，それを開発にそのまま反映できるようにすることを目的とした支援システムの開発を試みた．そして手法として，要求者が書く記述は非形式なものとしつつ，その記述をコンピュータによる自然言語処理により形式化し，処理結果を要求者が確認し，また記述を書き直すことで要求を定義する手順を考えた．

ソフトウェア開発において，要求者がその要求を正確に開発に反映させるためには，要求者自身がソフトウェアの初期仕様を詳細化できなければならない．しかし，要求者が設計/実装にそのまま利用できる形式で要求を表現し詳細化できれば要求が開発により正確に反映されることが期待できるが，開発者ではない要求者が特定の開発のためだけに仕様化手法を覚えることは大きな困難がともなう．

† 茨城大学工学部情報工学科
Department of Computer and Information Sciences, Faculty of Engineering, Ibaraki University

自然言語による記述は，要求者がどのような分野の専門家であっても用いることができる唯一の仕様化手法であると考えられる．自然言語記述がソフトウェア開発における初期仕様の記述言語であれば，要求者による詳細化は比較的簡単に行うことができると思われる．

これを踏まえたものとして，最初に書かれる要求仕様に自然言語を用いる手法もあるが，この自然言語で仕様を記述するにはやはりある程度の訓練が必要となる．さらに，要求者が非形式的な記述で詳細化しても，その記述が開発者の手作業によって機械的に仕様（形式化された自然言語を含む）に変換されるならば，変換過程で要求者の要求が正確に伝わらない場合がある．それは記述内容が複雑であるほど起こる可能性は高い．

一方，コンピュータによる完全な自然言語処理は不可能であり，しかも要求者による記述に要求が過不足なく反映されるとは限らない．人間にとっては十分な表現であってもコンピュータではその意図を把握することは表現の不規則性から不可能であり，また望ましくない[1]．本研究ではこれを解決するため，知識情報

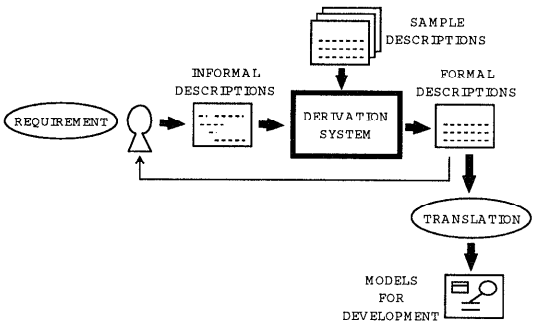


図1 支援システムの流れ
Fig. 1 Overview of supprting system.

の検索による記述構造解析および情報付加の処理も記述変換時に行うことにし，その結果を要求者が調べて再度記述を書き直すことによって記述を形式化することとした（図1）．

本論文では，この記述変換の手順およびそれを行う支援システムの考察と設計，そしてシステム実現にあたっての問題点を調べるために行った試験的な実装について示す．

2. 自然言語記述からの要求導出を支援する手法 ASRED

前章の目的を達成するため考察した手順を本研究ではASRED[2]（the Acquirement Supporting method of REquirement from Descriptions）と呼び，手法として体系付けた．ASREDの大まかな流れは図2のとおりである．

本研究の目的は「要求者による要求定義の実現」である．開発段階で形式的仕様を用いる必要がある以上，要求者の定義したものは何らかの形でこの形式的仕様に変換されなければならない．そして，これら形式的仕様は開発手法あるいは開発現場ごとに様々な形式が存在する．

本研究では，入力も出力も自然言語記述とし，非形式的な要求記述を自然言語記述のまま形式化する手順を検討した[3]．ただし，実際にはさまざまな利用者や自然言語，そして開発分野があるので，まずこれらに依存しない手順とすることを前提に構成した．そしてこの形式化をコンピュータによって支援するため，自然言語処理技術を導入して実際の処理を行わせることを考えた．ここでは「自然言語処理の不完全性」を前提とし，単語検索による既存文書の自動追加と要求者自身による記述結果のチェックによって自然言語処理を補い，これによって目的の形式的な要求記述を得ることにした．

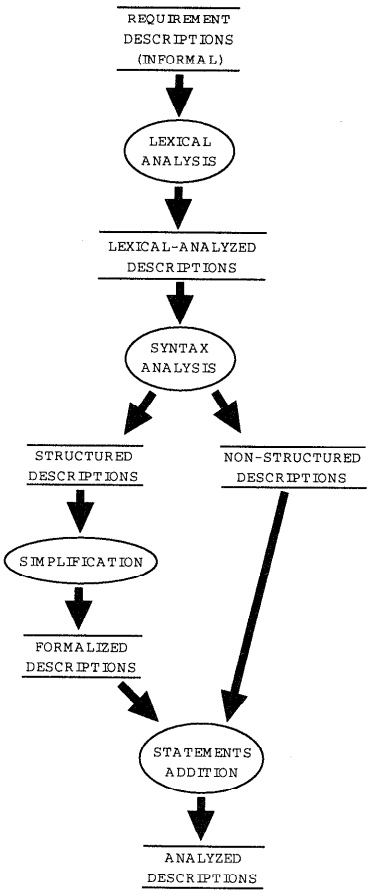


図2 自然言語記述からの要求導出を支援する手法 ASRED
Fig. 2 ASRED: Acquirement supporting method of requirement from descriptions.

ASRED は主に，字句/構文解析部，文章単純化部，文章追加部の 3 つに分かれる．字句/構文解析（lexical/syntax analysis）部では判別しうる単語および文章構造を認識し，結果として，認識できた単語列および構造化された文章を得る．文章単純化（simplification）部では，構造化された文章に対し，後の開発段階で用いる仕様図に変換しやすいように，できる限り名詞および動詞のみで構成される単文にする．名詞/動詞を重視するのは，開発に必要な情報の多くは機能およびその処理対象，あるいは実体およびその振舞いとして認識されるからである[4]．文章追加（statements addition）部では，文章構造が認識できなかった単語列および単純化できた文章中の単語を利用して関連文章を追加する．

3. 日本語による要求導出支援システム ARDES/J の設計および試作

ARDES（the system of Acquiring Requirement and DEriving Specification from descriptions written in a natural language）は，主に記述からの要求抽出手順を規定するASREDを拡張し，記述から要求仕様を導出する手順を支援システムとして体系付けたものである．基本的にはASREDに，システムとして実現するにあたって必要な外部情報やそれらを得る仕組みを明確にし，さらに，形式化された記述からの要求仕様の要素の導出を追加したものである．ARDESの大まかな流れは**図3**のようになる．

ASREDにおいて，字句解析/文章追加では多くの外部情報が必要であるが，それら外部情報はできる限りソフトウェアもしくはソフトウェア開発関連に限定されたものであり，それらは外部情報として分野/開発に関する情報，すなわちドメイン知識として獲得しておく必要がある．したがってARDESでは，これらドメイン知識を辞書情報として獲得する機構も用意する．この辞書情報は，主に字句解析で利用する単語辞書，そして文章追加で利用する用語辞書として保持する．

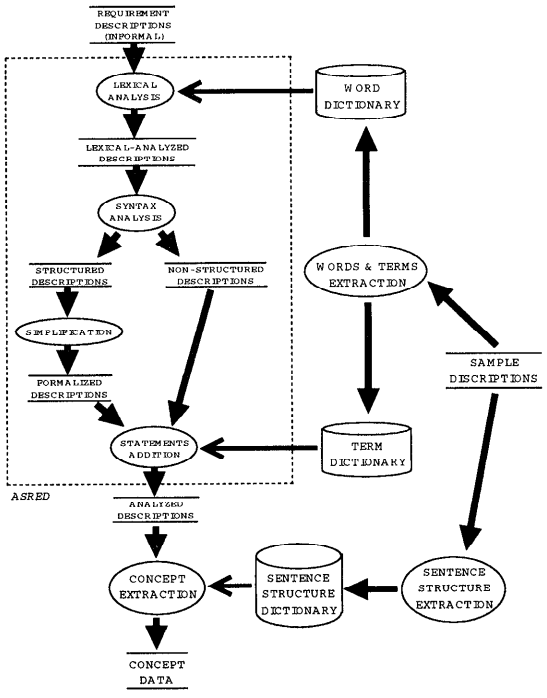


図3 自然言語による仕様導出支援システムARDES
Fig. 3 ARDES: System for acquiring requirement and deriving specification from descriptions written in a natural language.

形式化された記述は自然言語の体裁を保っているので要求者には理解しやすいが，このままでは開発者が開発上の問題を調べるのに適当ではない．このため，形式的な自然言語記述から開発に使われている既存の仕様に変換する作業が必要である．具体的には，単文ごとにその表す意味を調べ，仕様を構成する要素を導くことになる[5),6)]．この変換作業のためにもドメイン知識を辞書情報として用意する必要があるが，ここでは主に，単文の構造あるいは単語の組合せがどのような意味を持つかを対応表のように保持しておく形となる．

このARDESでは特定の自然言語や仕様図を想定していない．したがって，具体的な支援システムとして実現するためには，これら特定の自然言語や仕様図を前提とするものとしてARDESをさらに拡張する必要がある．使用する言語として日本語，出力される仕様としてOMT[4)]の各種モデル図を想定して再設計したものを本研究ではARDES/J（the system for Acquiring Requirement and DEriving Specification from Descriptions written in Japanese）と呼ぶ．**図4**はARDES/Jの大まかな流れである．

再設計の概要は以下のとおりである．

- 字句解析に先立つ前処理（pre-process）の追加．
- 抽出された概念をモデル図に変換する処理（diagram generation）の追加．
- 上記2つを含む処理および辞書情報/表の具体的な設計．

モデル図生成部については，概念データの要素と具体的なモデル図情報の要素を対応付けた表を各モデル図ごとに用意し，その表を利用してモデル図ごとに概念データからモデル図情報に変換する[6)]．

試験的な実装は，主に前処理，字句解析，構文解析，概念抽出，モデル図生成，ユーザインタフェースについて行った．前処理では，サンプル記述[1)]を実際に調べ，あくまで読みやすさ等に使われる読点等は削除するといったことを行うようにした．字句解析については，既存の解析アルゴリズム[7)]を利用してSchemeによるべた書き対応の解析プログラムの試作を行った．構文解析についてもSchemeを使用して簡単なパーサプログラムを試作した．今回利用した構文規則は**図5**のように主に名詞と助詞の組合せを重視したものだが，今回使用したサンプル記述のほとんどはこの規則で解析できた．しかし，まだ日本語文章独特の複文/重文構造が十分には認識できず，現在，構文規則およびパーサを改良している．なお，本研究で規定した単文の構

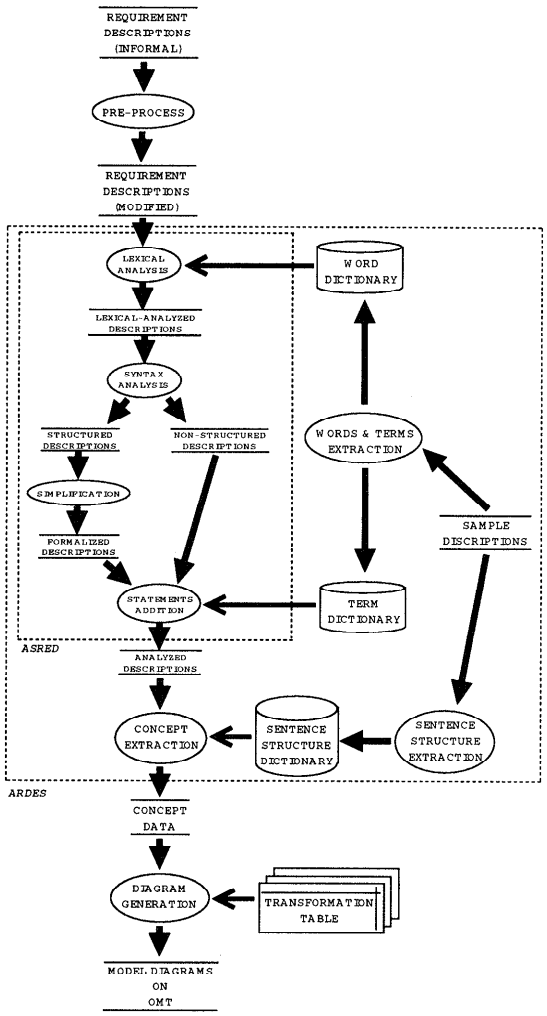


図4 日本語による要求導出支援システム ARDES/J
Fig. 4 ARDES/J: System for acquiring requirement and deriving specification from descriptions written in Japanese.

```
<文> ::= <単文> 読点
<単文> ::= 名詞 助詞 <動詞句>
         | 副詞 <動詞句>
         | 接続詞 <動詞句>
         | 連体詞 名詞 助詞 <動詞句>
         | 形容詞 名詞 助詞 <動詞句>
         | 形容動詞 名詞 助詞 <動詞句>
<動詞句> ::= 動詞 接続助詞 <単文>
         | 動詞
         | <単文>
```

図5 構文解析で使用した構文規則
Fig. 5 Syntax rule for syntax analysis.

```
<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 ;
```

図6 単文の構文規則
Fig. 6 Syntax rule of simple sentence.

```
<リスト> ::= (<名前> <引数> <引数> ...)
<引数>   ::= <種類> | <リスト>
<名前>   ::= " 任意の名前 "
<種類>   ::= att | is-a | has-a | ope
           | mes | rel | act | event | cond
```

図7 中間表現の形式
Fig. 7 Syntax rule of concept data.

```
Entity <Entityの種類> <Entity名> |
  <属性名>: <属性制約名>;
 ...
|;
Relationship <Relationshipの種類>(<Entity名>, ...);
```

図8 モデル図要素の表現形式
Fig. 8 Representation form of model diagrams.

文規則は図6である.

概念抽出部については，中間表現として汎用的な概念表現を規定し，日本語単文との対応付け保持のための辞書の構造を設計し，それらに合わせてサンプル辞書を作成して生成テストを行った．中間表現の形式は図7のとおりである.

モデル図生成部についても主に対応表の構造の設計を行った．概念抽出部と同じように，OMTの各モデル図の要素を表現する形式を規定し，中間表現との対応付けを記述するため表構造を設計し，それらを利用してサンプル的な表を作成して生成テストを行った．モデル図要素の表現形式は複数のモデルに対応できるようERモデルをベースにし（図8），各モデルに合わせて具体的な実体（たとえばクラスや継承関係）を辞書内で表現できるようにした.

本研究では，各種プログラムを操作するユーザインタフェースを，GUIツールキットを利用できるSchemeインタプリタ（STk）上に構築した．Schemeインタプリタなので，各種プログラムや辞書もこのインタプリタ上で実装した．今回はプロトタイプ用ということで，入出力記述ファイルや途中経過を記録するファイル，そして辞書ファイルの指定を直接ユーザが行うことができるようにした．さらに，同じように直接呼び出すことができる単純な対応付けプログラムを組み込むことで各種変換部を実現した.

これらの試作した各プログラム/ツールを利用して

```
対話を通じて，ユーザから要求仕様を獲得するプログラム合成
システムにＰＳＩがある．ＰＳＩは，緊密に相互作用しあうプロ
グラムモジュール（エキスパートと呼ぶ）からなる，知識依存型
のシステムである．ＰＳＩでは，システム全体の動作を仕様獲得
フェーズとプログラム合成フェーズに分けて考えている．仕様獲
得フェーズでは，まずパーサ／インタプリタ・エキスパートが入
力文を解析し，構文解析木を解釈してプログラムネットと呼ぶ
ネットワーク型データ構造を作成する．パーサ／インタプリタ・
エキスパートは約７０のプログラミング概念と約１７５語の語彙
に関する知識をもっている．プログラミング概念のなかには，
データ構造，制御構造，プリミティブな演算，複雑なアルゴリズ
ムの概念などが含まれている．
  …
```

図 9 サンプル記述
Fig. 9 Sample description.

```
対話を通じて，ユーザから要求仕様を獲得する，プログラム合
成システムにＰＳＩがある．ＰＳＩは緊密に相互作用しあう，プ
ログラムモジュールからなる，知識依存型のシステムである．
ＰＳＩではシステム全体の動作を仕様獲得フェーズとプログラム
合成フェーズに分けて，考えて，いる．仕様獲得フェーズではま
ずパーサ・インタプリタエキスパートが入力文を解析し，構文解
析木を解釈して，プログラムネットと呼ぶ，ネットワーク型デー
タ構造を作成する．パーサ・インタプリタエキスパートは約７０
のプログラミング概念と約１７５語の語彙に関する，知識をもっ
て，いる．プログラミング概念のなかにはデータ構造・制御構
造・プリミティブな演算・複雑なアルゴリズムの概念などが含ま
れている．
  …
```

図 10 前処理後のサンプル記述
Fig. 10 Sample description after pre-processing.

```
(define word-dict-txt
  '((プリミティブな @adj) (複雑な @adj) (曖昧な @adj)
    (新たな @adj) (適切な @adj) (このような @adj)
    (どんな @adj) (どのような @adj) (完全で @adj)
    (主な @adj)

    (緊密に @adv) (まず @adv) (たとえば @adv)
    (厳密に @adv) (次に @adv) (新たに @adv)

    (獲得する @verb) (通じて @verb) (呼ぶ @verb)
    (相互作用しあう @verb) (分けて @verb) (考えて @verb)
      …

    (・ @nakaten) (. @period) (, @comma)
    ( ( @lpar) () @rpar)))
```

図 11 サンプル記述解析用の単語辞書
Fig. 11 Word dictionary for sample description.

行った記述解析の例を，ARDES/J の各処理部ごとに順次示していく．

今回利用したサンプル記述は，文献 1) でプログラム合成システムの事例として記述されていたものである（**図 9**）．文字数は句読点・記号を含めて 1019 文字，全体は 4 段落で構成されている．このサンプル記述を前処理したものの一部を**図 10** に示す．

図 11 は，字句解析を行わせるために作成した単語辞書の一部である．今回はサンプル記述を解析するためだけに作成したので，他の記述に対しては基本的な助詞等以外は認識できない．字句解析後のサンプル記述の一部を**図 12** に示す．単語認識については，単純に長いものを優先的に認識させた．結果として，サンプル記述は全体で単語数 322（記号も 1 つの単語と

```
((対話 @noun) (を @pp) (通じて @verb) (ユーザ @noun)
 (から @pp) (要求仕様 @noun) (を @pp) (獲得する @verb)
 (プログラム合成システム @noun) (に @pp) (ＰＳＩ @noun)
(が @pp) (ある @verb))

((ＰＳＩ @noun) (は @pp) (緊密に @adv)
 (相互作用しあう @verb) (プログラムモジュール @noun)
 ( ( @lpar) (エキスパート @noun) (と @pp) (呼ぶ @verb)
 () @rpar) (から @pp) (なる @verb) (知識依存型 @noun)
 (の @pp) (システム @noun) (で @pp) (ある @verb))

((ＰＳＩ @noun) (で @pp) (は @pp) (システム全体 @noun)
 (の @pp) (動作 @noun) (を @pp) (仕様獲得フェーズ @noun)
 (と @pp) (プログラム合成フェーズ @noun) (に @pp)
 (分けて @verb) (考えて @verb) (いる @verb))

…
```

図 12 字句解析後のサンプル記述
Fig. 12 Sample description after lexical analyzing.

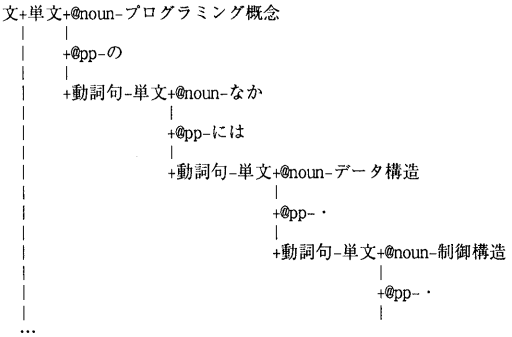


図 13 構文解析後のサンプル記述
Fig. 13 Sample description after syntax analyzing.

```
プログラミング概念 は データ構造 を 含む .
データ構造 は 概念 で ある .
プログラミング概念 は 制御構造 を 含む .
制御構造 は 概念 で ある .
プログラミング概念 は 演算 を 含む .
演算 は 概念 で ある .
演算 は プリミティブ で ある .
プログラミング概念 は アルゴリズム を 含む .
アルゴリズム は 概念 で ある .
パーサ は プログラミング概念 を もつ .
パーサ は 知識 を もつ .
知識 は 語彙 に 関する .
…
```

図 14 形式化されたサンプル記述
Fig. 14 Formalized sample description.

見なす）である．構文解析後のサンプル記述の一部を**図 13** に示す．

図 14 は，構文解析後のサンプル記述から手作業で作成した単文群である．構文解析部が未熟なため単文化は今回は手作業で行った．概念抽出用辞書の一部を**図 15** に示す．概念抽出部は Scheme で作成したので，処理を簡単にするため辞書もリスト構造で表現されている．単文群から抽出された概念の一部を**図 16** に，モデル図生成用の対応表の一部を**図 17** に示す．この表も概念抽出用辞書同様リスト構造で表現されている．概念から OMT のオブジェクトモデルを生成した結果

```
(((単文 (名詞句列 (名詞句 (名詞 名詞1) (助詞 "は"))
                  (名詞句 (名詞 名詞2) (助詞 "を")))
        (動詞 "含む"))
 (単文 (名詞句列 (名詞句 (名詞 名詞2) (助詞 "は"))
                  (名詞句 (名詞 名詞1) (助詞 "の"))
                  (名詞句 (名詞 "一部") (助詞 "で")))
        (動詞 "ある")))
 (名詞2 (名詞1 has-a)))
...
```

図15 サンプル記述解析用の概念抽出辞書
Fig. 15 Sentence structure dictionary for sample description.

```
(データ構造 (プログラミング概念 has-a))
(概念 (データ構造 is-a))
(制御構造 (プログラミング概念 has-a))
(概念 (制御構造 is-a))
(演算 (プログラミング概念 has-a))
(概念 (演算 is-a))
(プリミティブ (演算 is-a))
...
```

図16 サンプル記述から抽出された概念
Fig. 16 Concept data derived from sample description.

```
(((名前1 (名前2 has-a)))
 (("Entity" "class" 名前1 NONE)
  ("Entity" "class" 名前2 NONE)
  ("Entity" "aggregation" <nonel> NONE)
  ("Relationship" "upper" <nonel> 名前2)
  ("Relationship" "lower" <nonel> 名前1)))
(((名前1 (名前2 (名前3 ope))))
 (("Entity" "class" 名前1 NONE)
  ("Entity" "class" 名前2 NONE)
  ("Entity" "association" 名前3 NONE)
  ("Relationship" "none" 名前1 名前3)
  ("Relationship" "none" 名前2 名前3)))
...
```

図17 モデル図生成用の対応表
Fig. 17 Transformation table for generating models.

の一部およびそれを図として表現したものを**図18**に示す.

図19はユーザインタフェースの実現例である.

4. 関連研究

本研究と背景/目的の近い研究がいくつかなされている. 英語記述からオブジェクト構造を認識し設計するもの[8], 仕様書の記述/変換を統合して行うもの[10], 記述から仕様要素を導出して形式化する手法/ツール[11),12)]などである. また, 開発において非形式的な記述を扱うための研究[13),14)]や, 仕様記述の非形式性をコンピュータを利用して解消するための研究[15)]も行われている. しかしこれらの多くは主に設計段階で使用される手法/システムであり, また, 自動化が行えるほど手続き化されていなかったり, 設計/プログラミングや文書化の手間を省くといった目的に沿ったもの, 手作業の負担の軽減にとどまっているものがほとんどである.

```
Entity class データ構造 ||;
Entity class プログラミング概念 ||;
Entity class 概念 ||;
Entity class 制御構造 ||;
Entity class 演算 ||;
Entity class プリミティブ ||;
Entity aggregation <none1> ||;
Entity inheritance <none2> ||;
Entity aggregation <none3> ||;
Entity inheritance <none4> ||;
Entity aggregation <none5> ||;
Entity inheritance <none6> ||;
Entity inheritance <none7> ||;
Relationship upper(<none1>, プログラミング概念);
Relationship lower(<none1>, データ構造);
Relationship upper(<none2>, 概念);
Relationship lower(<none2>, データ構造);
Relationship upper(<none3>, プログラミング概念);
Relationship lower(<none3>, 制御構造);
Relationship upper(<none4>, 概念);
Relationship lower(<none4>, 制御構造);
Relationship upper(<none5>, プログラミング概念);
Relationship lower(<none5>, 演算);
Relationship upper(<none6>, 概念);
Relationship lower(<none6>, 演算);
Relationship upper(<none7>, プリミティブ);
Relationship lower(<none7>, 演算);
...
```

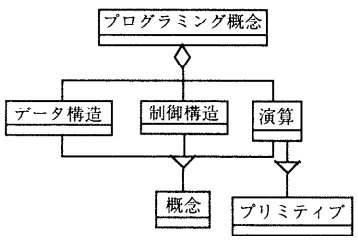


図18 生成されたオブジェクトモデル図
Fig. 18 Object model derived from sample description.

オブジェクト構造の認識/設計のための手法/システムの研究[8),9)]は, 自然言語(英語)で書かれた非形式的な要求記述書から重要な単語をオブジェクトモデルの観点から取り出し, それらを分類してモジュール構造を設計することを目的としている. 首尾一貫した手続きを提供しているので, 設計者によって異なりやすい設計結果が小さな違いで済む. しかしこれらはあくまで設計作業の支援システムであり, コンピュータ上での作業を効率化させるツール群にとどまっている.

ソフトウェア要求定義のためのモデル表現に関する研究[16),17)]では, 要求を形式的な要求フレームモデルで表現できるようにし, それらを用いて日本語やアイコンによって要求を定義/変換して高品質な仕様を効率良く作成することを目的としている. 日本語を用い

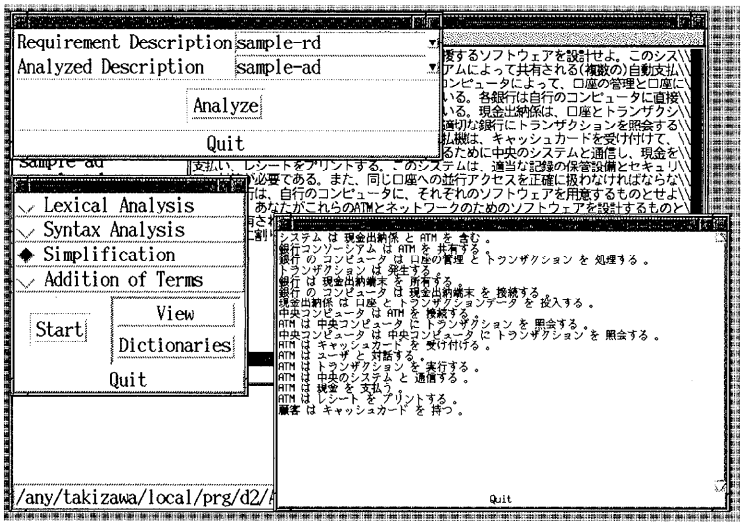


図19 ユーザインタフェースの実現例
Fig. 19 Examples for user interface.

た要求言語解析系も用意されているが，仕様作成のための要求フレームモデルに沿ったもののみを利用するので，非形式的な記述は想定されていない．

SoftDA[10]は，開発段階の上流工程と下流工程を連動させるため，各段階の仕様書の自動変換や設計/コードの自動修正を統合して行うシステムである．このシステムには日本語要求仕様文からのデータフロー図等の生成を行う機能もあるが，扱う文章はあくまで仕様として体裁が整えられたものであり，やはり設計支援のためのものである．

非形式的要求の段階的な洗練を行う手法[11]は，自然言語や図で表現された非形式的な仕様をカテゴリ化して段階的に洗練することで要求と仕様の一致を図ることを目的としている．しかしこの手法も設計者主導で行われる形式化であり，洗練の初期段階で公理/定理が記述可能なモデルベースの形式に書き直される．

また，日本語記述からオブジェクト図などの要素を抽出するためのシステムの研究もある[12]が，このシステムの主眼も設計に必要な情報の獲得であり，自然言語処理によってすでに得られた単語情報に対するオブジェクト要素の認識に重点が置かれている．

以上のように，自然言語を利用した仕様作成を導入しても，多くの場合は開発作業に詳しい設計者のための手法/システムである．本研究の場合，入力を自然言語記述とするのは要求者の要求を開発に反映させるためである．すなわち，手法/システムは要求者支援が目的であり，設計者支援ではない．したがって，仕様情報の獲得手法以上に，ソフトウェア開発という目的に沿った要求者へのインタフェースとしての自然言語処理システムを確立することが重要となる．

5. まとめ

本論文では，要求導出手法 ASRED および支援システム ARDES，そして日本語による支援システム ARDES/J について，その基本概念，設計，試験的な実装について述べた．手法/システムの手順および構造については明確にできたが，実装についてはいくつか課題を残した．

1つは文章の単文化処理である．今回は構文解析部が未熟なため処理プログラムは作成しなかったが，単文化にあたっては構文解析以外にもいくつか問題がある．本研究の場合は特に名詞句の適切な認識が必要だが，扱う文章が非形式的な記述であるため，構文解析が可能でも単文要素の抜出しの時点で複数の選択が可能になることもある．すべての可能な場合の単文を生成しそれを要求者がチェックすることでこの問題を解消する方法もあるが，単語の組合せ等から生成する単文をより少なくできれば要求者の負担も減る．単文化においてもこのための辞書情報を用意し実現するなどが考えられる．

もう1つの課題は，ARDES/J の統合支援システムとしての構築である．主に要求者自身がシステムを利用することを考えると，各段階の処理を意識することなく目的の作業が行えるような支援システムとして構築する必要がある．もちろん，設計者等による辞書構築が効率良く行えるようにすることも重要である．また，辞書の保持および実際の検索は別のマシンで行える方が負荷分散に有効である．今回は GUI ツールキッ

トが利用できるSchemeインタプリタ上ですべての試作を行ったが，検索部だけは最適化された独立プログラムとして実現しそれとの通信によって検索や辞書構築が行えるのであれば，クライアントとしての操作部は主にユーザインタフェースの提供を行うものであればよいことになる．したがって，支援システムの適切な分散化が利用者にとっても重要となる．

以上のような「賢い単文化」や「利用しやすいユーザインタフェース」は，要求者による要求定義を行いやすくするための機能である．しかしこれらはあくまで機械的な処理であるため，要求者自身による訂正/確認を繰り返す必要がある．本研究で提案したシステムは，この訂正/確認を自然言語を利用した作業に限定することで要求者の記述のための学習を最低限に抑えているが，この作業を繰り返すことで要求者自身の記述能力も高まることが予想される．そしてそれを前提とすることで，最終的には比較的単純な自然言語処理技術で支援できる可能性が高い．分散化とあわせて，汎用的なコンピュータシステムの組合せによる支援システム全体の構築が期待できる．

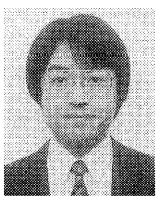
参考文献

1) 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).
2) 滝沢陽三，上田賀一：要求者支援のための記述解析システムASREDの開発，情報処理学会研究会報告，SE104-2, pp.9–16 (1995).
3) 滝沢陽三，上田賀一：オブジェクト指向概念に基づく要求仕様の導出支援システム，ソフトウェア工学の基礎I，pp.153–160，近代科学社 (1996).
4) Rumbaugh, J.，羽生田栄一（監訳）：オブジェクト指向方法論OMT，トッパン (1992).
5) 滝沢陽三，上田賀一：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，情報処理学会研究会報告，SE94-8, pp.57–64 (1993).
6) 滝沢陽三，上田賀一：要求仕様からモデル図情報を抽出するシステムDESPER，信学技報，Vol.SS95-31, pp.1–8 (1995).
7) 草薙 裕：LISPによる自然言語処理，工学図書 (1993).
8) 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).
9) 佐伯元司，蓬莱尚幸，榎本 肇：自然言語仕様からモジュール構造を抽出する手法について，情報処理学会研究会報告，SE57-2, pp.1–8 (1987).
10) 磯田定宏，黒木宏明：統合化CASEシステムSoftDAの機能—上流と下流の統合化，コンピュータソフトウェア，Vol.10, No.2, pp.26–37 (1993).
11) Amoroso, E.: Creating Formal Specifications from Informal Requirements Documents, *ACM SIGSOFT SEN*, Vol.20, No.1, pp.67–70 (1995).
12) 大野雅志，原田 実：オブジェクト指向分析支援システムCAMEO—日本語文章記述からの設計要素の自動抽出，情報処理学会研究会報告，SE99-14, pp.105–112 (1994).
13) Abbott, R.: Program Design by Informal English Descriptions, *Comm. ACM*, Vol.26, No.11, pp.882–894 (1983).
14) Miriyala, K. and Harandi, M.: Automatic Derivation of Formal Software Specifications from Informal Descriptions, *IEEE Trans. Softw. Eng.*, Vol.17, No.10, pp.1126–1142 (1991).
15) Balzer, R., Goldman, N. and Wile, D.: Informality in Program Specifications, *IEEE Trans. Softw. Eng.*, Vol.SE-4, No.2, pp.94–103 (1978).
16) 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).
17) 大西 淳，阿草清滋，大野 豊：要求フレームに基づいたソフトウェア要求仕様化技法，情報処理学会論文誌，Vol.31, No.2, pp.175–181 (1990).

(平成8年4月 9 日受付)
(平成9年1月10日採録)

滝沢 陽三（学生会員）

1969年生．1992年茨城大学工学部情報工学科卒業．1994年同大学院修士課程情報工学専攻修了．現在，同大学院博士後期課程情報・システム科学専攻に在学中．ソフトウェア工学，特に要求定義に関して興味を持つ．電子情報通信学会，日本ソフトウェア科学会各会員．



上田 賀一（正会員）

1961年生．1984年名古屋工業大学工学部情報工学科卒業．1986年同大学院修士課程情報工学専攻修了．1989同大学院博士後期課程電気情報工学専攻修了．工学博士．同年同大学電気情報工学科助手．1990年茨城大学工学部情報工学科講師となり，現在に至る．ソフトウェア工学，プログラミング言語，特に開発支援環境やメタ階層モデル，およびその記述言語に関する研究に従事．ACM，電子情報通信学会，日本ソフトウェア科学会各会員．