ソフトウェア工学 58 － 9
（１９８８．２．４）

自己記述性を有する仕様記述システムの試みについて

蓬莱尚幸* ・佐伯元司** ・榎本肇*

*富士通㈱国際情報社会科学研究所　**東京工業大学工学部

本稿では，仕様記述システムが自己記述性を持つことは広範囲なソフトウェアを対象とする仕様記述システムを構築する上で重要であることを明らかにし，仕様記述システムに自己記述性を持たせる手法について考察する．本稿で考える手法は，自己拡張された記述をシステムが基本的に持つ記述に変換することで自己記述性を実現している．仕様記述をテンプレートと呼ばれる雛形に対象ソフトウェア特有の語句を穴埋めしたものと考え，仕様記述自身とそれに対する変換規則を記述するための枠組みとして，テンプレート記述言語とテンプレート変換言語を提案する．我々が目指すシステムでは，それらの言語を用いて仕様記述者が記述した仕様記述形態の定義と変換規則を用いて拡張された記述形式を含む仕様記述をシステムで用意された基本的な部分で用意された記述形式（核部分）のみからなる仕様記述に変換する機能をシステムに備えることで自己記述性を実現している．さらに，仕様テンプレート記述言語およびテンプレート変換言語に対して仕様記述システムの核部分の仕様記述言語に変換することで意味付けし，自己記述を仕様記述システムの仕様として扱い，仕様記述とそれを作成するための仕様記述システムを統一的に扱うことを考察した．本稿では，核部分として Pure TELLでの仕様記述システムを例にとり論を進める．

On the Specification System with Self-description Mechanism

Hisayuki HORAI†, Motoshi SAEKI‡, Hajime ENOMOTO†

†IIAS-SIS, FUJITSU LIMITED, ‡Tokyo Institute of Technology

†1-17-25, Shin-Kamata, Oota-ku, Tokyo, 144, Japan
‡2-12-1, O-okayama, Meguro-ku, Tokyo, 152, Japan

The method to add the self-description mechanism to a specification system is presented. For the description of the self-description, we need the mechanism for manipulating specifications. We developed two languages. One is "Template Description Language". It is possible to define the syntactic and/or semantic structure of the specification by this language. The other is "Template Transformation Language". It is possible to define the transformation of templates by this language. The specification system with self-description mechanism must have these language and the mechanism for executing the transformation according to user's description of self-descriptions. We also presented the method for the semantics of these language with the specification language of the system. It become possible to manipulate both specification and self-description in the same manner.

1.はじめに

ソフトウェア開発過程における要求仕様定義段階の重要性は，それが開発過程の最初にあたり，以後の段階の入力として開発過程全体に影響を与える点から，活発に議論されている．要求仕様定義での最大の問題点は，いかにして仕様記述者の意図と合った矛盾のない仕様記述を得るかにある．そのために，仕様の検証方法の研究が多く行われている．仕様からプロトタイププログラムの自動合成や仕様自体の直接実行は仕様が規定しているソフトウェアの動作系列を仕様記述者に提示することで仕様記述者が仕様の検証を行うときの補助手段として有効である．また，曖昧な仕様記述者の意図から仕様記述を構築するときに自然言語の語句構造を用いる研究も行われている．

より広い範囲で利用可能な開発支援を行うためには，開発対象となるソフトウェア毎に異なる多様性に対応することのできるシステムでなければならない．しかしながら，現在研究開発されている仕様記述システムでは，そこで用いているモデルや仕様化技法および仕様記述言語による制約が強く，ソフトウェアの多様性に対する柔軟な対応という点では問題点があると思われる．我々は仕様記述システムに自分自身の機能を記述できる機能（自己記述性）を持たせる手法を用いてこの問題点の解決を図った．つまり，対象ソフトウェアの特質に合わせて仕様記述者が用いたい仕様化技法に対する記述を与えることで記述機能を向上させることができる仕様記述システムを構築することが本稿で述べる手法の狙いである．

その他の手段として，様々な仕様記述手法を組み込んだシステムを構築する手法も考えられるが，この手法では，そのシステムで用意していない仕様記述手法については対処できないため，新しい手法を用いたいときにはシステムの作り直しをしなければならず，広範囲なソフトウェアの仕様記述に関する上記の問題点の解決にはなっていないと思われる．

2.自己記述性

本章では，仕様記述システムに自己記述性を持たせることの有用性およびその際に必要となるシステムの全体像とそれが持つべき機能について述べる．

2.1.自己記述性の有用性

本節では，仕様記述システムが持つ自己記述性の用途を列挙することで，その有用性を明らかにする．

①記述概念の拡張：ソフトウェアの仕様を記述する際に統一した記述概念を用いて対象を捉えるということは重要である．しかしながら，対象ソフトウェアの性質の差異によってどのような記述概念（モデルや仕様化技法等）が適しているかは異なる．また，巨大なソフトウェアに関しては機能分割を行い各機能毎に設計する手法が重要であるが，その際，各機能毎に適する記述概念が異なることもありうると予想される．以上のことを考えると，仕様記述システムに様々な記述概念を扱う機能を持たせることは重要である．自己記述機能を用いて対象ソフトウェアの特質に合った記述概念に対する記述形式を与えることで記述概念を拡張することができる．

②ソフトウェア開発の支援：我々の最終的な目的は，仕様記述を中心としたソフトウェア開発支援システムの構築である．このシステムでは，ユーザが記述した仕様を他の記述形態に変換する機能が必要とされる．例えば，論理学を用いた検証を行うためには仕様をその意味である論理式に変換しなければならないし，プロトタイピングのためにはプロトタイププログラムを生成しなければならない．このような仕様を入力とする変換を記述することでソフトウェア開発支援システムの支援機能自体を拡張することもシステムに自己記述性を持たせる狙いの一つである．

③仕様記述言語の表現の拡張：使用する仕様記述言語で用いる表現の良し悪しは，要求仕様定義の作業環境やその作業の最終的な生産物である仕様記述の質に大きく影響する．仕様記述システムのユーザインタフェースの改良は，自己記述機能を用いて使用できる表現を増やすことで実現できる．

2.2.自己記述性を有する仕様記述システム

仕様記述システムが仕様記述言語の表現を拡張する機能扱う機能を持てば，前節で述べた全ての目的が実現できる．そこで，本稿では自己記述性を与える機能（自己拡張機能）を「仕様記述言語で扱える表現を拡張できる機能」と定義する．システムに表現を拡張できる機能を持たせることで，ユーザインタフェースの改良だけでなく，要求仕様定義のための記述概念の拡張やソフトウェア開発支援システムのツールの記述も行える．

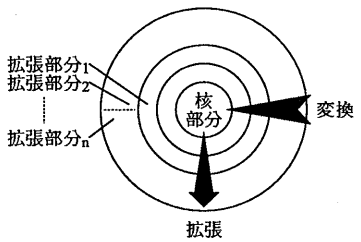

図1　自己記述性を有する仕様記述システム

自己記述性を持つ仕様記述システムの扱うことのできる記述に注目すると，システムの全体像は図1のようになる．システムの扱うことのできる記述は，核部分と拡張部分に分けられる．核部分はシステムが供給する最小範囲の記述能力である．そして，拡張部分は自己記述機能を用いて拡張された仕様記述言語の記述能力である．

仕様記述に対する検証等の様々な手法を拡張部分にも適用できなければ，自己記述を行う有効性は少ないと思われる．これは，それらの手法を核部分に対して用意し，拡張部分に関しては仕様記述を核部分の組み合わせに変換して適応することで実現できる．つまり，新たに加えようとする拡張部分の記述を核部分で与えられている記述の組み合わせに変換する機能を記述することで自己拡張機能の定義を行わせるればよい．システムには，この定義に基づき仕様記述全体を核部分の記述のみからなる仕様記述に変換する機能を持たせる．この機能が仕様記述システムが有する自己記述機能の本質である．全ての仕様記述は最終的に核部分に変換されるため，核部分は仕様記述システムとしての十分な記述能力を持つ必要がある．つまり，核部分だけで十分に仕様記述が行えるシステムになっていなければならない．自己記述性を有する仕様記述システムにおけるプロトタイピングや検証等の手法は，非常に小さい核部分についてのみに用意しておき，拡張部分については全て核部分に変換して適用すればよい．

自己記述機能の定義
　仕様テキストに関する記述が必要
　　↓
テンプレートの導入
　テンプレート記述言語（3.章）
　テンプレート変換言語（4.章）
　　↓
Pure TELL による意味付け（6.章）

図2　本手法のあらまし

自己記述機能の記述のためには，仕様記述者が作成する仕様記述自身を取り扱う必要がある．本稿では，テキスト表現のみを持つ仕様記述言語を対象とした．仕様記述を取り扱うために仕様記述を文法に対応するテンプレートに他のテンプレートや文字列を穴埋めしたものであると捉え，①拡張部分に対するテンプレートのシンタックス／セマンティックスを定義するためのテンプレート記述言語と②自己拡張機能をテンプレートの変換と捉えてそれを定義するためのテンプレート変換言語を導入した（図2）．テンプレートの導入に際しては，記述性や理解性のみならずパフォーマンスに関しても考慮した．例題には，我々が開発を進めている仕様記述に基礎を置くソフトウェア開発支援システムPure TELL に対する本手法の応用を用いる．　Pure TELLは自然言語の語彙に基づく仕様記述言語を中心にしたシステムである．その言語は仕様記述に必要かつ十分な機能を持つように設計されており，本手法の核部分として適している．本稿では，Pure TELL に関する説明は省略する．Pure TELL に関しては参考文献(1)に詳しい．次章以降の本手法の説明では，主格関係代名詞的用法のwhich と代名詞itの導入を例題にする．さらに，本稿では，テンプレート記述言語とテンプレート変換言語の核部分の仕様記述言語による意味付けについて述べる．

3.テンプレート記述言語

表現自体を取り扱う自己記述機能では表現を扱う単位が重要である．仕様テキストを単に文字列として取り扱い，表現の変換を文字列レベルのみで記述することは非常に困難である．仕様テキストの持つ構造を反映した自己記述の単位を導入してその単位毎に変換を記述することが重要である．我々の手法では表現系をパターンの集合と捉え，そのパターンに対して他のパターンや対象ソフトウェア特有の語句を用いて穴埋めしたものが仕様記述であると考える．このパターンをテンプレートと呼ぶ．表現を拡張することは，使用できるパターンの数を増やすことに対応する．つまり，我々の手法を用いるとテンプレート単位の自己拡張機能のモジュール化ができることを意味する．また，漸次パターンを増やすことによって図1のようなシステムのインクリメンタルな個別化／特殊化も行える．

3.1.シンタックスの取り扱い

テンプレートの定義の構文を図3に挙げる．本稿では構文を拡張ＢＮＦで記述する．｛…｝は省略可能を，｛…｝$^+$は1個以上の繰り返しを，｜は選択肢を，下線は非終端記号を，＜…＞は終端記号を表す．図4では，主格関係代名詞的用法のwhich と代名詞itを導入するためにテンプレート<term>と<nominative relative pronoun clause>を定義している．

＜テンプレート定義＞ ::=
　＜テンプレート変数＞　：：＝
　　＜穴埋めパターンの並び＞　end　;
＜穴埋めパターンの並び＞ ::=
　case　＜番号＞)　＜穴埋めパターン＞
　　｛　｜　＜穴埋めパターンの並び＞　｝
＜穴埋めパターン＞ ::=
　＜穴埋めパターン名＞　:　＜スロットの並び＞　.
　　｛　where　＜記述内容に関する記述＞　｝
＜スロットの並び＞ ::= ｛　＜スロット＞｝$^+$　｜　NULL
＜スロット＞ ::= ＜テンプレート変数＞｜＜文字列＞
＜穴埋めパターン名＞ ::=｛　＜識別子＞　｝
＜テンプレート変数＞ ::=　＜テンプレート＞＜変数＞
＜テンプレート＞ ::= ＜　＜識別子＞　＞
＜文字列＞ ::= '　｛　＜文字＞　｝$^+$　'

図3　テンプレートの定義の構文

テンプレート変数は，『そのテンプレートを穴埋めしてできた仕様記述の一部分（テンプレート適用）』を表する．

換言すると，『＜テンプレート＞　＜変数＞』を用いてテンプレートに穴埋めしてできた仕様記述の一部分を表す．テンプレートの定義の本体は，そのテンプレートがどのように穴埋めされるかを表した穴埋めパターンからなる．テンプレートの定義は，：：＝の左辺のテンプレート適用が右辺の｜で区切られたもののどれかでなければならないことを意味している．例えば，図４の１～21行目では，データの指定として核部分で用意されている普通名詞と固有名詞の組み合わせの他に，新しく導入する代名詞itや主格関係代名詞節を使うことができるようにテンプレート<term>を定義している．

```
1    <term> T ::=
2      case 1) proper noun type :
3              <common noun> CN <proper noun> PN .
4                where
5                  1-1) CN is the class of T.
6                  1-2) PN is the object of T.
7      case 2) pronoun type : 'it' .
8                where
9                  2-1) The class of the previous term of T
10                      is the class of T.
11                 2-2) The object of the previous term of T
12                      is the object of T.
13     case 3) relative pronoun type :
14             <nominative relative pronoun clause> NRPC.
15               where
16                 3-1) The modified phrase of the clause of T
17                      is the class of T.
18                 3-2) The class of the first result of
19                      transformation from NRPC.
20                 3-3) NRPC is the clasue of T.
21   end;
22
23   <nominative relative pronoun clause> NRPC ::=
24     case 1) : <common noun> CN 'which' 'is' <predicate> P .
25             where
26               1) CN is the modified phrase of NRPC.
27               2) P is the subordinate of NRPC.
28   end;
```

図４　テンプレートの定義の例

右辺の穴埋めパターンは，ふつう，テンプレートおよび文字列の並びからなる．例えば，図４の23～28行目では，テンプレート<nominative relative pronoun clause>が<common noun> と'which' と'is'と<predicate> の二つのテンプレートと二つの文字列の組み合わせによって穴埋めされることを意味している．図３に現れるNULLは，定義するテンプレートが省略可能であることを意味する特殊なテンプレートである．

変換規則を記述する際に，テンプレートがどのように穴埋めされているかによって規則を変えることができるように，穴埋めパターンには名前が付けれられている．穴埋めパターンの名前は１つのテンプレートの定義内では全て異ならなければならない．例えば，図４の<term>の定義では，穴埋めパターンの３つの候補に対して，それぞれ，proper noun type，pronoun type，relative pronoun typeという名前を付けている（図４の２，７，13行目）．また，右辺の穴埋めパターンが１つだけならば，名前付けを行う必要はない．その際には，図４の24行目のように省略できる．

3.2.記述内容の取り扱い

前節ではシンタックスの定義について述べた．仕様テキストのシンタクッスの取り扱いは，テンプレートを用いて十分行うことができる．しかしながら，シンタクッスに関する記述のみで自己拡張機能を定義することは困難である．

例えば，代名詞を導入するためには，仕様記述中に現れる各代名詞に対してそれの指す仕様記述中の他の部分を決定する方法を自己記述機能の定義として記述する必要がある．このように文脈情報を取り扱うことがしばしば必要となる．

文脈情報に代表される記述内容を取り扱えるための機能が必要となる．記述内容は，シンタックスに付随したものであると考え，シンタックスの記述単位（本稿ではテンプレート適用）を指定して決定／参照する．テンプレート記述言語では穴埋めパターン毎に記述内容に関する記述を行うことができる．

テンプレート適用の意味は，『＜意味値のクラス＞　＜変数＞　ｉｓ　＜意味内容の名前＞　ｏｆ　＜テンプレート変数＞.』の形をした自然言語文を Pure TELLの静的記述風に記述する．意味内容の値（意味値）に関する記述では仕様記述自体を取り扱わなければならないため，　Pure TELLの静的記述に比べてテンプレートをクラスとして扱える点のみ拡張されている．つまり，意味値のクラスとしてテンプレートの他に整数・文字列等がとれる．通常では意味値はシンタックスの違いを反映することが多く，意味内容を構文の記述単位に扱うためには，意味値に関する記述を穴埋めパターン毎に記述する方法が良いと思われる．

例えば，<term> Tの意味は，それが実際に指し示すデータのクラスと名前である．それぞれ，語句the class と the objectで表す．両者とも<term>がどのように穴埋めされるかによって異なる．

<term> Tの指すデータのクラスを表す普通名詞を表すテンプレート<common noun>型のthe class は，proper noun typeでは穴埋めされている<common noun> CNがそのまま意味値となり（５行目），pronoun type ではその<term> Tの直前に現れた<term>のテンプレート適用のクラスが（９～10行目），relative pronoun typeではその主格代名詞節の非修飾語が（16～17行目），それぞれ意味値となる．

また，<term> Tの指すデータの名前を表す固有名詞を表すテンプレート<proper noun> 型のthe objectに関する条件は６，11～12，18～19行目に記述されている．18～19行目に現れるthe first result of transformaitonは変換結果を参照するための語句である．このように，変換結果も意味として統一的に扱える．詳しくは4.章で述べる（18～19行目は，主格関係代名詞節を用いた<term> Tの the objectがその<term> Tを変換した結果である<term>のテンプレート適用のthe objectと等しいことを意味している）．

あるテンプレート適用の一部分を構成するテンプレート

適用（子テンプレート適用）を他のテンプレートの記述や変換規則の記述の際に参照する必要でしばしばある．これも子テンプレート適用を意味とすることで統一的に扱える．20行目のthe clauseや26行目のthe modified phrase や27行目のsubordinate がその例である．

pronoun typeの<term>のthe objectに関する記述を行うためには，各テンプレート適用の直前の<term>のテンプレート適用(the previous term) と最後に現れる<term>のテンプレート適用(the last term) を用いなければならない．このようなテンプレート適用の構造に沿って受け渡される情報を表す語句は全てのテンプレートに対して導入する必要がある．全テンプレートの全穴埋めパターンで記述すれば良いが，非常に記述量が増え，全てのテンプレートに対して導入した意図を表しえない．そこで，意味に関する語句をPure TELL の関数定義風に定義することも許す．図5はthe previous term とthe last term の定義例である．1，11行目に現れるtemplateは全テンプレートを総称するスーパークラス的なテンプレートである．16～17行目のthe last filling template も同様に Pure TELLの関数定義風に定義できる．このような意味に関する語句の定義形態は，テンプレート定義の穴埋めパターン毎の記述の省略記法と見做すことができる．

```
1   <term> T is the previous term of temlate TMPL
2    means that
3     1) Template TMPL1 is the left of TMPL.
4     2)
5      case 2-1) TMPL1 is <term> :
6              T is TMPL1.
7      case 2-2) TMPL1 is not <term> :
8              T is the last term of TMPL1.
9   end;
10
11  <term> T is the last term of temlate TMPL
12   means that
13    case 1) TMPL is <term> :
14           T is TMPL.
15    case 2) TMPL is not <term> :
16           T is the last term of the last
17            filling template of TMPL.
18  end;
```

図5　記述内容の定義の例

3.3.システムで用意されたテンプレート

仕様記述者は，テンプレートに他のテンプレートや対象ソフトウェア特有の単語を穴埋めしながら，仕様を作成してゆく．そこで，仕様記述者が記述する対象ソフトウェア特有の単語で穴埋めるためのテンプレートをシステムで用意する必要がある．これを，入力専用テンプレートと呼び，『＜＄ｉｎｐｕｔ＞』で表す．入力専用テンプレートには，仕様記述者が入力した文字列がそのまま穴埋めされる．

これとは逆に，文脈から出力する文字列が決まるテンプレートも用意する．これを，出力専用テンプレートと呼び，『＜＄ｏｕｔｐｕｔ＞』で表す．出力専用テンプレートの典型的な例は，Pure TELLにおける箇条書きの番号が挙げられる．

テンプレートを穴埋してできている仕様記述を表す文字列型の意味the stringが用意されている．入力専用テンプレートのそれは仕様記述者が入力した文字列が値となる．出力専用テンプレートに関しては，穴埋めされる文字列は文脈から決定されるので，the stringの意味値に関する記述がなければならない．文字列（例えば，図4の'it'，'which'，'is' 等）のthe stringはその文字列自身（つまり，it，which，is 等）である．入力／出力専用テンプレート以外のテンプレート適用のthe stringは，それを構成しているテンプレート適用や文字列のthe stringを文字列として連接したものである（各文字列は空白で区切られる）．NULLであるテンプレート適用のthe stringは空文字列である．

子テンプレートをその出現位置を指定して参照することもそれを意味と捉えて『～ is the ＜出現位置＞ filling of ～』を用いて統一的に扱う．

4.テンプレート変換言語

前述のとおり，自己拡張機能の記述は拡張部分の記述形態を基本部分の記述形態に変換する方法を記述することである．つまり，本手法では拡張部分のテンプレートから核部分で与えられているテンプレートへの変換方法を記述する枠組みを与えることが自己拡張性を与えることになる．しかし，全ての拡張部分を一度に核部分に変換するのは非常に困難だと思われる．この困難を緩和するために，我々の提案するシステムでは，図1の構造に沿って多段階変換を行って核部分に変換する．つまり，新たに導入したいテンプレートに対しては，それより核部分に近い（より基本的な）テンプレートに変換する方法を記述すればよい．

4.1.変換規則の定義

図6はこの変換規則の定義の構文である．図7は主格関係代名詞節と代名詞itを導入するための変換規則の記述例である．

```
＜変換規則定義＞ ::=
  ＜テンプレート＞＜変数＞  ＝＝＞
      {＜番号＞)  ＜テンプレート＞＜変数＞} +
   ｗｈｅｒｅ
     ＜変換に関する条件＞
   ｅｎｄ；
```

図6　変換規則の定義

変換規則は，拡張された記述の単位単位（本手法ではテンプレート）毎に，そのシンタクティックな構造や記述内容を参照しながら，より基本的な記述を生成する方法として，定義される．

```
1   <nominative relative pronoun clause> NRPC ==>
2     1) <term> T.
3     2) <sentence> S.
4    where
5     1) T is proper noun type.
6     2) String ST is new identifier.
7     3) ST is the string of <$input> I.
8     4) The identifier of <proper noun> PN is I.
9     5) <common noun> CN is the modified phrase of NRPC.
10    6) <predicate> P is the subordinate of NRPC.
11    7) T is constructed by [ CN PN ].
12    8) S is constructed by [ T 'is' P '.' ].
13  end;
14
15  <term> T ==>
16    1) <term> T1.
17    2) <sentence> S.
18   where
19    case 1) T is proper noun type :
20       1-1) T1 is T.
21       1-2) S is constructed by [ NULL ].
22    case 2) T is pronoun type :
23       2-1) T1 is the previous term of T.
24       2-2) S is constructed by [ NULL ].
25    case 3) T is relative pronoun type :
26       3-1) <nominative relative pronoun clause> NRPC is
27            the clause of T.
28       3-2) T1 is the first result of transformation from NRPC.
29       3-3) S is the second result of transformation from NRPC.
30  end;
31
32  <sentence> S ==>
33    1) <sentences> SS.
34   where
35    1) <term> T is the subject of S.
36    2) <predicate> P is the predicate of S.
37    3) <be verb> B is the predicate of S.
38    4) <term> T1 is the first result of transformation from T.
39    5) <predicate> P1 is the first result of transformation from P.
40    6) <sentence> S1 is constructed by [ T1 B P1 '.' ].
41    7) SS is the conjunction of the second result of
42         transformation from T and the second result of
43          transformation from P and S1.
44  end;
```

図7　変換規則の定義の例

変換規則の定義は，(1) ==>の左辺にある変換の対象となるテンプレートと(2) ==>の右辺に列挙される変換結果のテンプレートと(3) whereに続くこの変換の本体である変換に関する条件／操作とからなる．

変換結果のテンプレートを複数個取れるようにしたのは，文脈のある記述を生成したいからである．例えば，図8のように，主格関係代名詞節は変換後に変数と置き変わるが，その節を含む文と同じレベルにその変数の値を従属節に従って制約する文も同時に生成しなければならない．このようなことは，下位のテンプレートに関する変換において複数個の結果を用意し，上位のテンプレートに関する変換において下位のテンプレートの変換結果を参照して変換を行うことで記述できる．主格関係代名詞節の例では，テンプレート<nominative relative pronoun clause>で制約となる文を生成し，テンプレート<term>や<sentence>などを通して，文の集合を表すテンプレート<sentences> の変換で他の文と連接する（図7の2～3，16～17行目参照）．

4.2.変換に関する条件の記述

変換に関する条件の記述は，意味内容の定義と同様にPure TELL の静的記述風に記述する．そこでは，3.章で述べた意味値への参照のほかに，以下の条件を表す語句がシステムで用意されている．

①テンプレートの合成

テンプレートを合成は，『＜テンプレート変数＞ is constructed by ［ ＜穴埋めパターン＞ ］．』で表す．

例えば，図7の12行目は，<sentence> Sが<term> T と'is'と <predicate> P '.'とによって穴埋めされて合成されることを意味する．

②新しい文字列の生成

変換の途中で適当な文字列（例えば，図8のNEW1に関する文字列）で穴埋めされた入力専用テンプレートを生成する必要がある．new identifierはこのような文字列を生成するための語句である．（図7の7行目参照）この新しく生成された文字列は，仕様に現れない全く新しい文字列である．

一方，出力専用テンプレートを新しく生成したいときには，一般的にはこのnew identifierを用いることはない．そのかわり，the stringの意味値を決定するような条件を記述することになる．

③変換結果の参照

文脈を持つテンプレートの構造を得るために，あるテンプレートの変換で他のテンプレートに対する変換結果を参照するために，『＜テンプレート変数＞ is the ＜順序数＞ result of transformation from ＜テンプレート変数＞．』を用意する（図7の28，29，37，38，41～43行目参照）．

```
    ⋮
4) …… integer which is the length of string S ….
    ⋮
                ↓
    ⋮
4) …… integer NEW1 …….
5) Integer NEW1 is the length of string S.
    ⋮
```

図8　主格関係代名詞節の変換例

5.記述例

本稿で例題として用いている Pure TELLに主格関係代名詞的用法のwhich と代名詞itの導入に関する変換規則のあらましを説明する．

図7の1～13行目のテンプレート<nominative relative pronoun clause> に対する変換規則では，(1)変数（固有名詞）の生成（6～8行目），(2)実際にこの節と置き換えるた

めのproper noun typeの<term>の生成（11行目），(3)導入した変数の条件を表す文の生成（12行目）を定義する．

図7の15～30行目の<term>に対する変換規則では，<term>への穴埋めされ方に応じて（19，22，25行目），実際に置き換える<term>を定義している．(1)proper noun typeのときは何も変換しない（20行目）．(2)pronoun typeのときはその<term>の直前に現れる<term>を用いる（23行目）．(3)relative pronoun type のときはその節の変換結果である<term>を用いる（28行目）．さらに，relative pronoun typeのときのみその節の変換結果である条件を表す文を用いて2番目の変換結果を決定する（21，24，29行目）．

<sentence>（文）は Pure TELLに本来的に存在するテンプレートであるが，主格関係代名詞の導入のために変換規則が付け加えられる．この変換結果は，<sentences>（文の並び）となる．図7の32～44行目の <sentences>に対する変換規則では，(1)主語や述語の変換結果を利用してproper noun typeの<term>のみからなる文を生成し（35～40行目），(2)主語や述語内の<term>で生成された条件を表す文を連ねる（41～43行目）．

この他に，<sentences>に対してそれを構成する各々の文の変換結果である文の並びを連ねる操作や，<predicate>等の<term>と<sentence>との間にあたるテンプレートに対して条件を表す文を連ねつつ<sentence>に向けて吸い上げるような操作を記述する必要がある．

6.仕様記述としての自己記述

拡張部分は核部分の中の自己拡張機能を複数回使って核部分に変換できることになる．つまり，自己拡張機能を記述することは仕様記述システムの拡張部分の仕様記述であると捉えることができる．そこで，仕様記述である以上は自己拡張機能そのものも核部分の仕様記述言語で意味付けできなければならないと考え，本手法で用いたテンプレートとその変換をテキスト表現変換ソフトウェアの仕様と対応させることで本手法自身を定義する．自己拡張機能の記述について仕様化することには，自己拡張機能に関して検証等を仕様記述に対するそれらを用いて行うことができるという利点がある．本稿で例として用いている Pure TELLでは，入出力関係が仕様の本質であるシステムを記述するために用意された静的記述を用いて定義できる．

6.1.テンプレート記述言語の仕様化

テンプレートのシンタックスに関する部分をクラス定義を用いて定義する方法について述べる．図9に例として，図4で定義したテンプレートに対するクラス定義を挙げる．テンプレート毎にその名前と同じ名前のクラスを直和型として定義する．例えば，テンプレート<term>と<nominative relative pronoun clause>に対して，それぞれ，クラスtermとnominative relative pronoun clauseをそれぞれ定義する（1，12行目）．この直和型の各要素は穴埋めパターンに対応して穴埋めパターン名をインスペクタ述語とする直積型である（2，5，7，13行目）．この直積型の各要素は穴埋めパターン内のスロットに対応し，the ～ fillingをセレクタ関数とするテンプレートに対応するクラスまたは文字列に対応する列挙要素が1個だけである列挙型である（3～4，6，8～9，14～17行目）．

```
1   Term is sum class
2     case 1) proper noun type : product class
3       1-1) the first filling : common noun.
4       1-2) the second filling : proper noun.
5     case 2) pronoun type : product class
6       2-1) the first filling : it_class.
7     case 3) relative pronoun type : product class
8       3-1) the first filling :
9           nominative relative pronoun clause.
10  end;
11
12  Nominative relative pronoun clause is sum class
13    case 1) product class
14      1-1) the first filling : common noun.
15      1-2) the second filling : which_class.
16      1-3) the third filling : is_class.
17      1-4) the fourth filling : predicate.
18  end;
```

図9　シンタックスの仕様化

入力専用テンプレートおよび出力専用テンプレートに対するクラスに対しては，1個の文字列からなるクラスとする．NULLも文字列と同様に扱い，列挙要素が1個だけの列挙型クラスnullを対応させる．

意味に関する記述は，導入した語句（図4のthe stringなど）に対するPure TELL での定義に対応させることで意味付けする．テンプレート記述言語では穴埋めパターン毎に散在している意味に関する記述は Pure TELL風になっているので，意味を表す語句毎に集めてきてテンプレートをそれに対応するクラスと置き換えるだけでPure TELL の関数定義にすることができる．また，図5のような意味に関する語句の定義は，テンプレートをクラスに置き換えるだけでよい．図10は，図4，図5で示した意味に関する記述に対する関数定義である．

6.2.テンプレート変換言語の仕様化

意味に関する記述と同様に，変換規則も Pure TELLの関数定義で仕様化できる．図11は図7の変換規則の定義に対する関数定義である．

各変換規則は，『＜変換対象＞ is transformed to ＜変換結果＞．』の形の関数定義で意味付けされる．変換結果は複数取りえるため，テンプレートの列となる（1～2，15，32行目）．この変換結果となるテンプレートの列と各変換結果との関係が定義本体に加えられる（12，29，43行目）．

また，変換結果を参照するための語句 the result of transformation to は，transformed toとconstructed by

```
Common noun CN is the class of term T
 means that
  case 1) T is proper noun type :
       CN is the first filling of T.
  case 2) T is pronoun type :
       CN is the class of the previous term of T.
  case 3) T is relative pronoun type :
       CN is the modified phrase of the clause of T.
end;

Proper noun PN is the object of term T
 means that
  case 1) T is proper noun type :
       CN is the second filling of T.
  case 2) T is pronoun type :
       CN is the object of the previous term of T.
  case 3) T is relative pronoun type :
       CN is the class of the first result of transformation
         from the first filling of T.
end;

Nominative relative pronoun clause NRPC is
   the clause of term T
 means that
  1) T is relative pronoun type.
  2) NRPC is the first filling of T.
end;

Common noun CN is the modified phrase of
   nominative relative pronoun clause NRPC
 means that
  1) CN is the fist filling of NRPC.
end;

Predicate P is the subordinate of
   nominative relative pronoun clause NRPC
 means that
  1) CN is the fourth filling of NRPC.
end;

Term T is the previous term of temlate TMPL
 means that
  1) Template TMPL1 is the left of TMPL.
  2)
   case 2-1) TMPL1 is term :
        T is TMPL1.
   case 2-2) TMPL1 is not term :
        T is the last term of TMPL1.
end;

Term T is the last term of temlate TMPL
 means that
  case 1) TMPL is term :
       T is TMPL.
  case 2) TMPL is not term :
       T is the last term of the last filling template of TMPL.
end;
```

図10　意味の仕様化

を用いて定義できる．その他のシステムが用意した語句に対しても，PureTELLで意味付けできる．

このようにテンプレート記述言語およびテンプレート変換言語を仕様化することで，仕様記述システムに自己記述性を与えるためには，仕様テキストの構文を扱うための枠組み（本稿では，テンプレートとそれに対する操作）を与えればよいことが明らかになった．

7.まとめ

本稿では，それ自身が仕様記述言語として十分な機能を持つ核部分を前提にして，仕様記述言語の表現形式を拡張

```
Nominative relative pronoun clause NRPC
 is transformed to sequence of template TS
 means that
  1) Term T is proper noun type.
  2) String ST is new identifier.
  3) ST is the string of input only type I.
  4) The identifier of proper noun PN is I.
  5) Common noun CN is the modified phrase of NRPC.
  6) Predicate P is the subordinate of NRPC.
  7) T is constructed by CN and PN.
  8) Sentence S is constructed by T, a is_class, P and a dot_class.
  9) TS is made by T and S.
end;

Term T is transformed to sequence of template TS
 means that
  1)
   case 1-1) T is proper noun type :
      1-1-1) Term T1 is T.
      1-1-2) Sentence S is constructed by a null.
   case 1-2) T is pronoun type :
      1-2-1) T1 is the previous term of T.
      1-2-2) S is constructed by a null.
   case 1-3) T is relative pronoun type :
      1-3-1) Nominative relative pronoun clause NRPC is
           the clause of T.
      1-3-2) T1 is the first result of transformation from NRPC.
      1-3-3) S is the second result of transformation from NRPC.
  2) TS is made by T1 and S.
end;

Sentence S is transformed to sequence of template TS
 means that
  1) Term T is the subject of S.
  2) Predicate P is the predicate of S.
  3) Be verb B is the predicate of S.
  4) Term T1 is the first result of transformation from T.
  5) Predicate P1 is the first result of transformation from P.
  6) Sentence S1 is constructed by T1, B, P1 and a dot_class.
  7) Sentences SS is the conjunction of the second result of
       transformation from T and the second result of
         transformation from P and S1.
  8) TS is made by SS.
end;
```

図11　変換規則の仕様化

することでシステムに自己記述性を持たせる試みについて述べた．そのためには仕様記述から仕様記述への変換を記述するための枠組みを提供すればよく，われわれはテンプレートを中心とした枠組みを提案した．

本稿では，「仕様記述言語の表現の拡張」の例である Pure TELL における表現の拡張を用いて説明した．今後は，仕様記述者が与えたシステムに関する記述をもとに実際に変換する手法を検討し，本手法を導入した仕様記述システムを作成し，記述概念の拡張を含めた自己拡張機能を記述し，例題の記述を通して本手法の評価を行いたい．また，他の表現形式（図や表）に関する自己拡張の手法についても考察してゆきたい．

参考文献：

(1)Saeki,M., Horai,H., Toyama, K., Uematsu,N., and Enomoto, H. : Specification Framework Based on Natural Language, Proc. of 4th International Workshop on Software Specification and Design, pp.87-94 (1987)