ソフトウェア工学 71－5
（1990．2．8）

自然言語記述に基づくソフトウェア設計支援ツール

細谷隆志*† 若尾正樹† 金子剛† 佐伯元司† 榎本肇‡
*富士通株式会社 †東京工業大学 ‡芝浦工業大学

本論文は自然言語（英語）で記述された非形式仕様から形式仕様のモジュール構造を得るための設計作業を支援するツールについて述べる．本設計手法では，自然言語文の構成要素である語句をオブジェクトモデルにおけるソフトウェアの構成要素であるクラス，オブジェクト，属性，メソッドといった概念に対応づけるために，自然言語文の意味において重要な役割を果たす動詞に注目する．そしてそれらの動詞が取りうる格構造をオブジェクトモデルの観点からの分類し，各分類ごとに自然言語文の構成要素をソフトウェアの構成要素に対応づけるための規則を定義する．これらの分類法を用いて設計者は，仕様文中の名詞，動詞及びそれらの主語，目的語を表に整理する．規則をこれらの表に適用することによってモジュール構造ドキュメントの原型となるテンプレートに各表の要素を埋め込んでいき，モジュール構造ドキュメントを導く．支援ツールでは自然言語仕様記述，表，モジュール構造ドキュメントをファイルとして管理する．さらに，表やテンプレートに当てはめられた各要素が，自然言語仕様の中のどの部分にあるかをポインタによるリンクで表し，人間が分類や関係の状態を参照できるようにしたり，設計過程における生成物間の因果関係を保存・管理できるようにしている．これを実現するために，単語と単語をリンクできるデータベースとデータベースの構造に合った構造化エディタを開発し，マン・マシンインターフェースとしてX-Windowシステムを使用する．さらに，簡単な例題の作業を通して手法・ツールの評価を行なう．

Support Tool of Software Design based on natural language specifications

Takashi Hosoya*† Masaki Wakao† Tsuyoshi Kaneko† Motoshi Saeki† Hajime Enomoto‡
* Fujitsu Limited †Tokyo Institute of Technology ‡Shibaura Institute of Technology

A support tool to design incrementally a formal specification from an informal specification written in natural-language is presented. Through the design task, the structure of software modules based on an object oriented model is interactively extracted from the informal English description. To associate each word in the sentences with a software concept, e.g. *class, attribute,* and *method,* we conceptrate on type of verb patterns. We have provided the categories of verb and noun patterns, and associated rules to derive the module structure documents. Designers gradually produce various kinds of intermediate documents such as *verb table, noun tabel* etc. as the design task goes on. The integration management of those products are supported by our tool. It is necessary to manage the relationships among the elements in the different products. We have developed a structured editor for the products, and database system which can reconstruct the relationships amongs the elements according to the progression of design tasks. This tool is implemanted on X Window System of SUN workstations.

1 はじめに

ソフトウェアを開発するための様々な技法や，その技法に基づいた支援ツールが開発され，実用にむけての研究がすすめられている[6]．そのなかで自然言語の仕様からソフトウェアを作り上げるための技法がある[1, 5]．自然言語がソフトウェア作成技法中で使用される理由として，ソフトウェアを作成する人間にとって，記述がしやすい，読みやすいといった点を挙げることができるだろう．また，自然言語では自由な記述が行なえるため，ソフトウェアの仕様に対して，作成者が未決定部分の記述をあとに延ばし，具体的に明確になった時点で，その記述を追加するという，仕様を増殖的に作成していくことが可能である．

自然言語を使用して記述を行なったソフトウェア仕様書の管理をハイパーテキストにより行なう試みもなされている[8, 9, 2, 3]．本論文では，人間が比較的容易に記述できる自然言語の仕様をもとに，そこからモジュール構造を得る手法に基づいた支援ツールを作成し，簡単な例題に対して適用することにより技法・ツールの評価を行なう．この自然言語仕様からのモジュール構造を得る手法では，自然言語の記述を動詞（Verb）に主眼を置き，その動詞が取りうる格構造をオブジェクトモデルの観点から分類し，動詞と名詞の関係，動詞と動詞の関係を明確にすることによって文章からモジュール構造を抽出する．抽出作業過程においては中間生成物として種々の表が作られる．得られた表をもとに，あらかじめ用意した規則と，人間の自然言語理解の能力によりモジュール構造ドキュメントの原型であるテンプレートに単語をはめ込み，モジュール構造ドキュメントを得る．このモジュール構造ドキュメントは，人間が自然言語の記述がなくとも常識的に理解できるという理由で省略されている部分が欠けている．この部分を明確化し，記述を補うことによって完全なモジュール構造ドキュメントを得る．

支援ツールでは自然言語仕様記述，表，モジュール構造ドキュメントをファイルとして管理する．さらに，表やテンプレートに当てはめられた各要素が，自然言語仕様の中のどの部分にあるかをポインタによるリンクで表し，人間が分類や関係の状態を参照できるようにしたり，設計過程における生成物間の因果関係を保存・管理できるようにしている．リンクのはり方は設計の規則に依存している．そのため仕様の変更や作業の後戻りで，自然言語記述や表が変更した場合，規則に従ってリンクのはりかえが半自動的に行なわれる（リンクの動的なはりかえ）．規則により明確に行なえるものはツールが自動的に行ない，不明確な部分は候補を示すことにより人間の判断を促す．またリンクのはりかえが自動的にできない場合には，メッセージを出力しリンクのはりかえが必要なことを知らせる．

このような処理を実現するために支援ツールでは，自然言語の単語を処理する単位として扱う必要がある．そのため，単語を処理・蓄積の単位としたデータベース（DB）とエディタの開発を独自に行なった．さらにマン・マシンインターフェースとしてWindowシステムを使用し，自然言語仕様からモジュール構造を得る手法をサポートしている．

インプリメントに使用した言語はC言語である．自然言語仕様からモジュール構造を得る手法で対象としている自然言語仕様は，制限のない自然言語で記述された仕様である．形式的な仕様記述言語はTELLの記述言語[4]としている．

2 自然言語記述からソフトウェア設計を行なう手法

本設計手法では，ソフトウェアシステムを形式的にとらえるための重要なモデルとしてオブジェクト指向型のモデルを導入している．オブジェクト指向モデルでは，実体を持ったものをオブジェクト，共通の性質を持ったオブジェクトの集まりをクラス，オブジェクトの内部状態を表すものを属性と呼んでいる．属性はオブジェクトに操作を加えることによって変化する．あるオブジェクトに操作を加えることは，そのオブジェクトにメッセージを送信することと考えられる．操作により属性がどのように変化するかを記述したものがメソッドである．このようなオブジェクト，クラス，属性，メソッドといったソフトウェアコンポーネントを自然言語の要素である名詞，動詞，名詞と動詞の関係，動詞と動詞の関係などに対応づけてソフトウェア設計を行なう．

1つの自然言語仕様から，ソフトウェアコンポーネントへの対応づけを1回のみ行なっただけでは，得られるモジュール構造ドキュメントは記述の抜けた部分の多い，不完全なものになる．これは入力となる自然言語仕様の一部が省略されており，記述内容が完全なものでないからである．そのため得られたモジュール構造ドキュメントの不完全な部分をさらに明確に自然言語で記述し，同様の方法によりモジュール構造ドキュメントを求め，前のモジュール構造ドキュメントの不完全な部分と入れ換えを行ない，より完全なモジュール仕様書にすることができる．このようなことを何度か繰り返すうちに完全な仕様書を得ることができる．(図1)

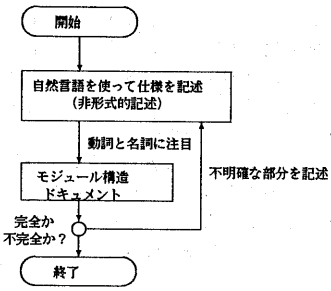

図 1: 全体的な手順

2.1 設計のための表

ソフトウェアコンポーネントに自然言語の要素を対応づけるには文の動詞に注目し，文の格構造を分類・整理を行なう．この作業を行なうために表を用いている．表の種類は以下の4つがある．(表の例は6章を参照)

- 動詞表
- 名詞表
- 動作表
- 動作関係表

このような表を使用し，文の格構造を分類し，モジュール構造を得る手順を図2に示す．この作業はあらかじめ用意されている規則が適用され，自動的に進む部分もあるが，人間が自然言語理解の能力によって作業を進めていく．

(1) 動詞表
自然言語の記述から動詞を抜きだし，それを表1のようなカテゴリに従って分類する．また，抜き出した動詞に関連する主語（subject），目的語（object），補語（complement）も抜きだし，同じ表に記述する．

(2) 名詞表
自然言語の記述から名詞を抜きだし，それを表2のようなカテゴリに従って分類する．

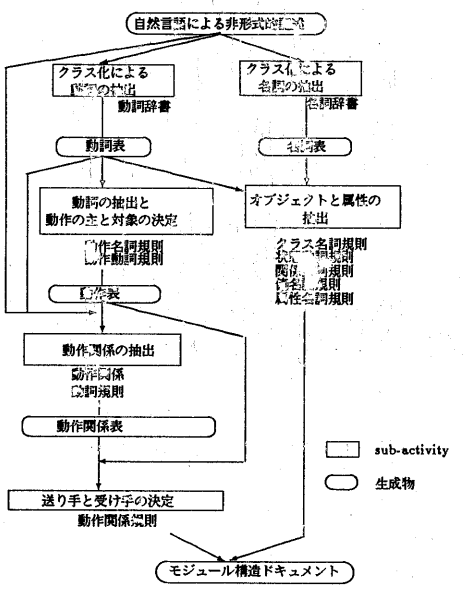

図 2: モジュール構造を得る手順

表 1: 動詞抽出カテゴリ

カテゴリ名	意味	例
関係動詞 (relational)	オブジェクト同士の関係, またはオブジェクトと属性との関係を表す	have,include,contain
状態動詞 (state)	オブジェクトの属性値の状態を表す	keep,remain
動作動詞 (action)	オブジェクトの属性値やオブジェクト間の関係の変化を表す	move,transmit
動作関係動詞 (action relational)	動作と動作の関係を表す	cause,start

表 2: 名詞抽出カテゴリ

カテゴリ名	意味	例
クラス名詞 (class)	オブジェクトやオブジェクトの集合を表す	lift,button
値名詞 (value)	オブジェクトの値や値の集合を表す	1,'c',integer
属性名詞 (attribute)	オブジェクトの属性値や属性値の集合を表す	direction(of the lift)
動作名詞 (action)	オブジェクトに行なわれる動作を表す	transmission,up-lift

```
[?class name]
  attribute
    *[[?attribute name]:[?attribute domain]]*
  relation of actions
    *1)[?action relation]*
  methods
    *[[?verb][?objestive word] *[?prepositional phrase]*
      pre-condition
        *1)[?sentence]*
      in-condition
        *1)[?sentence]*
      post-condition
        *1)[?sentence]*
      response
        *1)[?command sentence]* ]*
```

図 3: テンプレート

(3) 動作表
動詞表においてカテゴリが動作動詞の動詞を取りだし，動詞の動作主と動作対象を決定する．動作主と動作対象の決定は，主語と目的語に動作動詞の動作主・動作対象決定規則を適用することにより行なう．

(4) 動作関係表
動作表と自然言語の記述から，動作関係動詞の動作主・動作対象決定規則に従って，動作と動作の因果関係およびその種類を洗い出す．種類は，動作を行なっている時区間同士がどのように関係しているかをもとに，引継型，中途開始型，包含型，禁止型の4つに分類した．考え方は，ある動作から他の動作を呼び出すような形を包含関係とし，排他制御などは禁止型，非同期処理は途中開始型である．たとえば，
The controller deactivates the furnace by first closing the oil value and then, after 5 seconds, stopping the motor.
という文は，close 動作が終了してから5秒後に stop 動作が行われることを表しており，close 動作の終了と stop 動作の開始が時間的に離れているため，この2つの動作は引継型である．

2.2 モジュール構造ドキュメントの生成

動作表と動作関係表から，メッセージの送り手 (sender) と受け手 (receiver) を決定し，また，名詞表と動詞表からオブジェクトとその動作対象を抽出し，それらを図3に示すテンプレートにはめ込み，モジュール構造ドキュメントを作り上げる．

1回の作業では不完全なモジュール構造ドキュメントにしかならない．モジュール構造ドキュメント中のまだ埋められていないスロットが不完全な部分を示す．不完全な部分についてはさらに自然言語で記述する．この作業を refine と呼ぶ．新しく作成されたモジュール構造ドキュメントを，前のモジュール構造ドキュメントの不完全な部分と入れ換えを行なう．この作業を paste と呼ぶ．

3 支援ツールの基本方針

上記のような技法に基づいたツールで扱う文書データは，種々の構造をしている．自然言語による仕様記述では，段落づけや単語と単語の間隔なども自由であるし，表は段落づけなどは無く，単語が決まった位置に並んでいる．このような構造の異なる文書データを一括して管理し，処理しなければならない．

設計作業を進めていくには，格構造の分類を行なう動詞や名詞がどの文から抽出されたか，各表でどのように決定されたかを保存・管理しておく必要がある．そのために単語単位にポインタを持たせ，リンクを行なう必要がある．このリンクは，自然言語の仕様とモジュール構造ドキュメントの対応を明確に決定するため，のちに仕様を読んで理解する人にとって便利である．

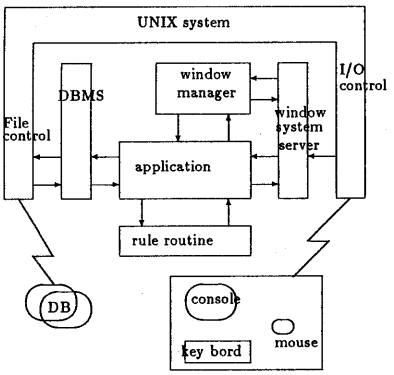

図 4: 支援ツールの構成

この単語のリンクは，設計手法の規則に従う必要がある．そのため自然言語の記述やテーブルを修正し，規則に一致しない部分が発生した場合は，修正される前のリンクや，修正前に作成された表，モジュール構造ドキュメントは削除され，新たに修正内容に従って作成される必要がある．この作業は規則に従って自動的に行なわれ，規則が判断できないような場合についてはメッセージを出力し，人間の判断を促すようにサポートすることが必要である．

さらに，ポインタによる多くのリンクを持っているため，その整合性を保証することが必要である．たとえば，1度オープンして修正を行っていたファイルに対して，リンクのはりかえのためにそのファイルをオープンした場合には，リンクのはりかえの情報は修正を行っていたファイルには反映されず，後でリンクをたどった時に矛盾が生じてしまう．

これらの要求をまとめると以下のようになる．

1. 抽出された単語の自然言語仕様中の文を参照，規則による自動的なリンクのはりかえが可能．
2. 作業中に生じる生成物（自然言語仕様も含む）に対して排他制御が可能．
3. 自然言語の仕様記述・表・モジュール構造ドキュメントの入出力と編集が可能．
4. すぐれたユーザインターフェースを持つ．

このような要求に対し，UNIX を用いて以下のように対応を行ない，図4のようなシステム構成とした．

1. 単語ごとのポインタを持つデータベース（DB）の開発
2. DB の構造に合ったエディタの開発（構造エディタ）
3. Window システムの使用

4 データベースの実現

4.1 ファイルの構造

テキストにポインタを持ち，ユーザが自由にリンクを行ない，そのリンクをもとに検索を行なうファイルシステムとしてはハイパーテキスト [10] が有名である．本支援ツールが必要としているファイルシステムも，ハイパーテキストの考え方を取り入れている．支援ツールの1つの DB ファイルは次の3つの部分から成る．

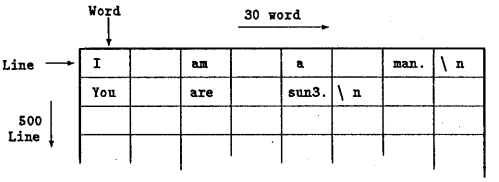

図 5: DB のテキストテーブル（自然言語の場合）

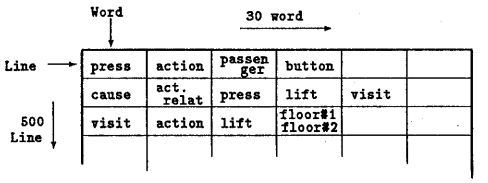

図 6: DB のテキストテーブル（表の場合）

- ヘッダ
- テキストテーブル
- ポインタテーブル

1つのファイルには，1つの自然言語の記述，または1つの表，または1つのモジュール構造ドキュメントのどれかが入る．

4.1.1 ヘッダ

ヘッダはテキストテーブルとポインタテーブルの管理を行なうため，以下の項目からなる．

1. 自分自分のファイル名（フルパスネーム）
2. テキストテーブルの使用量の情報
3. ポインタテーブルの使用量の情報
4. ファイルに対するリンクのためにポインタテーブルを指すポインタ

4.1.2 テキストテーブル

種々の構造をした文書の構造を単一の DB システムで扱うために，DB の内部では，自然言語の記述やテンプレートによるモジュール構造ドキュメントは単語とスペースに分割し，2次元の配列のイメージで管理している（図5）．また，中間生成物のテーブルはもともと単語間のスペースが無いため，2次元の配列をそのままテーブルとして使用している（図6）．この2次元の配列をテキストテーブルと呼び，1つの単語を入れる単位をセルと呼ぶ．単語間のスペースは任意個入力されるため，スペース数をテーブル中に保存している．図5のように，1つの文を横に納め，改行マークがきたら次の行ということでその下にずれる．そのため横の配列の1まとまりをテキストテーブルの行と呼ぶ．基本的に行は降順に入れられるが，行の追加や削除を行なったときには降順にはならない．行の順番を保証するために，各行はその行の上の行と下の行を指すポインタ（配列の添字）を持っている．このポインタは行間ポインタと呼ぶ．

4.1.3 ポインタテーブル

単語単位にリンクすることができるように，ポインタを管理するポインタテーブルを持っている．

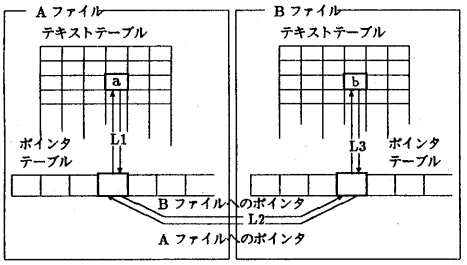

図 7: ポインタの関係

単語と単語のリンクでは他のファイルの単語とリンクすることが多いため，ポインタの情報はファイル名とポインタテーブルへのポインタから成る．このポインタを外部ポインタと呼ぶ．リンクされている単語が他の単語の追加などによりテキストテーブル上の位置（配列の添字）が変化したときに，その変化をリンクしている他のファイルに反映する必要がある．そのため外部ポインタの情報はポインタテーブルに置き，単語の位置の変化はポインタテーブルとテキストテーブルの間のポインタを変えることだけで行なうことができる．このポインタを内部ポインタと呼ぶ．ポインタテーブルの内部ポインタはテキストテーブルの内部ポインタとは違い，テキストテーブルのセルを範囲指定できる．そのため，リンクするときに単語や行の範囲指定ができる．ポインタテーブルは以下の項目から成る．

- テキストテーブルの単語または行の範囲またはヘッダを示す内部ポインタ
- 他のファイルの単語を示すための外部ポインタ（ファイル名とポインタテーブルへのポインタ）

4.2 リンクの実現

単語をリンクするときには，他のファイルとリンクすることがほとんどである．そのためポインタテーブルのセルには外部ポインタの情報として図7のように，相手のファイル名とポインタテーブルのセルの位置（配列の添字）を持っている．この図7では，A ファイルの単語 a と B ファイルの単語 b をリンクした場合である．まず，A ファイルのテキストテーブルの単語 a のセルとポインタテーブルのセルをリンクする（L1）．次に A ファイルのポインタテーブルのセルと B ファイルのポインタテーブルのセルをリンクする（L2）．そして B ファイルのポインタテーブルのセルとテキストテーブルのセルをリンクすることで，A ファイルと B ファイルのリンクが出来る．リンクは双方向性を持っているので，どちらのファイルからでもリンクをたどることができる．この場合は単語と単語のリンクであるが，行やファイルの単位でも同様に行うことが可能である．このようにテキストテーブルのセルだけでなく，行，それにヘッダにポインタテーブルを指す内部ポインタを持っている．テキストテーブルが修正され，リンクされている単語の位置が変更されたときには，ポインタテーブルとテキストテーブルの間の内部ポインタは随時変更される．

テキストテーブルの内部ポインタは，常にポインタテーブルのどこかを指している訳ではない．単語，行，ファイルがどこにもリンクしていないときは特に意味はない．ポインタテーブルの内部ポインタも同様である．

4.3 排他制御

リンクを行なう操作は，DBMS の関数として与えられている．この操作はファイルをオープンし，中身を書き換え，クローズする処理を行なうため，ある Window ですでにオープンしているファイルを，さらに別の Window でオープンしてしまう可能性がある．この場合，ファイルの整合性が保てなくなるため，排他制御を行なう必要がある．本支援ツールで行なった排他制御は，図8のような表を使って行なった．この表は，ツールの中に外部変数の構造体として定義される．オープンしたファイルはこの表に，ファイル名，ファイル記述子，使用している領域のアドレス，利用数を登録する．次にオープンするときにはファイル名で表を検索し，同じファイルがあればオープン処理は行なわずに，利用数を+1 し，表の情報を使って同じ領域を使用する．クローズのときは利用数を-1 し，利用数が 0 になればクローズ処理を行ない，そうでなければ行なわない．このような排他制御を行ない，ファイルの整合性を保つ．

ファイル名	abc	calc	....	
ファイル記述子	2	3	....	
領域アドレス	xxxx	xxxx	....	
利用数	1	3	....	

図 8: ファイルの排他制御表

4.4 ディレクトリの構造

4.4.1 ファイルの命名規約

この支援ツールで使用するファイルは 8 種類ある．これは自然言語仕様記述，表，モジュール構造ドキュメントなどの生成物を納めるためのファイルである．設計手法の規則により動的にリンクをはりかえるため，表3のようにファイルには種類に応じて規則的なファイル名が付けられる．ファイル名はどれも先頭に種類を示す記号があり，"_" の後に任意の名前がくる．(表中の *nn..n*) この DB のそれぞれのファイルは，UNIX 上のファイル管理に従って管理されている．

表 3: ファイルの命名規約

ファイルの種類	内容	命名規則
自然言語記述	自然言語による仕様の記述	na_*nn..nn*
動詞テーブル	自然言語より抽出した動詞のテーブル	vt_*nn..nn*
名詞テーブル	自然言語より抽出した名詞のテーブル	nt_*nn..nn*
動作動詞テーブル	動詞の動作を明示するテーブル	at_*nn..nn*
動作関連テーブル	動作の関連を明示するテーブル	ar_*nn..nn*
クラスファイル	クラスごとのモジュール構造ドキュメント	c_*nn..nn*
paste ファイル	不明確だった部分が置き換えられたモジュール構造ドキュメント	p_*nn..nn*
refine ファイル	モジュール構造ドキュメントの不明確な部分を明確にするために記述した自然言語の仕様記述	r_*nn..nn*

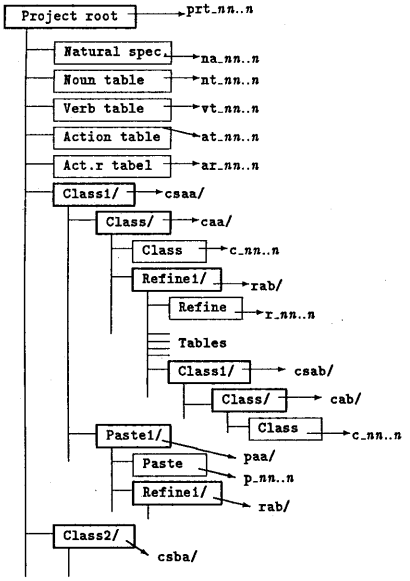

図 9: ファイルの階層構造

4.4.2 ファイルの階層構造

クラスファイルはソフトウェアコンポーネントのクラスに対応したモジュール構造ドキュメントである．クラスは，1回のモジュール構造ドキュメントの生成で複数個得られる．このクラスファイルからさらにrefineファイルが付け足され，各表，クラスファイルと増殖的に作られる．このようなファイルを1つのディレクトリで管理したのでは，たくさん作られたクラスファイルや表のレベル（どの自然言語の記述に対するものか）をユーザが混乱したり，格構造によりリンクを動的にはりかえる時に，どのテーブルが現在処理対象としているテーブルか分からなくなる．そのため対応策としてレベルごとにディレクトリを作り，管理を行なうことが考えられる．本ツールでは図9のようにディレクトリを作成している．矢印で示された文字列はディレクトリまたはファイル名の実際の名前である．

基になるディレクトリをプロジェクトルートと呼ぶ．このディレクトリの下に自然言語による記述と各表，クラスファイルとpasteファイルを管理するためのディレクトリ（csaa/）がくる．クラスファイルとpasteファイルを管理するディレクトリの下にはクラスファイルのディレクトリ（caa/）とpasteファイルのディレクトリ（paa/）がくる．pasteファイルは，クラスファイルの記述の不完全な部分を詳細化したクラスファイルに入れ換えたものなので，クラスファイルの一部と考える事ができる．そのためpasteファイルを管理するディレクトリpaa/はcsaa/の下にある．クラスファイル（c_nn..n）やpasteファイル（p_nn..n）はそれぞれを管理するディレクトリ（caa/，paa/）の下にある．たとえば，caa/c_nn..nとcab/c_nn..nのpasteを行なった場合，pasteファイルはpaa/p_nn..nである．

クラスファイルやpasteファイルの記述の不完全な部分を補うために自然言語で記述されたrefineファイルはそれぞれのクラスファイル，pasteファイルのディレクトリ（caa/，paa/）にrefineのディレクトリ（rab/）を作り，その下に置かれる．このrefineファイル（r_nn..n）があらたに自然言語仕様記述となり，操作が繰り返される．

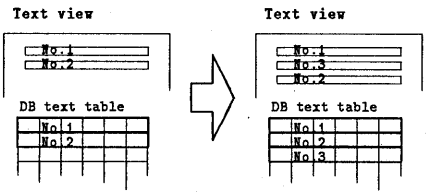

図 10: 表示される行の順序とDB内の行の順序の違い

4.4.3 ディレクトリの命名規約

ディレクトリは各ファイルが作成されたときに必要に応じて自動的に作成される．クラスファイルは同じ自然言語記述から複数個できるため，階層の番号づけと同じ階層内での番号づけが必要である．ディレクトリの命名規約は表4の通りである．nは同じ階層内の順番を示している．またxxはa～zまでの任意の文字であり，はじめの桁は同じ階層内のディレクトリの連番であり，次の桁はディレクトリの階層を示す．(図9参照)

表 4: ディレクトリの命名規約

ディレクトリの種類	内容	命名規則
classn /	クラスファイルとそのクラスのpasteファイルを管理する．	csxx
class/	クラスファイルを管理する．	cxx
refinen /	refineファイルを管理する．	rxx
pasten /	pasteファイルを管理する．	pxx

5 エディタとWindow

5.1 構造エディタ

文章を入力・編集するためのエディタを作成する上で問題となるものは以下のような2つである．

1. DBのデータはテキストの形になっていないため，表示しているカーソルの位置とDB内のデータの位置を合わせる必要がある．
2. DBに対して，行の追加・削除を行なうと行の並びが降順でなくなる．(図10)

そこで単語を単位とした構造エディタを作成した．カーソルは単語ごとに移動し，削除，追加も単語単位で行なう．エディタにはカーソル移動モードと入力モードがある．通常はカーソル移動モードであり，単語単位のカーソルの移動，単語の削除が行なえる．単語を入力したいときにはスペースキーを押すことにより入力モードに変わり，1文字単位の入力・削除が行なえる．この入力モードでは複数の単語を入力してもよく，通常のテキストの入力と同様に行なわれる．一度入力した単語のつづりを修正する場合も，同様にして単語を編集することができる．表示とDBの内容のカーソル位置の一致は，エディタがセルに行の先頭からの位置を保持させ，それにより単語の位置を随時計算し，一致を図っている．

表示している行とDB内の行の対応は，表示上の行番号とDB内の行のポインタ(配列の添字)を表として持つことにより行なっている．(図11) この表はエディタ内に内部変数の構造体として定義される．このような表で管理することにより，行の削除，追加やスクロールも表のポインタを移動させるだけで行なえる．

通常，文は単語とスペースが交互に現れると考えられる．そのため単語の直後に新たな単語を追加した場合は，自動的にスペースが

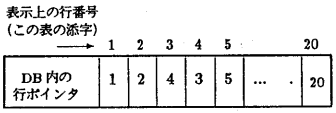

図 11: 表示にための行の対応表

追加される．また，他の単語や行，ファイルへのリンクを持っている単語・行を削除した場合，警告メッセージを表示し判断を促すモードと，無条件に削除を行なうモードの２つを選ぶことできる．当然，この場合は削除された部分以外のリンクはそのまま保持される．

5.2 Window

本支援ツールはユーザインターフェースを向上させるために X-Window システムの上に構築されている．Window 上でのコマンドは以下のようである．

1. ファイルのオープン・クローズ
2. Window のオープン・クローズ
3. Window の移動・アイコン化
4. ポップアップメニューの対応
5. Window 間の単語のカット・アンド・ペースト
6. マウスによるポインタリング
7. 格構造の分類を行なうための規則の起動．

操作はポップアップメニューにより行なう．

6 例題

例題として lift 問題 [7] を用い，その自然言語仕様記述からモジュール構造を得ることを試みた．図 12はエディタの入力モードで例題の入力を行っているところである．この文章から作成した各テーブルを示す (図 13)．図 14は自然言語記述のなかで指定した button が動詞テーブル上で反転表示されている．図 15は図 13から得たモジュール構造ドキュメントの一部である．不完全な部分が多く残っているため，さらに自然言語記述による詳細化が必要である．

7 評価

問題点を以下のようにまとめた．

1. ファイルのオープン処理に時間がかかる．
2. 図 14のように，あるファイルで指定した単語を，ポインタをたどってリンクされた別のファイルの単語を表示するのに時間がかかる．

上述のような問題点はあるものの，自然言語仕様からモジュール構造を得る手法をもとに支援ツールを作成し，実際の使用に十分耐えるものであることを確信した．

8 今後の課題

まず，評価での問題点を解決し運用経験を積むことが必要である．また，本支援ツールの形式的仕様記述は TELL の記述言語としているが，この TELL の記述言語は図による記述も行えるため，形式的な記述の図的表現もサポートするようにし，より人間が記述しやすいようにすることも必要である．将来的には，自然言語からの形式的記述への手法を，中間生成物 (表等) と関係規則を与えることにより，その手法で作業が行えるようにしたい．

謝辞

本研究の御支援，御援助いただいた富士通株式会社 国際情報社会科学研究所 蓬莱尚幸氏に感謝いたします．

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．情報処理学会誌，第 28 巻 第 7 号，1987．

[2] 田村 他．ハイパーテキストを用いた設計支援プロセスツールの試作．情報処理学会ソフトウェア工学研究会，68，1989．

[3] 高田 広章．ハイパテキストとそのプログラミング環境への応用．情報処理学会ソフトウェア工学研究会，64，1989．

[4] 榎本 他．ソフトウェア開発システム（TELL）における自然言語による使用記述言語（NSL）とその応用例情報処理学会ソフトウェア工学研究会，34-14，1984

[5] 佐伯 他．自然言語仕様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 第 30 巻 第 11 号，1989．

[6] IEEE Softwarer．March 1988, Vol. 5 No. 2．

[7] Problem Set for the Fourth International Workshop on Software Specification and Design. *Proc. of 4th International Workshop on Software Specification and Design*, 1987, pp. ix–x.

[8] JEFF CONKLIN and MICHAEL L. BEGEMAN. gIBIS: A Hypertext Tool for Exploratiory Policy Discussion. *acm Transactions on Office Information Systems*, October 1988 Vol. 6 No. 4 303–420.

[9] Colin Potts and Glenn Bruns. Recording the Reasons for Design Decisions. In *Proc. of 10th ICSE*, pp. 418–427 1988.

[10] Bush,V. As we may think. In *Atrantic Monthly 176*, 1 pp. 101–108 (July 1945)

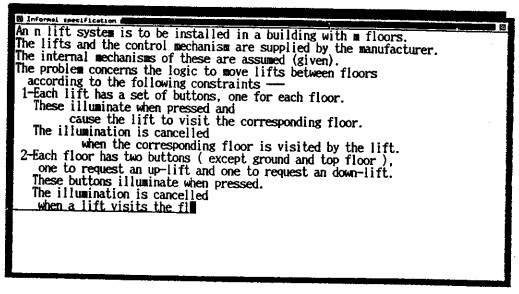

図 12: 例題の入力

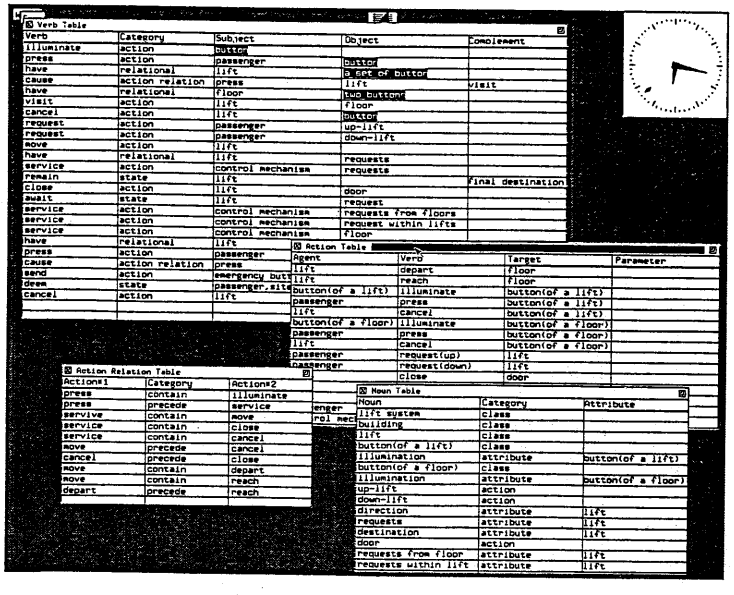

図 13: 例題のテーブル

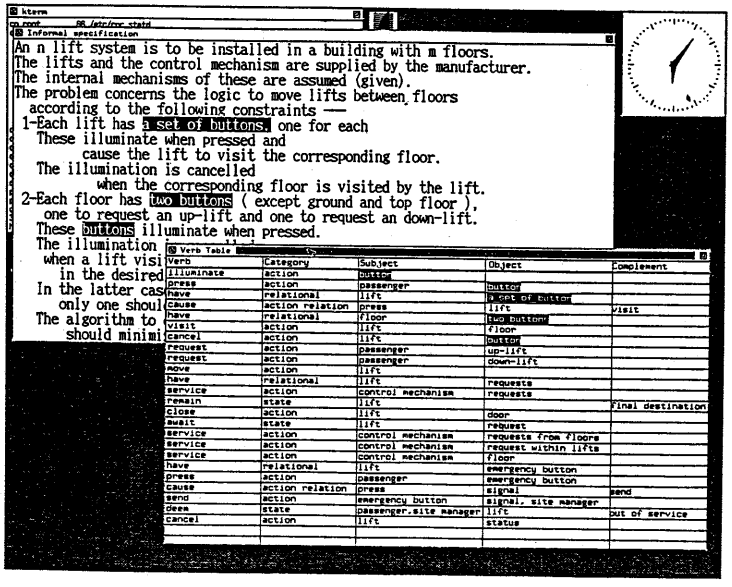

図 14: 自然言語記述と動詞テーブル

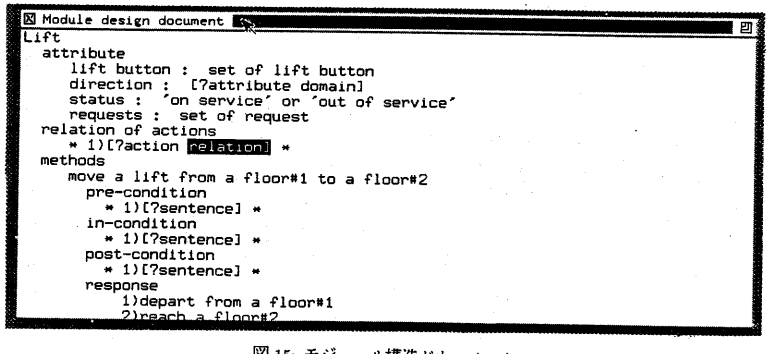

図 15: モジュール構造ドキュメント