

ソフトウェアツール作成支援システム：ISMOS

中田 修二 山崎 剛
日本電気(株) ソフトウェア生産技術研究所

1. はじめに

ソフトウェアツールの一つの分野としてソフトウェアに関する各種の仕様やプログラムに関する情報をデータベース化して解析するツールがある。一例としてPSL/PSA[7]を挙げる事ができる。

本報告で述べるISM

OSは、このようなデータベースを利用したソフトウェアツール(以下ではDB指向ツール、または単にツールと呼ぶ)の生産性、保守性を向上させるための、作成支援システムである。

本報告の2章～6章ではDB指向ツールの実現方法のレビュー、DB指向ツールの生成方法、ISMOSの概要等を述べる。7章～9章ではISMOSを利用したツール生成の具体例を紹介し、その実験結果を考察する。

2. DB指向ツール

本章では、3章以降へ進む前準備として、現状でのDB指向ツールの実現方法をレビューする。図1にDB指向ツールの一般的な構成と、現状でのインプリメントの手段を示す。

DB指向ツールは、機能の面から見て5つの部分から構成されると考えられる。各部分の実現方法としては、DBMSの部分は既存のファイルシステムや汎用DBMSが利用できる。またデータベースの解析の一部やレポート出力の部分も、DBMSが提供している問い合わせ言語やレポート作成ユーティリティなどを使用してかなり実現できる。一方、入力データ処理やデータベース格納の部分につい

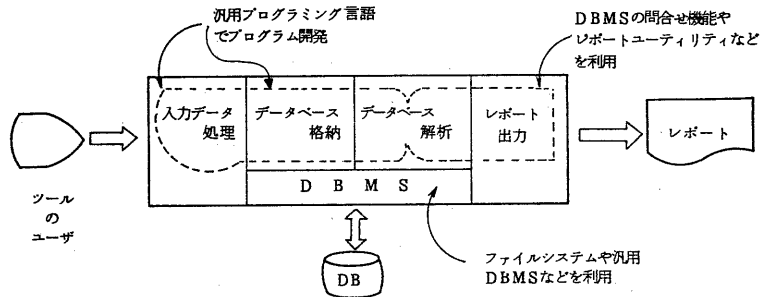


図1 DB指向ツールの構成と実現方法の現状

ては適当なツールがないために、プログラミング言語を用いてプログラムを作成して実現しているのが、現状の大半と思える。この理由としては、これらの部分は、各ツールで扱う問題に固有の言語、固有のスキーマを持つデータベースへのデータの格納などを処理するため、多くのツールを汎用的に実現するアプローチにはそぐわないと考えられているためではないかと思われる。この結果、現状ではDB指向ツールには、少なからぬソフトウェアの開発、保守のコストと期間が必要とされる場合が多いと判断される。

3. DB指向ツールの生成方法

3.1 モデル・ベースト アプローチ

本報告で述べるモデル・ベースト アプローチ(Model-based approach)は、DB指向ツールをより容易に実現するための一つのアプローチである。このアプローチでは実現目標のツールを形式的に定義し、この定義を用いてDB指向ツールを実現する。ここで言うツールの定義とは、ツール固有の言語(i.e., ユーザインタフェース)と、ツールが扱うべき固有の情報構造(i.e., ツールが対象とする問題を構成する情報とその相互

関係などで、最終的にはDB指向ツールのデータベースの構造に反映される)の二つである。以下ではツール固有の言語をユーザ言語と呼び、ツール固有の情報構造をユーザモデルと呼んでいる。したがってモデル・ベースト アプローチとはユーザ言語とユーザモデルを形式的に定義する事により、この定義を用いてDB指向ツールを実現するアプローチであるという事ができる。

モデル・ベースト アプローチの主要素として下記の3つの要素がある。括弧の中の名前は、それぞれ我々の試作におけるメタモデル名、定義言語名、システム名である。

- ・ユーザモデルを表わすために使用する高位のモデル。メタモデルと呼ばれる事もある。(ISMOM)
- ・ユーザモデルとユーザ言語を形式的に定義するための定義言語。(ISMOL)
- ・DB指向ツールの作成支援システム。(ISMOS*)

すなわちISMOMはユーザモデルを作成する時に枠組として使用するメタモデルである。個々のDB指向ツールのユーザモデルはISMOMで表わした1つの特別な場合(i.e., ISMOMの実現値の1つ)としてとらえられる。現在のISMOMは実体関連モデル(Entity-Relationship Model)[5]を拡張、変更したものである。ISMOLはISMOMを使って作成したユーザモデルとユーザ言語の定義を記述するための言語である。これらについては文献[1]において述べたので、本資料ではISMOSを中心に報告する事とする。

3.2 ISMOSでのツール生成の手順

ISMOSを用いてDB指向ツールを生成する手順を図2に示し、以下に記す。

- 1) ユーザの要求を分析し、ユーザモデルを明らかにする。この時ISMOMを用いる。またユーザ言語に対する要求も明らかにする。この段階ではユーザが自分の要求を完全に述べる事ができない場合も多いので、必要に応じて何通りかのユーザモデルやユーザ言語を用意して、以下のステップへ進む事もありうる。
- 2) ユーザモデルとユーザ言語をISMOLで定義する。この定義仕様をISMOL仕様、またはI-仕様と呼んでいる。
- 3) I-仕様をISMOSへ入力すれば、希望したDB指向ツールが得られる。

すなわちISMOSを用いる事により、ユーザは実現したいツールの外部仕様を形式的に定義すれば、ツールの内部を構成するプログラム群を生成できる。図1の入力データ処理やデータベース格納の部分の様に、従来はソフトウェア開発作業を必要としていた部分も生成されるので、DB指向ツールの開発・保守のコストや期間をより小さくできる。

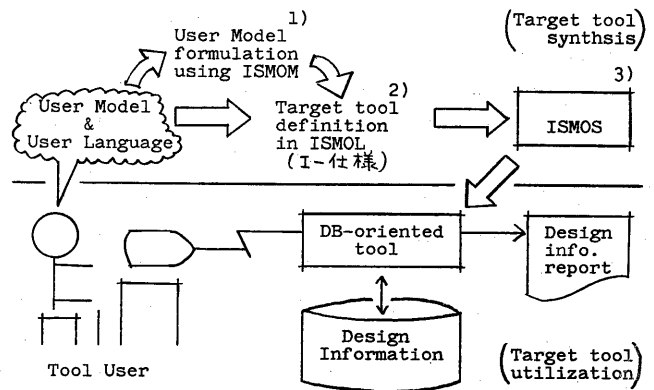


図2 ISMOSでのツール生成の手順

* ISMOS: Information Systems Semantic MOdelling System

4. ISMOSの構成と機能

ISMOSは表1に示すように2つのサブシステムから構成される。「モデル&言語定義(MOLD)」サブシステムはツールの定義のために使用される。「ツール生成(TGEN)」サブシステムは定義されたツールを生成するために使用される。表1の左側に各サブシステムの主な機能を、右側にその主な構成要素を示す。

を生成する。ユーザ言語コンパイラ・コンパイラはユーザ言語の構文チェックを生成する。ドキュメント生成ユーティリティはユーザ用の言語マニュアルやその他の資料を作成する。さらにTGENサブシステムはツールで使用するデータベースの用意など、ユーザがツールを即時に利用できるような環境をととのえる。

4.1 MOLDサブシステム

MOLDサブシステムの目的は、ツールの作成者がツールを定義するのを支援し、ツール生成のための準備を行なう事にある。

ツールの生成にあたって、ツール作成者がイー仕様を記述すると、イー仕様構文チェックを使用して構文誤りをチェックできる。次に正しいイー仕様をイー仕様プロセッサに入力する。イー仕様プロセッサはイー仕様の内容を、ツール生成サブシステムが使用し易い形に変換して「ツール定義情報データベース(MOLDB)」へ登録する(図3)。これはツール生成のための準備である。

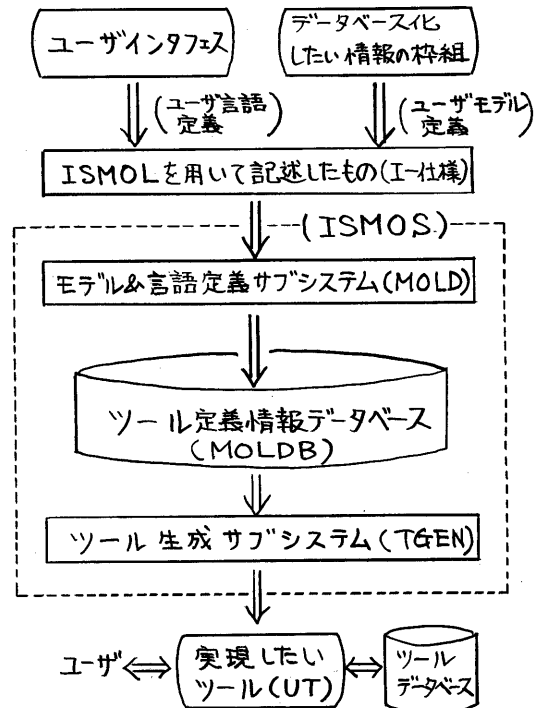


図3 ツール生成の過程

4.2 TGENサブシステム

TGENサブシステムの目的は、MOLDBの中のツール定義情報を利用して、目的とするツールを構成するプログラム

群を生成し、かつユーザのために、生成したツールを利用できる様に環境をととのえる事にある。

ツールの作成者はTGENサブシステムを用いて、通常2つのプログラムを生成する。ユーザシステム・ジェネレータはユーザが要求したツールそのもの

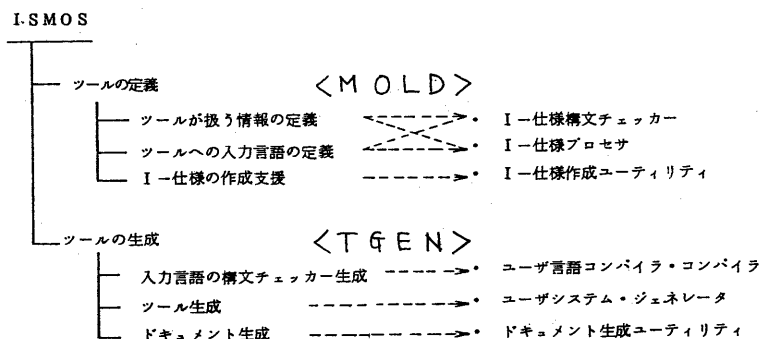


表1 ISMOSの主な機能と構成要素

5. 生成されるDB指向ツールの機能

IS MOSの機能と共に、生成されるDB指向ツールの持つ機能を図4に示す。ツールを利用する際に、ユーザはユーザ言語の構文チェッカを利用して入力データの構文誤りを発見させ、誤りをなくしてからツールへ入力して、情報のデータベース化を行なえる。データベース化した情報に関して各種のレポートを出力したり、データベースの問い合わせを実行できる。

ツールの機能とプログラム生成との関係を以下に述べる。入力データのチェックやデータベース化に関しては、ユーザ様で定義された通りにすべてプログラム生成される。一方、データベースの検索やレポートの生成については、標準的なもの（たとえば一覧表、相互参照表、一定の関連をつぎつぎたどる展開表など）

については生成される。一方、特定のツールに固有の特殊な処理ロジック、データの出力形式が必要なものについては生成できないので、個別に作成しなければならない。しかし一度作成すれば、それは部品として他のツールでも利用できる場合が多い。

6. IS DOSのシステムとの比較

ミシガン大学のIS DOSプロジェクトでは、モデル・ベーストアプローチによるIS L D SとS E Mというシステムが稼動している〔8〕。このシステムと本報告のIS MOSとではいくつかの相異点があるが、基本的な相異はIS DOSではツールの定義情報をS E Mという汎用のインタプリタに実行時点で解釈させてツールとして働かせているのに対し、IS MOSではT E Nサブシステムを用いてツール毎に専用のプログラムを生成して、これをユーザが利用する点である。

7. IS MOSによるツールの試作例

7.1 T A X I S

IS MOSの適用実験の一つとして、トロント大学で現在開発中のT A X I S〔4,6〕と呼ばれる会話型情報システムの設計方法論のためのツールを生成してみた。我々がT A X I SをIS MOSの適用実験の中に加えた理由は、T A X I Sの設計方法論の基礎となっているモデル（拡張パトリ・ネット）が明確な点、そのモデルを記述する言語が用意されていること、T A X I Sの応用例が資料にもり込まれている事、会話型情報システムの設計の重要性などによる。

まず最初に行なうべき事はT A X I Sにもり込まれている基本的な概念を把握する事である。一般に会話型情報システムではユーザとアプリケーションシステムとは端末を通じて相互に会話を行ない

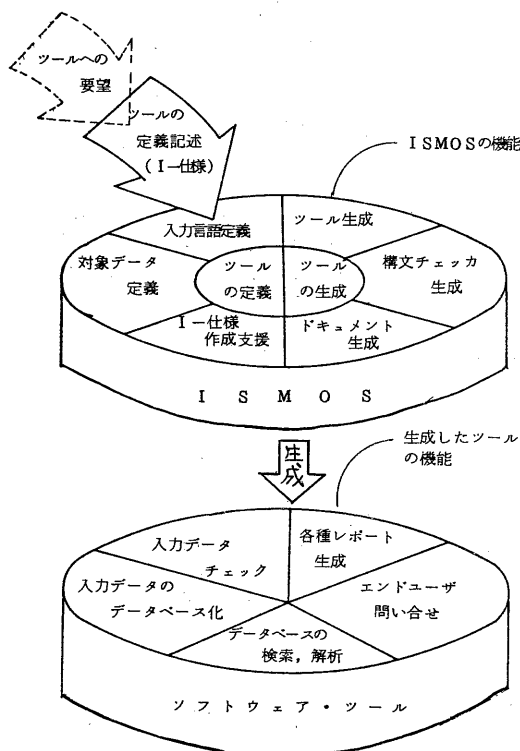


図4 IS MOSと生成したツールの機能

処理を進める。この時ユーザからの指示によりアプリケーションシステムは、システムやデータベースの状態(state)を調べ、条件(condition)に応じたいくつかの動作(action)を起こして、システムを次の状態へ移行させる処理(transaction)を実行する。この後システムはユーザからの次の指示を待ち受ける。TAXISでも、状態、条件、動作、ある状態から次の状態への移行の処理という4つの基本概念を用いて会話型情報システムを表わしている。これらの基本概念の関連を(Entity-Relationship図式を用いて)表わしたのが図5である。

図6はTAXISの記述言語で書かれた仕様の一例で、航空券支払い処理の中の一つのtransactionを表わしている。航空券購入者が支払いを行ない、その支払い額が記録された状態(payment-recorded)になると、partial-paymentと名付けられたこのtransactionが起動される。このtransactionでは、まず現在までの支払い額が航空運賃に満たないかどうかの条件をチェックし、満たない場合にはさらに支払いを促進するメッセージを出力する動作を行なう。その後システムは一部代金の支払い済みの状態(partial-payment-made)へ移行する事が示されている。

ISMOSを用いてTAXISのためのツールを生成するためには、図5で表わされたTAXISの基本概念を、図6で例示したTAXISの言語の形式で受け付けられる様にしなければならない。そこでツールの基本概念とその言語の形式をISMOLを用いて定義する(図7)。このエー仕様をISMOSへ入力するこ

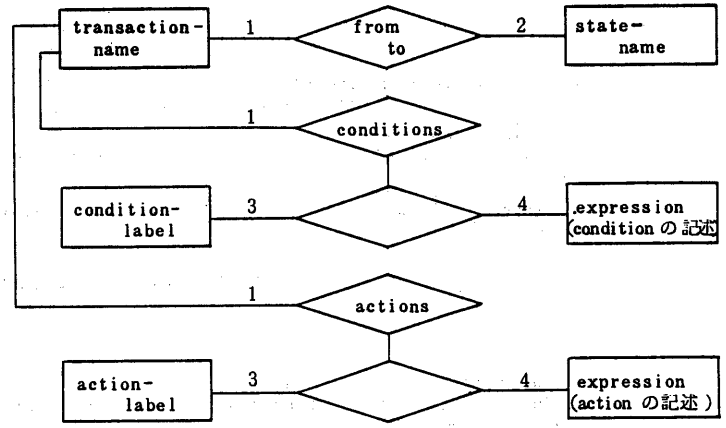


図5 会話型情報システムを表わす基本概念の関連

とによりTAXISのためのツールを生成できる。

ISMOSで生成したTAXIS用ツールを利用した時の、図6に相当するtransactionの仕様を図8に示す。図6と図8を比べれば書き方の詳細が少し異なるが、実用上は問題とならないと判断される。(ツールの定義で、TAXIS本来の言語仕様に完全に一致させる事が理想であるが、現在の試作版のISMOSでは少し制限があり、この様に似て非なる言語仕様となった。)

上記のTAXIS用ツールを生成するために必要とした所要工数は、TAXISの理解と分析に5人日、エー仕様作成に4人時であった。ただし、ここではエー仕様のかき方には十分習熟している事を前提としている。

7.2 Cプログラム・アナライザ

Cプログラム・アナライザ(CPA)と名付けたツールをISMOSを用いて作成した。CPAは、C言語のソースプログラムを入力として、プログラムの解説、保守のための資料を出力するユーティリティである。

図9にCPAの構成と、ISMOSによるCPAの生成との関係を示した。ISMOSで生成する部分は、CPAの中

```

partial-payment:
  from f1: payment-recorded:
  to t1: partial-payment-made:
  conditions
    partial-payment?:
      (payment-to-date < (f#,date).flight.ticket-cost);
  actions
    ask-for-more-payment:
      s-request(travel-agent.code,payment-amt,payment-method);
end partial-payment;

```

図 6 TAXIS 言語による transaction の記述例 [4]

```

ISMOL-SPECIFICATION taxis_model_by_ismol ;
  USER-LANGUAGE taxis_language ;
  USER-SYSTEM taxis_system ;
ENTITY state_name ;
ENTITY transaction_name ;
ENTITY condition_label ;
ENTITY action_label ;
ATTRIBUTE expression ;
  VALUES NAME-CONSTANT ;
SEMANTIC-UNIT from_to_su ;
  COMBINATION P2 1E=transaction_name
                  2E=state_name ;
  CODE-B 01 from
           01 to ;
  CODE-C 01 ---- ;
SEMANTIC-UNIT condition_su ;
  COMBINATION P4 1E=transaction_name
                  3E=condition_label
                  4A=expression ;
  CODE-B 01 conditions ;
  CODE-C 01 ---- ;
SEMANTIC-UNIT action_su ;
  COMBINATION P4 1E=transaction_name
                  3E=action_label
                  4A=expression ;
  CODE-B 01 actions ;
  CODE-C 01 ---- ;
STATEMENT from_to_stm ;
  USED-FOR from_to_su ;
  SUBJECT transaction_name <<1E>> ;
  PREDICATE {fromto} (<<2E>>: ) ;
STATEMENT condition_stm ;
  USED-FOR condition_su ;
  SUBJECT transaction_name <<1E>> ;
  PREDICATE conditions (<<3E>> ##is##
                       <<4A>> ##end##: ) ;
STATEMENT action_stm ;
  USED-FOR action_su ;
  SUBJECT transaction_name <<1E>> ;
  PREDICATE actions (<<3E>> ##is##
                    <<4A>> ##end##: ) ;

```

生成するツールをユーザが利用するとき
のユーザ用言語の名前の定義

生成するツールの名前の定義

生成すべきツールの基本概念の定義

基本概念間の関連の定義

生成したツールをユーザが利用するとき
のユーザ用言語の入力形式の定義

図 7 ISMOL による TAXIS ツールの定義 (主要部分のみ表示)

```

transaction_name partial_payment ;
  from payment_recorded ;
  to partial_payment_made ;
  conditions
    partial_payment is
      'payment_to_date < (f#,date).flight.ticket-cost' end ;
  actions
    ask_for_more_payment is
      's-request(travel-agent.code,payment_amt,payment_mothod)' end ;

```

図 8 ISMOS を利用して生成した TAXIS ツールに設計情報を入力する場合の記述例

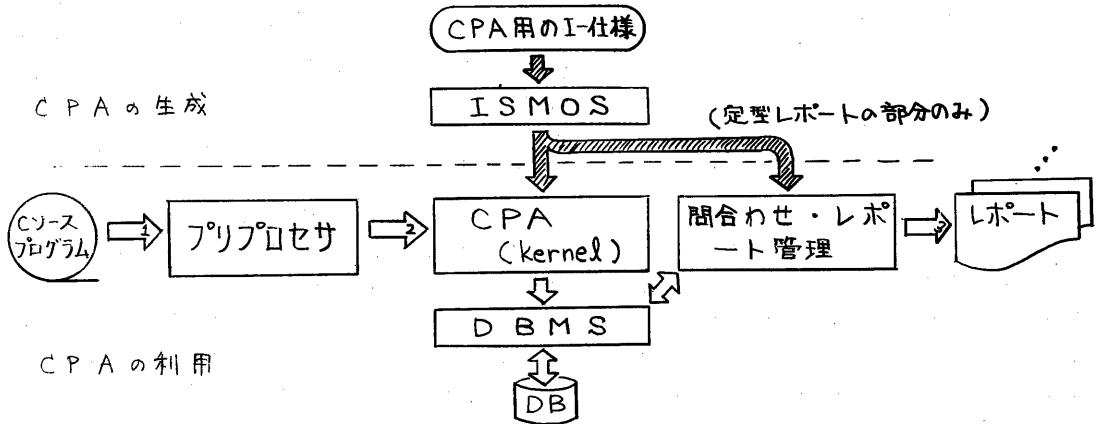


図9 Cプログラム・アナライザの構成

```

1  /***** */
2  /*      */
3  /*      mfile      */
4  /*      */
5  /***** */
6
15 main(argc, argv)
16 int argc;
17 char *argv[ ];
18 {
19     int    fpl;
20     float  fp2;
21
27     subfunc11(fp1, fp2)
28
55 }

```

```

/***** */
/* The extracted */
/* info. for mfile */
/***** */
fil mfile;

has func main
f_def_inf
formal_arg argc argv at 15;

has var fpl
var_inf data class is 'auto'
data_type_is 'int' at 19;

SUBJECT func main;

call subfunc11
call_inf act_arg fpl fp2
at 27;

```

図10 Cソースファイル: mfile

図12 mfile から抽出した情報

```

1  /***** */
2  /*      */
3  /*      subfile1    */
4  /*      */
5  /***** */
6
13 subfunc11(p1, p2)
14 char p1;
15 float p2;
16 {
17
20     comsub1(p1);
21
28 }
31 subfunc12()
32 {

```

```

/***** */
/* The extracted */
/* inf. for subfile1 */
/***** */
fil subfile1;

has func subfunc11
f_def_inf
formal_arg p1 p2
at 13;

has var p1
var_inf
data_class is 'auto'
data_type_is 'char'
at 14;

has func subfunc12
f_def_inf at 31;

```

図11 Cソースファイル: subfile1

図13 subfile1 から抽出した情報

ファイル名	関数名	仮引数 変数
mfile	main	argc
		argv
		15
subfile1	subfunc11	p1
		p2
		13
	subfunc12	31
subfile2	subfunc2	formp

図14 定義された関数の一覧表

```

/*-----*/
variable fpl is defined in file mfile
at statement number : 19
data class is '\auto\'
data_type or def_id is '\int\'
/*-----*/
In function : main fpl is used as
actual argument
to call subfunc11 at 27
actual argument fpl is passed to
formal argument p1
/*-----*/
In file : subfile1 subfunc11 is defined
with formal argument p1
at statement number : 13
/*-----*/
variable p1 is defined in file subfile1
at statement number : 14
data_class is '\auto\'
data_type or def_id is '\char\'
!!! WARNING : p1 has different datatype
compared with given variable : '\int\'
/*-----*/
In function : subfunc11 p1 is used as
actual argument
call comsub1 at

```

図15 関数の呼び出しの入れ子における変引数と仮引数のタイプチェックレポート

核部と問合せ・レポート管理の中の定型レポート出力部分である。CPAのイー仕様ではCPAの中核部へのデータ入力のインタフェース(図9の□⇒)の定義とデータベースへ格納する情報の定義がされている。

プリプロセサはこのソースプログラムから各種の情報を抽出し、イー仕様で定義された中核部へのデータ入力の形式に変換して出力する。既存のプログラミング言語のような、処理の手続きの記述が主体でE-Rモデルに基づかない言語をツールで扱うためには、現在の所このようなプリプロセサが必要である。

図10から図15まではCPAへの入力データと出力レポートの一例である。

図10、図11に示したようなC言語のソースプログラムをプリプロセサに入力すると、プリプロセサは図12、図13に示したような出力を行なう。この出力は図9の□⇒に該当するデータである。

問合せ・レポート管理を用いて色々なレポートを得る事ができる。図14は定型レポートの1つで、各ファイル中に定義されている関数とその仮引数、定義の記述のある行番号を一覧表として表示している。

一方、図15は非定型レポートの1つであり、関数の呼び出しのいれ子の構造をたどって、最初の仮引数のデータ型とつぎつぎと叫ばれる各関数の仮引数のデータ型との一致をチェックするためのものである。この様なレポートは特別のロジックに基づいているのでISMOSは自動生成しない。ユーザが作成したルーチンを問合せ・レポート管理へ組み込んで利用できる。

8. ツール生成の実験結果

ISMOSによるツール生成の実験結果について述べる。表2は7章で述べたTAXISとCPA、それに交換システムのソフトウェアに対する要求仕様定義

ツール: RLP[3]の3個のツールをISMOSを用いてそれぞれ生成した結果である。

たとえばTAXISの場合には、179行のイー仕様から、2351行のC言語のソースプログラムを、ツールとして生成している。このソースプログラムの生成とコンパイルは、約1MIPSの処理能力を持つコンピュータで、96.7秒のCPU時間(TSSの通常の利用環境で平均約6分程度の経過時間)で実行できた。

図16は、種々のツールのイー仕様サイズと、そのイー仕様をISMOSに入力して生成したツールのオブジェクトプログラムのサイズとの関係を表わしたものである。ツールのオブジェクトプログラムのサイズの計算式

$$T_s = 1.4 I_s + 33$$

により、イー仕様記述量から、生成されるツールのオブジェクトサイズを予想する事が可能である。また、イー仕様サイズが変の時のツールのオブジェクトサイズ(33KB)は、ツールのオブジェクトの中でイー仕様に依存しない部分をたし合わせたサイズである。

9. ISMOSの利用による利点

ISMOSを用いて種々のツールを生成した経験により、モデル・ベーストアプローチによるツール生成の利点のいくつかが確認された。以下にその利点を列挙する。

生産性の向上

ツール作成のための所要時間、所要工数を大幅に短縮できる。すなわちツール定義者はイー仕様を作成すれば、ISMOSを用いて目的とするツールを自動生成することができる。

保守性の向上

生成されたツールの機能変更柔軟に

生成されたツール		RLP ^[3]	CPA	TAXIS
機能		要求定義	Cプログラムアナライサ	会話型情報システム設計
I-仕様サイズ ¹ (行)		90	403	179
ユーザ言語構文キタ	生成時間 (CPU秒)	50	70	55
	Cソース (行)	983	1086	1011
	プログラムサイズ ² (Kバイト)	15	17	17
ユーザツール	生成時間 (CPU秒)	81	135	97
	Cソース (行)	2271	2524	2351
	プログラムサイズ ² (Kバイト)	39	49	43

表2 ツール生成の実験データ

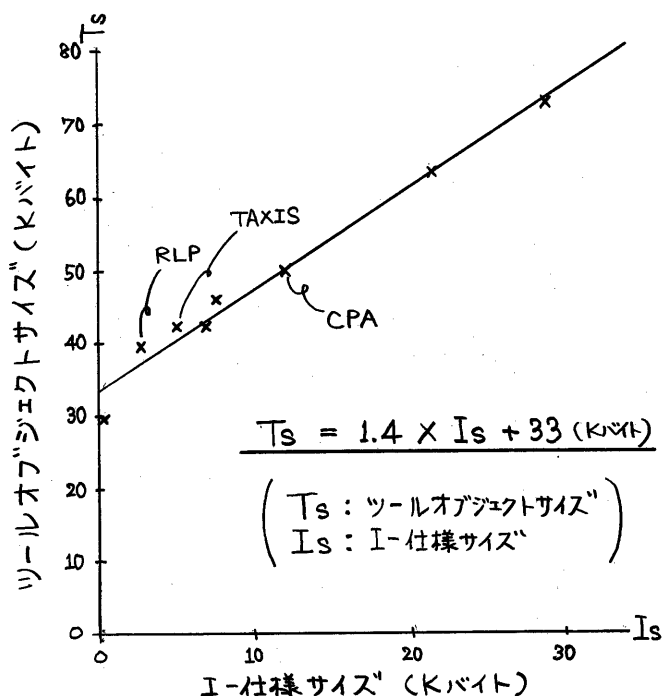
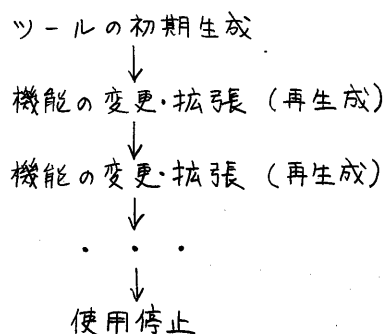


図16 I-仕様サイズとツールオブジェクトサイズとの関係

対応できる。たとえばユーザモデルへの項目の追加やユーザ言語の入力形式の変更を行ないたい場合には、そのツールのI-仕様の定義記述の該当する部分を追加または修正し、ISMOSを再度用いてツールの再生成を行なうだけで、ツールを保守することができる。この結果、ISMOSを用いた場合のツールのライフサイクルは以下のようになる。



高い信頼性

ISMOS自信の信頼性を高めておくことにより、生成されたツールの信頼性も高く保持できる事が期待される。

特に、従来のソフトウェア開発方法では、システムが応できあがった直後には多くのバグが発見されるのが通例であったが、ISMOSで生成したツールには、生成時点からきわめてバグが少なく、すぐに利用できている。これは、ISMOSの場合にはI-仕様という一種の要求定義ともみなせる仕様からプログラムを生成しているため、従

* 実際には、現在ではバグは見つかっていない。

来はソフトウェア開発のライフサイクルの名段階(論理設計→物理設計→プログラム設計→プログラミング)で、人により組み込まれていた論理エラーや不注意なミスによるバグがはいつつもなれたと考えられる。

生成したツールのわかりやすさ

ISMO言語を用いて記述されたツールの仕様、簡潔で形式化されたツールの定義となっているため、ユーザは仕様を見ることにより、生成されたツールをかなり理解できる。またISMOのユーザがツールを利用するのをやり易くするために、ユーザ用のドキュメントを出力している。

10. おわりに

DB指向ツールの作成を支援するためのツールとして開発したISMOについて、DB指向ツールの生成方法、ISMOの概要、その適用例と適用結果を報告した。

現在のISMOはまだまだ試作実験の段階にあると考えられる。今後の課題としては、メタモデルISMOとツール定義言語ISMOの機能強化、生成するツールへそなえさせるユーザインタフェースの向上、生成したツールの実用面からのフィードバックの反映などが挙げられる。

最後にISLDSとSEMについておしえていただき、本報告のISMOに多くの示唆を与えて下さったミシガン大学ISDOプロジェクトのTeichroew教授、Kang博士、また日頃御指導いただく当研究所の藤野所長、津所長(代)、寺本課長に深謝致します。

参考文献

- [1] 中田, 寺本: "情報システムの仕様化技術への意味モデル的アプローチ", 情処学会・ソフトウェア工学研究会資料24-3, 1982年6月
- [2] 山崎, 中田: "概念モデルに基づいたツール作成支援システム: ISMO S", 情処学会・第26回全国大会, 6J-8, 1983年3月, pp.617-618
- [3] 中田: "A Model Based Tool Generation and its Application to Telecommunication Systems Requirements Definition Tool", International Workshop on Software Development Tools for Telecommunication Systems, 1983年4月, Anaheim, CA, pp.38-41
- [4] Barron, J.L.: "Dialogue Organization and Structure for Interactive Information Systems", Technical Report CSRG-108, The University of Toronto, 1980.
- [5] Chen, P.P.: "The Entity-Relationship Model - Toward a Unified View of Data", ACM trans. Database Syst., 1,1, pp.9-36, 1976
- [6] Mylopoulos J., et al.: "A Language Facility for Designing Database-Intensive Applications", ACM Transaction on Database Systems, Vol.5, No.2, Jun, 1980, pp.185-207.
- [7] Teichroew, D. and Hershey III, E.A.: "PSL/PSA: A Computer-Aided Technique for Structure and Documentation and Analysis of Information Processing Systems", IEEE Transaction on Software Engineering, Vol.SE-3, No.1, Jan, 1977, pp.41-48.
- [8] Teichroew, D., et al.: "Application of the Entity-Relationship Approach to Information Processing Systems Modeling", in Entity-Relationship Approach to Systems Analysis and Design, Chen, P.P. (ed), North-Holland, 1980. pp.15-38