ソフトウェア工学 47－4
（1986. 5. 28）

設計仕様解析ツール：Dela

杉野一正　　大川　勉　　高野　彰
（三菱電機（株）　情報電子研究所）

ソフトウェアの設計を支援する機能として、設計段階から仕様の内容を確認することと、ソフトウェアの設計内容をわかりやすく示すことが考えられる。前者は、誤りの早期発見につながり、後工程からの手もどりによる修正作業の大幅な削減とソフトウェアの信頼性の向上をもたらす。後者は、ソフトウェアの再設計や拡張、再利用を実施する場合極めて有効であり、ソフトウェアの寿命を延ばしたり、重複したソフトウェアの開発を防ぐ。これらの機能を実現するツールとして、トップダウン設計を支援し、豊富な論理エラー検出機能を有する設計仕様解析ツールDelaと、設計段階からモジュール構成図、プログラム図、相互参照表を出力し、設計の進行過程を視覚的に表現するプログラム図生成ツールDIGとの連携について述べる。

WGSE 47-4
"Dela : A SOFTWARE TOOL FOR ANALYSIS OF FORMAL DESIGN SPECIFICATIONS" (in Japanese)

by Kazumasa SUGINO, Tsutomu OHKAWA and Akira TAKANO
(Information Systems & Electronics Development Laboratory, Mitsubishi Electric Corporation)

Dela(DEsign Language Analyzer) is a system development tool which analyzes specifications described in formal design language for consistency and completeness, and which provides a graphical interface to support the stepwise refinement process of software design and to represent interactions between data and procedure.
The design language on dela is based on Ada, it has special facilities for specifing input-output data, function of process, performance, Japanese text and pending item.

1．はじめに

ソフトウェアの設計を支援するツールを作成する場合、次の３つの項目を支援することが考えられる。

(a) 設計仕様が容易に作成できること

(b) 設計仕様の内容を解析できること

(c) 過去に作成されたソフトウェアの設計仕様の内容を容易に再利用できること

設計仕様解析ツールDela（ DEsign Language Analyzer ）は、(b)を中心にトータルシステムとして（プログラム図エディタ・コンパイラPEC(1)、ネットワーク図エディタ(2)、プログラム図生成ツールDIG(3) との連携を図ることで）(a)、(b)、(c)全体を支援することを目的とするものである。ここでは、項目(b)に相当するDelaの解析機能と項目(c)に相当するDelaとDIGとの連携について述べる。

2．ツールの背景

Delaは、次のようなソフトウェアの開発工程のモデルおよびソフトウェア設計手法を支援することを目的とするツールである。

通常ソフトウェアの開発工程のモデルは、図１のような作業の流れで示される。

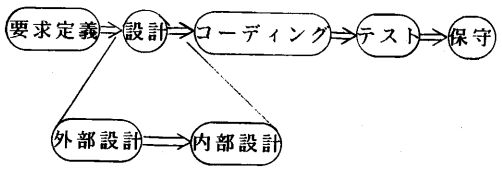


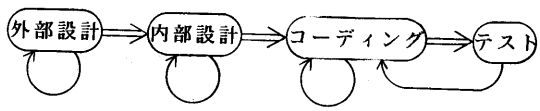
図１　ソフトウェアの開発工程

この開発工程に対して(c)の過去のソフトウェアの設計内容の有効利用を考慮する場合、そのソフトウェアの内容を容易に理解できる形で示す必要がある。そのためには、どのように設計が進められたかという設計の進行過程の情報と、どのような修正がなされたかという設計の履歴の情報があると役に立つ。特に後者は、コーディングやテスト、保守の段階から設計段階へのフィードバックを反映する情報と見なせる。そこで、このような設計のフィードバックをも考慮したモデルを次に示す。

ここでは設計へのフィードバックを基本修正、再設計、拡張・再利用の３つに分類する。以下、それぞれについてモデルを示す。

(1) 基本修正

ソフトウェア開発過程で頻繁になされているフィードバックである。具体的には、テスト中に誤りを見つけコーディングしなおす作業や、コーディング中に他のプログラムとの整合をとるためソースを修正する作業、および外部設計・内部設計中にモジュールを追加・削除・分割したり、共通変数を追加・削除する等の作業を示す。次の図２のようなモデルである。



開発工程

フィードバック

図２　基本修正

(2) 再設計

ここでの再設計とは、開発工程のテスト・コーディング等の後工程から外部設計・内部設計へのフィードバックを示す。具体的には、テスト中に発見した誤りがモジュールの内部処理方式やモジュールの分割に影響を及ぼす場合、また、内部設計・コーディング中に実現不可能になり、処理方式の変更を必要とする場合の作業を示す。次の図３のようなモデルである。

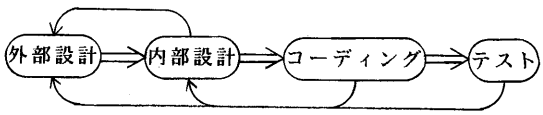


図３　再設計

(3) 拡張・再利用

ここでの拡張・再利用とは、保守の段階からの外部設計・内部設計へのフィードバックを示す。なお、保守は、一度でき上がったソフトウェアに対して何らかの作業を施すことを意味する。具体的には、機能拡張を実施するため、モジュールを追加したり、データ構造を変更したり、部品化を進めるため、関連モジュールをまとめたり、処理手順を変更したりする作業を示す。次の図４のようなモデルである。

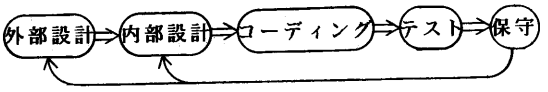


図４　拡張・再利用

なお、フィードバックに関して要求定義へのフィードバックも考えられるが、ここでは、取り扱わない。これらのフィードバックに対して、基本修正では、Delaの解析機能によるエラーメッセージ出力（エラーの種類、エラーの発生位置）によって修正が容易になる。また、再設計および拡張・再利用では、Delaで解析を行い内容を確認しつつ作成される設計仕様を適当なタイミングでDIGに渡し、その豊富な図表現に変換することで、ソフトウェアの設計過程や修正状況を容易に理解することが可能になる。なおかつ、その図表現により追加や修正の影響の度合も把握し易くなる。また、追加や修正後、Delaの解析機能により他の部分との整合性を即座に確認することができる。

次にどのような設計手法を支援するかについて述べる。ソフトウェアの設計手法として代表的なものに制御フロー中心設計がある。これは、制御の流れに従って機能分割を行いそれを詳細化していく手法である。Delaではこの制御フロー中心設計を後述する手続きの階層化や段階的詳細化により支援する。また、データ抽象化設計も支援する。これは、データとそれに関係する手続きや関数などの機能をひとまとめにして、それらのデータや機能を使う側に対して内部処理方式などの情報を隠してしまう技法である。DelaはAda(4)を基礎にして考えた設計言語を用いるため、Adaと同様にパッケージ等によりデータ抽象化設計を支援する。

３．Delaの概要

図５に設計仕様解析ツールDelaを中心にプログラム図生成ツールDIGとの連携を示す。なお、この章では、Delaの特徴と、DIGとの連携をとったDelaの特徴について述べる。

設計仕様解析ツール：Ｄｅｌａ

杉野一正　　大川　勉　　高野　彰
（三菱電機 株　情報電子研究所）

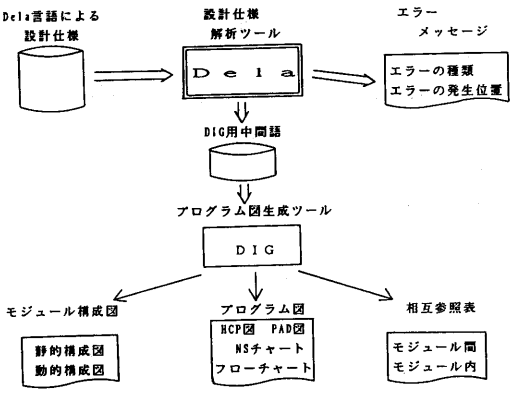


図５　設計仕様解析ツールDelaの概要

上図に示すように設計仕様解析ツールDelaは、Dela言語で記述された設計仕様を解析し、DIG用の中間語とエラーメッセージを出力するツールである。

また、プログラム図生成ツールDIGを用いてこの中間語からモジュール構成図、プログラム図、相互参照表を生成する。すなわち、設計仕様の内容を図表現に変換する。

３．１　Delaの特徴

設計仕様解析ツールDelaは、Adaの言語仕様を基に設計仕様記述に適するように改良したDela言語で表現された設計仕様の内容を解析する。そのDelaに関して、言語仕様と解析機能に分けて特徴を示す。

３．１．１　Delaの言語仕様

Dela言語は、Adaと同様にデータタイプやパッケージ等の定義・機能をもっている。さらに設計言語として必要と思われる以下の事項を追加している。

(1) 読解容易性

設計仕様として記述内容を解り易くするため、変数名、型名、手続き名、コメント等に日本語が利用できる。

(2) 段階的詳細化

設計の初期段階からコーディングに到るまで単一の言語で記述できることが望ましい。このため、設計内容があいまいなものに対して、未決定事項として表現することができる。この未決定事項を詳細化することで設計を進めていく。

（例）

IF ｛テーブルオーバフロー｝ THEN
　　｛エラーメッセージ出力｝；
⇓詳細化
IF（テーブルの大きさ＜テーブルの上限）THEN
　　ERROR_OUT(テーブルの名前,｛位置｝)；

(3) 厳密な仕様記述

手続きや関数、パッケージに対して次の３つの宣言が追加されている。

・入出力仕様：　変数やパラメタを入力データと出力データに分類し、どこから得るか、どこへ出力するかを宣言する。

（例）

カーソル位置：INPUT FROM VISUAL；
テキスト　　：OUTPUT TO FILE；

・性能仕様：　手続きや関数の予測実行時間や主記憶サイズを宣言する。

（例）

PERFORMANCE : SPEED 10..20 SIZE 200；

・機能仕様：　手続きや関数の機能を宣言する。

（例）

SPECIFICATION : ｛データ構造によりポップアップメニューを生成する｝;

(4) 手続きの階層化

ソフトウェアの開発をSYSTEM、BLOCK、PROGRAM、MODULEという階層構造に分けて設計を進めることを支援する。

３．１．２ Delaの解析機能

Delaでは次のような項目に対して解析を行い、誤りがあれば、エラーメッセージ（エラーの種類と発生位置）を出力する。

(1) パラメタとアーギュメントの関係

型の不一致、モード（入力、出力）の不一致を検査する。

(2) 入力仕様と手続き本体の関係

入力モードで与えられたデータに値をセットしていないか検査する。

(3) 出力仕様と手続き本体の関係

出力モードで与えられたデータに値をセットしているか検査する。

(4) 手続き関係

BLOCKがSYSTEMを呼び出していないか、PROGRAMがBLOCKやSYSTEMを呼び出していないかなどを検査する。

(5) 性能仕様の関係

PROGRAMのSIZEがその中のMODULEのSIZEの合計より大きいかなどを検査する。MODULEのSPEEDがそれを呼び出すPROGRAMのSPEEDより小さいかなどを検査する。

(6) 未決定事項とその他の部分との関係

未決定事項に関しては、その存在自体が解析の対象になる場合（例えば、手続きや関数のパラメタに未決定事項を用いたときでも、パラメタの個数に関する解析の対象となり、その過不足を検査する）誤りを検査する。また、逆に未決定事項により、解釈に自由度がある場合（列挙要素で未決定事項を用いたとき、実行部で他の列挙要素に該当しない要素名が存在しても、未決定事項に対応すると解釈して）余分なエラーメッセージを出力しない。これらにより、設計の進み具合いの異なる仕様間の解析を実現する。

次に診断システムの設計仕様を例にDelaによる設計仕様の記述や解析結果のエラーメッセージ出力を示す。

(a) パッケージ仕様

```
1 ------------------------------------------
2 --                患者の病歴                --
3 ------------------------------------------
4 PACKAGE KARTE IS
5 --
6     TYPE 認識番号 IS RANGE 0..99999;
7     TYPE 項目     IS (心臓, 胃, 脳, (その他));
8 END KARTE;
```

型名やコメントに日本語を用いており、列挙要素に未決定事項を用いている。

(b) 設計の第一段階

```
1 ------------------------------------------
2 --              診断システム              --
3 ------------------------------------------
4 --
5 WITH KARTE;          -- 患者の病歴
6 --
7 SYSTEM SINDAN IS
8 --
9     SPECIFICATION;
10        (患者の症状を聞き、体調を調べ病歴と照合し、治療法を決める)；
11    患者　：　認識番号；
12    患者の訴え　：　(症状等)；
13    治療指示　：　(治療法等)；
14 --
15    患者, 患者の訴え　：　INPUT;
16    治療指示　：　OUTPUT;
17 --
18 BEGIN
19    (患者の訴えにより検査項目を限定する)；
20    (体調を調べ、病歴と照合し、判定する)；
21    (判定に基づいて治療法を決定する)；
22 END SINDAN;
```

(c) 設計の第二段階

```
1 ------------------------------------------
2 --              診 断 シ ス テ ム          --
3 ------------------------------------------
4 --
5 WITH KARTE;          -- 患者の病歴
6 --
7 SYSTEM SINDAN IS
8 --
9     SPECIFICATION;
10        (患者の症状を聞き、体調を調べ病歴と照合し、治療法を決める) ;
11    患者 :  認識番号 ;
12    患者の訴え  :  (症状等) ;
13    治療指示  :  (治療法等) ;
14 --
15    患者, 患者の訴え  :  INPUT;
16    治療指示  :  OUTPUT;
17 --
18    PROGRAM  検査項目限定 ( (患者) ; (検査項目) ) ;
19    PROGRAM  体調判定  ( (患者) ; (検査項目) ; (結果) ) ;
20    PROGRAM  治療法決定  ( (結果) ; (治療指示) ) ;
21 --
22 BEGIN
23    検査項目限定 ( (認識番号) , (症状) , (検査項目リスト) ) ;
24    体調判定  ( (認識番号) , (検査項目リスト) , (検査結果) ) ;
25    IF  (結果が異常である)  THEN
26        治療法決定 ( (検査結果) , (治療指示) ) ;
27    END  IF;
28 END  SINDAN;
```

(d) 設計の第三段階

```
1 ------------------------------------------
2 --              診 断 シ ス テ ム          --
3 ------------------------------------------
4 --
5 WITH  KARTE, CHIRYOU;  -- 患者の病歴, 治療関係
6 USE   KARTE, CHIRYOU;
7 --
8 SYSTEM  SINDAN  IS
9 --
10    SPECIFICATION;
11       (患者の症状を聞き、体調を調べ病歴と照合し、治療法を決める) ;
12    PERFORMANCE : SPEED  10. . 20
13                  SIZE  1000. . 1500;
14    患者  :  認識番号 ;
15    患者の訴え  :  (症状等) ;
16    治療指示  :  (治療法等) ;
17 --
18    患者, 患者の訴え  :  INPUT;
19    治療指示  :  OUTPUT;
20 --
21    PROGRAM  検査項目限定 ( (患者) ; (検査項目) ) ;
22 --
23    PROGRAM  体調判定  ( (患者) ; (検査項目) ; (結果) ) IS
24       SPECIFCATION;
25          (指定された検査項目に従って患者を検査し、体調を判断する) ;
26       PERFORMANCE : SPEED  5. . 7
27                     SIZE  800. . 1600;
28       MODULE  カルテ書き込み ( (患者) ; (検査結果) ) ;
29    BEGIN
30       (検査を行う) ;
31       カルテ書き込み ( (認識番号) , (検査結果) ) ;
32       (患者の体調を判断する) ;
33    END  体調判定;
34 --
35 BEGIN
36    検査項目限定 ( (認識番号) , (症状) , (検査項目リスト) ) ;
37    体調判定  ( (認識番号) , (検査項目リスト) , (検査結果) ) ;
38    IF  (結果が異常である)  THEN
39        治療法決定 ( (検査結果) , (治療指示) ) ;
40    END  IF;
41 END  SINDAN;
42 --
```

(b)、(c)、(d)は、システムの設計が進んで行く過程を示している。特に(b)では、システムの機能説明と入出力変数と処理の流れの概略を未決定事項を用いて記述してある。また、(c) では、前の処理の部分の未決定事項を詳細化し、手続き宣言等を追加してある。そして、(d)では、前の手続き宣言の部分を詳細化し、治療法決定は、他のシステムでもよく使用されるのでCHIRYOUというパッケージに納めている。

なお、設計の第三段階の(d)のDelaによる解析結果は、次のようになる。

(e) 設計の第三段階の解析結果

```
1 ------------------------------------------
2 --              診 断 シ ス テ ム          --
3 ------------------------------------------
4 --
5 WITH  KARTE, CHIRYOU;  -- 患者の病歴, 治療関係
                  *
* パッケージが存在しない
6 USE   KARTE, CHIRYOU;
7 --
8 SYSTEM  SINDAN  IS
9 --
10    SPECIFICATION;
11       (患者の症状を聞き、体調を調べ病歴と照合し、治療法を決める) ;
12    PERFORMANCE : SPEED  10. . 20
13                  SIZE  1000. . 1500;
14    患者  :  認識番号 ;
15    患者の訴え  :  (症状等) ;
16    治療指示  :  (治療法等) ;
17 --
18    患者, 患者の訴え  :  INPUT;
19    治療指示  :  OUTPUT;
20 --
21    PROGRAM  検査項目限定 ( (患者) ; (検査項目) ) ;
22 --
23    PROGRAM  体調判定  ( (患者) ; (検査項目) ; (結果) ) IS
24       SPECIFCATION;
25          (指定された検査項目に従って患者を検査し、体調を判断する) ;
26       PERFORMANCE : SPEED  5. . 7
27                     SIZE  800. . 1600;
                                  *
* 階層間でサイズ情報が矛盾している
28       MODULE  カルテ書き込み ( (患者) ; (検査結果) ) ;
29    BEGIN
30       (検査を行う) ;
31       カルテ書き込み ( (認識番号) , (検査結果) ) ;
32       (患者の体調を判断する) ;
33    END  体調判定;
34 --
35 BEGIN
36    検査項目限定 ( (認識番号) , (症状) , (検査項目リスト) ) ;
                  *
* パラメタの個数が矛盾している
37    体調判定  ( (認識番号) , (検査項目リスト) , (検査結果) ) ;
38    IF  (結果が異常である)  THEN
39        治療法決定 ( (検査結果) , (治療指示) ) ;
40    END  IF;
41 END  SINDAN;
42 --

*** 結果 ***
*字句エラー *構文エラー *名前・式  *IOエラー *設計エラー
          0          0          0          0          3
```

最初のエラーは、パッケージ「CHIRYOU」がまだ作られていないことを示している。次のエラーは、システム「SINDAN」の性能のサイズの上限より、プログラム「体調判定」の性能のサイズの上限が大きいことを示している。最後のエラーメッセージは、プログラム「検査項目限定」において、宣言部と実行部でパラメタとアーギュメントの個数が一致しないことを示している。

このように未決定事項を用いてスムーズに設計を進め、解析機能により設計の各段階で誤りを指摘することができる。この解析機能は、その豊富な論理エラー検出機能とそのエラーメッセージ出力により、設計のフィードバックである基本修正を支援する。また、未決定事項に関する解析機能により詳細レベルの仕様と概略レベルの仕様との間の解析を可能にしている。これにより、設計の進捗がまちまちになりがちな、複数人で設計される仕様に対しても設計内容の解析ができ、誤りの早期発見につながる。

３．２　DelaとDIGの連携形態での特徴

Delaは、DIGの入力データである中間語を生成することにより、制御構造図、モジュール構成図、相互参照表の３種類の図表を出力することができる。そもそもDIGは、複数の言語に対してプログラムから容易に図表を生成できるように、共通中間語を設定している。その中間語には、プログラムコード、コメント、および制御フローに関する情報を格納する必要がある。Delaでは、詳細化途中の設計仕様からこれらの情報を含む中間語を生成し、各種図情報を出力する。これによって、設計内容を図表現で把握することができる。

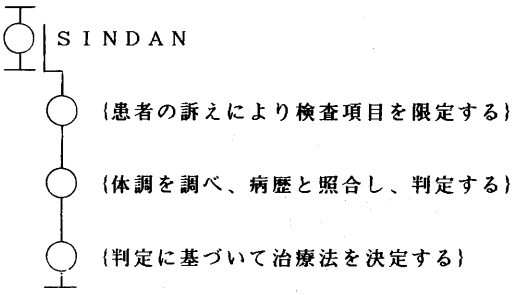
(1) 各モジュールの制御構造の作成過程や変遷を制御構造図で捕えられる。具体的には、例の診断システムのDela言語による設計仕様に対して、図６のようなHCP図を出力する。

右図のごとく、Dela言語を用いた設計過程の適切なタイミングでHCP図を出力すれば、ソフトウェア設計者の処理フローの作成過程を一目で理解できる。また、設計初期のHCP図により主なデータの流れが容易に理解できる。

(2) モジュールの分解過程やモジュールの移動状況等をモジュール構成図で捕えられる。これにより、モジュールの階層構造の形成過程、モジュールの結合状況が把握できる。具体的には、例の診断システムのDela言語による設計仕様に対して図７のようなモジュール構成図を出力する。

右図のごとくモジュール構成図を適切なタイミングで出力すれば、SINDANのモジュールの機能分割の状況が一目で理解できる。

＊診断システムの(c)に対するHCP図



＊診断システムの(d)に対するHCP図

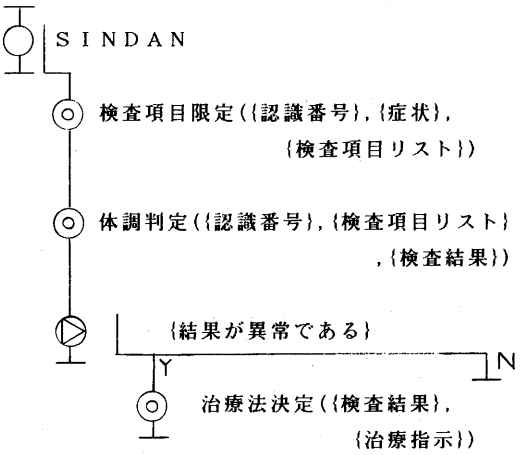


図６　設計仕様のHCP図出力

＊診断システムの(c)に対するモジュール構成図

＊診断システムの(d)に対するモジュール構成図

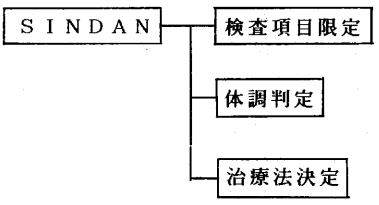


図７　設計仕様のモジュール構成図出力

(3) モジュール内のデータや、モジュール間における共通データの生成過程や利用状況を相互参照表で捕えられる。これにより、主なデータとモジュールとの関連を明確に把握できる。具体的には、例の診断システムのDela言語による設計仕様に対して図8のような相互参照表を出力する。

これにより、データの構造や属性等が、一目で理解できる。また、型や変数がどのモジュールで宣言され、どのモジュールで使用されているかが容易に解る。

なお、再設計や拡張・再利用においてそれぞれの図情報に基づいて処置したのち、下位モジュールや上位モジュールとの整合性をDelaの解析機能によって、容易に確認することができる。このように、修正から確認まで設計段階から支援することが可能である。

＊診断システムの(a)に対する相互参照表

```
※  相互参照表（モジュール間）  ※  パッケージ  KARTE          86／ 3／19  ページ      1
宣言位置        識別子              コメント／参照情報          属性
      7         項目                                          列挙型
      7           心臓                                        列挙要素名
      7           肺                                          列挙要素名
      7           腸                                          列挙要素名
      6         認識番号            ＋SINDAN                  整数型  RANGE  0 ..  99999
      4         KARTE                                         パッケージ名

データの数  ：        6
```

＊診断システムの(c)に対する相互参照表

```
※  相互参照表（モジュール内）  ※  モジュール  SINDAN          86／ 3／19  ページ      6
宣言位置        識別子              コメント／参照情報          属性
      5         KARTE                                         パッケージ名
      7         SINDAN              ＋9                       システム名
     11         患者                ＋15                      変数名  認識番号
     12         患者の訴え          ＋15                      変数名  （症状等  ）
     18         検査項目限定        ＋23                      プログラム名
     13         治療指示            ＋16                      変数名  （治療法等  ）
     20         治療法決定          ＋26                      プログラム名
     19         体調判定            ＋24                      プログラム名

データの数  ：        8
```

図8　設計仕様の相互参照表

これらの図情報により、設計のフィードバックに対して、次のような形で支援する。

再設計では、修正する変数・モジュールが影響を及ぼす変数・モジュールは、相互参照表の宣言位置や参照場所、参照状況により判別できる。また、関連するモジュールはモジュール構成図から、関連する変数は制御構造図からその使用状況がわかる。

拡張・再利用では、特定の変数・モジュールに関連する変数・モジュールは、相互参照表により判別でき、モジュール構成の変遷はモジュール構成図の履歴で、データ構造の変遷は相互参照表の履歴で、制御フローの変遷は制御構造図の履歴で把握することができる。

4．おわりに

Delaはすでに、MELCOM-COSMO900Ⅱ上で動作するものが完成しており、現在、スーパーミニコンピュータMELCOM70 MX/3000上への移植を行っている。なお、DelaとDIGの連携形態での長所として、再設計や拡張・再利用に対する有効性を強調した。しかし、設計・コーディング・テスト段階にも十分に活用できる。なぜなら、出力される図情報は、第三者が設計内容を理解するのに極めて有効な情報だからである。したがって、製作工程で遅れを生じ人員を補充する場合、Delaによる出力情報を用いれば投入人員が短期間で戦力になる。

・今後の課題

より使いやすく、なおかつ高機能な設計支援システムを実現するために、次の４つの課題がある。

(1) Delaの入力インタフェース

現在、Dela言語による設計仕様は、日本語ワークステーション上に例の診断システムのように作成されている。このような仕様書を作成する場合、Dela言語特有の構文規則を理解する必要がある。これを改善するために、ユーザインタフェースの良い構造エディタ等が入力部に望まれる。これが実現すれば、Delaの細かい構文規則にこだわらず、設計仕様の作成をスムーズに進めることができる。このようなエディタとしてプログラム図エディタ・コンパイラPECやネットワーク図エディタを考えている。

(2) Delaの言語仕様の拡張

Ada言語にあってDela言語にない機能として、データフロー設計法を支援する task と再利用を促進する generic がある。Delaの記述能力向上のためにも取り入れる必要がある。また、言語仕様の変更に伴い解析機能の向上（例:task 間のエラー解析）も必要である。

(3) Ada言語とのインタフェース

Ada言語とDela言語に対して図９のような双方向の変換が望まれる。これにより、Dela言語による設計仕様とAda言語によるパッケージとの整合性が確認できる。すなわち、設計情報と詳細なソースプログラムとの解析が可能になる。また、設計情報の詳細化が進めば、最終的にAda言語にスムーズに変換できる。なおかつ、Ada ソースをDela言語に変換することでDIGを通してAdaソースの図表出力が容易に得られる。

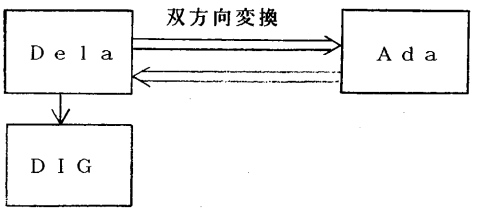


図９　DelaとAda双方向変換

(4) 未決定事項のソースコードへの自動変換

設計仕様中の未決定事項の詳細化の各段階で実際のソース（パッケージ中の宣言等）との整合性を確認している。これを一歩進めれば、ある程度詳細化が進んだ未決定事項と対応するソースコードとの自動変換が可能になる。これにより、設計仕様からソースコードへの自動生成に近づくことになる。

参　考　文　献

(1) 上野他、プログラム図エディタ／コンパイラ、第30回情処全大、4T-10

(2) 寿原、堀川、高野、汎用的なネットワーク図エディタ、第32回情処全大、2H-2

(3) 大川他、プログラム図生成ツール：DIG、情報処理学会ソフトウェア工学研究会、40-4、1985

(4) ANSI/MIL-STD-1851A, Ada Prog. Lang., 1983.

(5) E. Gabber, The Middle Way Approach for Ada Based PDL Syntax, Ada LETTER. Vol. 2, No. 4, pp. 64-67, 1983.

(6) 紫合、ソフトウェア設計法について、コンピータソフトウェア、Vol.1 No.2 July 1984

(7) 松本、ソフトウエア工学演習、電気・電子・通信・情報工学演習シリーズ、朝倉書店

(8) VEFSNMO, E. A. M.: "DASON" -A SOFTWARE ENGINEERING TOOL FOR COMMUNICATION APPLICATIONS INCREASING PRODUCTIVITY AND SOFTWARE QUALITY, Proc. 8th International Conference On Software Engineering (1985)

(9) 高野、春原、ソフトウェア設計言語：Dela、第29回情処全大、6P-2

(10) 杉野、高野、春原、ソフトウェア設計言語Delaの図形表現、第30回情処全大、7T-5

(11) N.Wirth,「アルゴリズム＋データ構造＝プログラム」（片山卓也 訳）コンピュータ・サイエンス研究書シリーズ，科学技術出版社