ソフトウエア工学 16－2
（1980. 11. 17）

性能テストのためテスト項目作成手法について

古川善吾　野木兼六　　越智　毅
（日立製作所　システム開発研究所）（同　ソフトウェア工場）

1. はじめに

ソフトウェアの開発費用の中でテストの占める割合は，40～50％にも達している。それにもかかわらず，テストに必要な作業の多くが人手で行なわれているため，テストの品質は必ずしも高くない。特に，テストの品質を決定する上で重要なテストケースの作成は，個人の経験や勘に頼っているためテストケースの漏れや重複が多く，ソフトウェアの品質保証上の大きな問題になっている。この問題を解決するために，近年，テストケースを作成するための方法論や支援ツールの研究が盛んに行なわれている。

テストケースを作成する考え方には，ソフトウェアの外部仕様に基づく性能テストと，内部仕様に基づく構造テストがある[1]。構造テストについては，各種の研究がなされてきているが，性能テストについては，その必要性は認識されながらも[2]現在の所，研究は殆んど行なわれていない。

われわれは，性能テストを系統的に行なう1つの方法として，原因結果グラフからテストケースを作成する手法[3][4]の検討を行ない[5]，支援ツールとしてAGENT（Automated GENeration system for Test cases）システムを開発した。

本稿では，AGENTの概要と試行結果について述べる。以下，第2章ではテストの考え方について，第3章では原因結果グラフについて，第4章ではAGENTについて，第5章では試行結果について述べる。

2. テスト手法

2.1 テストの概念

ソフトウェアのテストの理論的側面の研究はGoodenough等[6]によって始められた。彼らは，テスト基準を導入することによって，テストデータの作成を系統立てることを提案した。即ち，テスト基準が信頼でき（reliable）かつ確か（valid）である時，すべてのテストデータについてテストが成功（successful）ならば，そのプログラムに誤りがない，というものである。これによって，テスト基準がテストの品質を決める重要な要因であることが認識された。

しかし，最近，Weyuker等[7]が示した様に，信頼できかつ確かなテスト基準は，「すべての入力データを選択する」（全数テスト）しかない。これは，テストの基本的な性質である「テストは，テストを実施した入力データに対してのみ正しさを保証する」と直観的に合致する結論である。しかし，実用的なプログラムに対する全数テストは，事実上不可能であり，テスト基準としての全数テストは，現実的でない。

一方，Howden[8]は，誤りの種類を限定した上で，プログラムの制御フローに基づいたテスト基準について検討している。また，Weyuker等[7]は，制御フローと問題（仕様）による入力空間の分割に基づいたテスト基準を考えることを提案している。

われわれは，合理的なテストデータ作成の方法として，まず，入出力空間を有限個の部分領域に分割し，その後，テストデータを部分領域から選択することを

考えてきた。これは，テストによってプログラムの正しさを保証することは事実上不可能であるという認識に立った上で，いかに効果的にテストを行なうかについての考察に基づくものである。まず，ソフトウェアの外部仕様や内部仕様に基づいて論理的に入出力空間の分割を行なうことによって，チェック漏れをなくすようにする。次に，テストデータの選択に当っては，テストを実施した入力データに対してのみ正しさが保証されるので，できるだけ多くのテストデータを選ぶことが望ましい。しかし，現実のテストでは時間的制約があるために，どんなデータが誤りを発見しやすいかなどの経験や勘を頼りに，テストデータを選択する必要がある。

2．2　テストケースの作成法

テストケースは，入出力空間の部分領域を規定する条件であり，入力条件と期待出力の対として表わされる。入力条件は，入力として何を与えるかを定めたものであり，期待出力は，与えられた入力に対してどういう出力があった時，プログラムが正しいと判定するかを示す。出力が正しいか否かは，外部仕様に合っているか否かによって判定されるため，期待出力は外部仕様に基づいて作成される。一方，入力条件が何に基づいて作成されるかによって，機能テストと構造テストに分けられる。機能テストは，ソフトウェアの外部仕様に基づいて入力条件が定められるし，構造テストは内部仕様に基づいて入力条件が定められる。

機能テストは，ソフトウェアの外部仕様通りにプログラムが作成されているか，機能上抜けているところがないか，調べるのに適している。一方，構造テストは，プログラムに余分な部分がないか，自己矛盾がないか，調べるのに適している。この様に，2つのテスト法は互いに他の方法では発見が困難な誤りを発見するのに適しており，補ない合う関係にある。

構造テストについては，テストケースの作成法[9]，テストデータの作成法[10][11]，テスト状況の測定法[12][13][14]など，方法論や支援ツールの研究がなされてきた。しかし，機能テストについては，原因結果グラフを用いた手法[3][4]，や有限状態オートマトンを用いた手法などの研究があるに過ぎない。

そのため，われわれは，機能テストを系統的に行なうための，原因結果グラフを用いた手法について検討してきた[5]。原因結果グラフは，ソフトウェアの外部仕様を入力条件（原因）と出力状態（結果）の間の論理関係によって表わしたものである。この原因結果グラフから，系統的に，原因と結果の組み合わせとしてテストケースを作成することができる。AGENTシステムは，テストケースの作成を自動的に行なうことによって，機能テストを支援するツールである。

3．　原因結果グラフ

原因結果グラフを用いたテストは典型的な機能テストであり，図3-1に示す手順で行なわれる。まず，ソフトウェアの外部仕様を原因（入力条件）と結果（出力状態）の間の論理関係を用いた原因結果グラフで表わす。次に，原因結果グラフから，論理関係に基づいて必要十分なテストケースを作成する。さらに，テストケースの入力条件を満たすテストデータとして，条件を満たす代表値や誤りを発見しやすい値などを設定する。

原因結果グラフの例を図3-3に示す。この例は，図3-2の仕様，「90円のジュース自動販売機」を表わしている。

原因結果グラフは表3-1に示す要素から成っている。

表3-1　原因結果グラフの構成要素

No.	構成要素	内容
1	原因	・入力についての条件を表わす命題
2	結果	・出力の状態を表わす命題
3	関係	・原因と結果の間の論理関係
4	制約条件	・原因の間のあり得ない組み合せを示す。
5	中間節点	・原因と結果の間を結ぶのに用いる補助的な節点

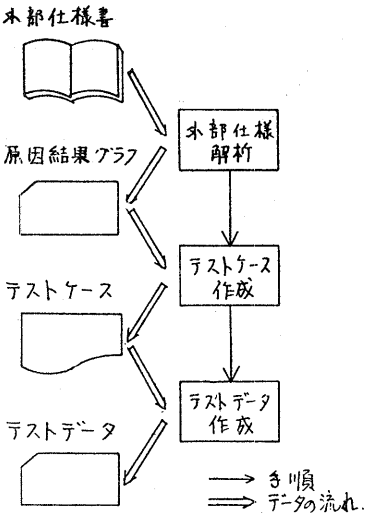

図3-1　原因結果グラフを用いたテスト手順

(1) 原因

原因は，入力についての条件を表わす命題である。これが入力条件になる。原因は常に，その成立，不成立が判定できるものでなければならない。

(2) 結果

結果は，出力の状態を表わす命題である。これがテストケースの期待出力になる。

(3) 関係

関係は，原因と結果の間の論理関係である。図3-4に示す基本論理関係を組み合わせて用いる。関係は，結果を生じるための必要かつ十分な原因を結びつけるものであり，原因と結果の間の因果関係を表わしている。

(4) 制約条件

制約条件は，原因の間のあり得ない組み合わせを示すものである。図3-5に示す4種類がある。

(5) 中間節点

原因と結果が1つの基本論理関係で表わせないとか，共通の関係を表わすのに用いる。

仕様：90円のジュースの自動販売機

100円硬貨又は90円を投入して，オレンジジュース又はリンゴジュースのボタンを押すと対応したジュースが出てくる。100円硬貨を投入した時は，ジュースと一緒に10円が返却される。但し，つり銭がない場合，つり銭切れランプが点灯されており，100円硬貨を投入しても硬貨が返却される。

図3-2　仕様の例

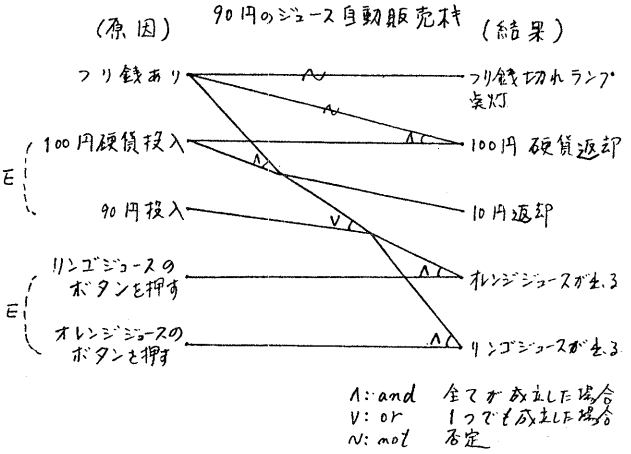

図3-3　原因結果グラフの例

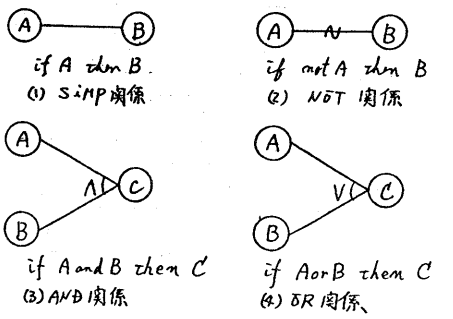

図3-4 基本論理関係

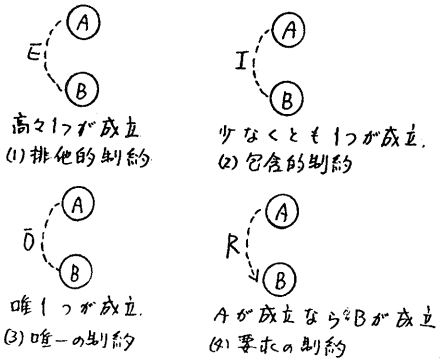

図3-5 制約条件

以上の要素を用いてソフトウェアの外部仕様を原因結果グラフで表わし，次に決定表を作成すると同様にしてテストケースを作成する。

4. AGENTシステム

AGENTシステムは，原因結果グラフからテストケースを自動的に作成するシステムである。図4-1に示す様に，AGENTシステムは，入力解析部とテストケース作成部から成っている。

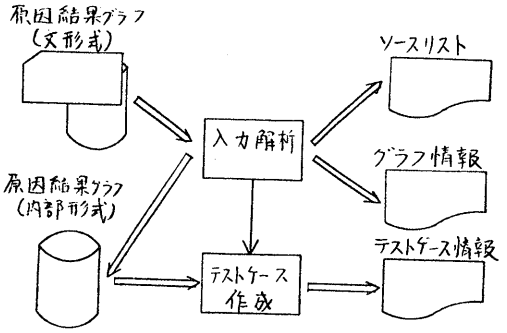

図4-1 AGENTの概要

入力は，原因結果グラフを文形式で表現したものである。文には以下の様なものがある。

(1) 節点文

節点の種類（原因，結果，中間）や節点が成立した時と不成立の時の命題（自然語）を記述する。節点の種類は，内部で決められる種類（先行点がない…原因，後続点がない…結果，その他…中間）との照合に用いるもので省略しても良い。また，命題は，テストケースの意味の説明に用いるので，中間節点の様にテストケースに現われないものは，節点文で定義する必要はない。

(2) 関係文

各基本論理関係毎に子節点と親節点，論理関係の種類を記述する。各論理関係毎に記述するのは，中間節点を明確にするためである。

(3) 制約条件文

制約条件に関連する節点と制約条件の種類を記述する。

節点，関係，制約条件は8桁以内の名票によって識別される。原因結果グラフを表わす上記の3つの文の他に，表題文と終了文がある。

入力解析部では構文チェックの他に，原因結果グラフにループがないか，他の節点と繋がりのない節点がないか，などのチェックを行なう。出力には，次の様なものがあり，例を図4-2から図4-5に示す。例は，「90円のジュース自動販売機」である。

(1) ソースリスト

原因結果グラフのソースリストを示す（図4-2）。節点I1とI2は中間節点であるため，定義されておらず未定義の警告メッセージが出されている。

```
** SOURCE LIST **
SEQ.    SOURCE TEXT
  1  @ +S,+L,+N,+R,+C,+T,+F/
  2   TITLE = '** TEST - 01 **'
  3   NODE N1 = ' ﾂﾘｾﾝ ｱﾘ '
  4        N2 = ' 100ｴﾝ ｺｳｶ ﾄｳﾆｭｳ '
  5        N3 = '  90ｴﾝ ﾄｳﾆｭｳ '
  6        N4 = ' ｵﾚﾝｼﾞ" ｼﾞｭｰｽ ﾉ ﾎ"ﾀﾝｵｽ '
  7        N5 = ' ﾘﾝｺﾞ" ｼﾞｭｰｽ ﾉ ﾎ"ﾀﾝｵｽ '
  8        X1 = ' ﾂﾘｾﾝ ｷﾞﾚ - ﾗﾝﾌﾟﾃﾝﾄｳ '
  9        X2 = ' ｺｳｶ ﾍﾝｷｬｸ '
 10        X3 = ' 10ｴﾝ ﾍﾝｷｬｸ '
 11        X4 = ' ｵﾚﾝｼﾞ" ｼﾞｭｰｽ ﾃﾞﾙ '
 12        X5 = ' ﾘﾝｺﾞ" ｼﾞｭｰｽ ﾃﾞﾙ '
 13  *
 14  * RELATION
 15   RELATE R1 = N1 N2 AND I1
 16          R2 = I1 N3 OR I2
 17          R3 = NOT N1 SIMP X1
 18          R4 = NOT N1 N2 AND X2
 19          R5 = I1 SIMP X3
 20          R6 = I2 N4 AND X4
 21          R7 = I2 N5 AND X5
 22  *
 23   CONST C1 = N2 N3 EXCLUSIVE
 24         C2 = N4 N5 EXCLUSIVE
 25  *
 26   END

 ** DIAGNOS MESSAGE **
1  100 WARNING  NAME-I1       -NOT DEFINED
2  100 WARNING  NAME-I2       -NOT DEFINED
```

図4-2 ソースリストの例

(2) 節点リスト

節点の名票と種類，成立,不成立時の命題を示す（図4-3）。節点文で不成立時の命題が与えられていない時は，成立時の命題にNOTをつけて表わす。

```
** NODE LIST **
NO.   LABEL      KIND      MEANING
  1   N1         CAUSE      ﾂﾘｾﾝ ｱﾘ
                           /NOT ( ﾂﾘｾﾝ ｱﾘ )
  2   N2         CAUSE        100ｴﾝ ｺｳｶ ﾄｳﾆｭｳ
                           /NOT ( 100ｴﾝ ｺｳｶ ﾄｳﾆｭｳ )
  3   N3         CAUSE        90ｴﾝ ﾄｳﾆｭｳ
                           /NOT ( 90ｴﾝ ﾄｳﾆｭｳ )
  4   N4         CAUSE      ｵﾚﾝｼﾞ" ｼﾞｭｰｽ ﾉ ﾎ"ﾀﾝｵｽ
                           /NOT ( ｵﾚﾝｼﾞ" ｼﾞｭｰｽ ﾉ ﾎ"ﾀﾝｵｽ )
  5   N5         CAUSE      ﾘﾝｺﾞ" ｼﾞｭｰｽ ﾉ ﾎ"ﾀﾝｵｽ
                           /NOT ( ﾘﾝｺﾞ" ｼﾞｭｰｽ ﾉ ﾎ"ﾀﾝｵｽ )
  6   I1         INTERM     ** MEANS OF NODE NOT GIVEN **
                           /** MEANS OF NODE NOT GIVEN **
  7   I2         INTERM     ** MEANS OF NODE NOT GIVEN **
                           /** MEANS OF NODE NOT GIVEN **
  8   X1         EFFECT     ﾂﾘｾﾝ ｷﾞﾚ - ﾗﾝﾌﾟﾃﾝﾄｳ
                           /NOT ( ﾂﾘｾﾝ ｷﾞﾚ - ﾗﾝﾌﾟﾃﾝﾄｳ )
  9   X2         EFFECT     ｺｳｶ ﾍﾝｷｬｸ
                           /NOT ( ｺｳｶ ﾍﾝｷｬｸ )
 10   X3         EFFECT     10ｴﾝ ﾍﾝｷｬｸ
                           /NOT ( 10ｴﾝ ﾍﾝｷｬｸ )
 11   X4         EFFECT     ｵﾚﾝｼﾞ" ｼﾞｭｰｽ ﾃﾞﾙ
                           /NOT ( ｵﾚﾝｼﾞ" ｼﾞｭｰｽ ﾃﾞﾙ )
 12   X5         EFFECT     ﾘﾝｺﾞ" ｼﾞｭｰｽ ﾃﾞﾙ
                           /NOT ( ﾘﾝｺﾞ" ｼﾞｭｰｽ ﾃﾞﾙ )
```

図4-3 節点リストの例

(3) 関係リスト

関係の名票と親節点の名票種類，論理関係の種類，子節点の名票を示す（図4-4）。

(4) 制約条件リスト

制約条件の名票と制約条件の種類，関連する原因節点の名票を示す（図4-5）。

次に，テストケース作成部では，テストケースを作成する。出力には次のものがあり，前と同じ例について図4-6，図4-7に出力例を示す。

(5) テストケースの説明

節点文で与えられた命題を用いてテストケースの意味を説明する。図4-6に，テストケースの一部を示す。例えば，テストケース1は，入力条件として，釣り銭切れの時，100円硬貨を投入して，オレンジジュースのボタンを押しても硬貨が返却されるだけであることを示している。

(6) テスト用帳票

テストケースをテスト用帳票にまとめたものである。図4-7に例を示す。'x'は，原因や結果の命題が成立することを示し，空白は，命題が不成立であることを示す。ここに示した例では，6個のテストケースが作成される。

```
** RELATION LIST **
NO.   LABEL    FATHER(K)   OP    SON
  1   R1       I1      (X) AND         N1      :    N2
  2   R2       I2      (X) OR          I1      :    N3
  3   R3       X1      (E) SIMP    NOT N1      :
  4   R4       X2      (E) AND     NOT N1      :    N2
  5   R5       X3      (E) SIMP        I1      :
  6   R6       X4      (E) AND         I2      :    N4
  7   R7       X5      (E) AND         I2      :    N5
```

図4-4 関係リストの例

```
** CONSTRAINT LIST **
NO.    LABEL   CONDITION    NODE
  1    C1      EXCLUSIVE        N2        :    N3
  2    C2      EXCLUSIVE        N4        :    N5
```

図4-5 制約条件リストの例

```
** DESCRIPTIVE TEST CASE LIST **
TEST CASE     1

   CAUSE
     (-  N1      )  NOT ( ﾂﾘｾﾝ ｱﾘ )
     (+  N2      )   100ｴﾝ ｺｳｶ ﾄｳﾆｭｳ
     (-  N3      )  NOT (  90ｴﾝ ﾄｳﾆｭｳ )
     (+  N4      )   ｵﾚﾝｼﾞ" ｼﾞｭｰｽ ﾉ ﾎﾞﾀﾝｵｽ
     (-  N5      )  NOT ( ﾘﾝｺﾞ" ｼﾞｭｰｽ ﾉ ﾎﾞﾀﾝｵｽ )
   EFFECT
     (+  X1      )   ﾂﾘｾﾝ ｷﾞﾚ - ﾗﾝﾌﾟ"ﾃﾝﾄｳ
     (+  X2      )   ｺｳｶ ﾍﾝｷｬｸ
     (-  X3      )  NOT ( 10ｴﾝ ﾍﾝｷｬｸ )
     (-  X4      )  NOT ( ｵﾚﾝｼﾞ" ｼﾞｭｰｽ ﾃﾞﾙ )
     (-  X5      )  NOT ( ﾘﾝｺﾞ" ｼﾞｭｰｽ ﾃﾞﾙ )
TEST CASE     2

   CAUSE
     (+  N1      )   ﾂﾘｾﾝ ｱﾘ
     (-  N2      )  NOT ( 100ｴﾝ ｺｳｶ ﾄｳﾆｭｳ )
     (-  N3      )  NOT (  90ｴﾝ ﾄｳﾆｭｳ )
     (+  N4      )   ｵﾚﾝｼﾞ" ｼﾞｭｰｽ ﾉ ﾎﾞﾀﾝｵｽ
     (-  N5      )  NOT ( ﾘﾝｺﾞ" ｼﾞｭｰｽ ﾉ ﾎﾞﾀﾝｵｽ )
   EFFECT
     (-  X1      )  NOT ( ﾂﾘｾﾝ ｷﾞﾚ - ﾗﾝﾌﾟ"ﾃﾝﾄｳ )
     (-  X2      )  NOT ( ｺｳｶ ﾍﾝｷｬｸ )
     (-  X3      )  NOT ( 10ｴﾝ ﾍﾝｷｬｸ )
     (-  X4      )  NOT ( ｵﾚﾝｼﾞ" ｼﾞｭｰｽ ﾃﾞﾙ )
     (-  X5      )  NOT ( ﾘﾝｺﾞ" ｼﾞｭｰｽ ﾃﾞﾙ )
TEST CASE     3

   CAUSE
     (+  N1      )   ﾂﾘｾﾝ ｱﾘ
     (+  N2      )   100ｴﾝ ｺｳｶ ﾄｳﾆｭｳ
     (-  N3      )  NOT (  90ｴﾝ ﾄｳﾆｭｳ )
     (+  N4      )   ｵﾚﾝｼﾞ" ｼﾞｭｰｽ ﾉ ﾎﾞﾀﾝｵｽ
     (-  N5      )  NOT ( ﾘﾝｺﾞ" ｼﾞｭｰｽ ﾉ ﾎﾞﾀﾝｵｽ )
   EFFECT
     (-  X1      )  NOT ( ﾂﾘｾﾝ ｷﾞﾚ - ﾗﾝﾌﾟ"ﾃﾝﾄｳ )
     (-  X2      )  NOT ( ｺｳｶ ﾍﾝｷｬｸ )
     (+  X3      )   10ｴﾝ ﾍﾝｷｬｸ
     (+  X4      )   ｵﾚﾝｼﾞ" ｼﾞｭｰｽ ﾃﾞﾙ
     (-  X5      )  NOT ( ﾘﾝｺﾞ" ｼﾞｭｰｽ ﾃﾞﾙ )
                :
```

図4-6 テストの説明の例

AGENTシステムで作成されたテストケースは，次の様な誤りを発見することができる。

(1) 原因,結果が常に成立する。

(2) 原因,結果が常に成立しない。

(3) 原因結果の成立,不成立が逆になっている。

例えば，「90円のジュース自動販売機」の例で，6個のテストケースで，原因結果の成立,不成立がすべて設定されており，(1)~(3)の誤りを発見できる。

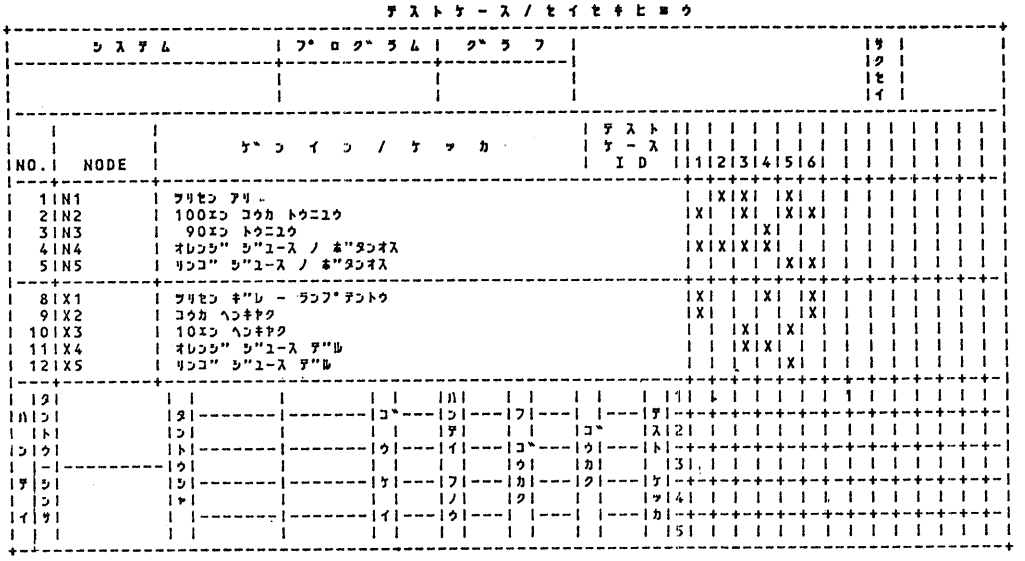

図4-7　テスト用帳票の例

5．　試用結果

AGENTシステムを用いていくつかのソフトウェアの原因結果グラフからテストケースを作成した。テストケースの作成を試みたソフトウェアの種別，プログラムの大きさ，グラフの数，節点数，テストケースの数を表5-1に示す。また，同じソフトウェアについて，従来と同様に外部仕様を読んで人手でテストケースを作成したものもある。この試用を通じて次の様なことがわかった。

表1　AGENT試用のまとめ

NO.	種別	規模	グラフ数	節点数 平均	節点数 MIN	節点数 MAX	テストケース数	ステップ数/グラフ	テストケース数/グラフ
1	ユーティリティ	0.3 KS	4	24	10	35	41	89	10.3
2	業務	28	8	18	3	25	82	350	10.3
3	制御	4.0	23	26	5	43	146	174	6.3

(1) 原因結果グラフの大きさ

表1に示した様に，1つのソフトウェアに対し複数個のグラフを作成した。これは，条件数が多すぎて1枚の原因結果グラフに書けないとか，原因と結果の間の因果関係が複雑なために論理関係だけでは外部仕様を表わせない，などのためである。この問題点を解決するために，外部仕様のまとまった機能や，処理の段階毎に原因結果グラフを作成した。各原因結果グラフは，表わそうとする機能の複雑さや入力条件の取り方によって，節点数やテストケースの数が表1の様に変わってくる。われわれの経験によると，1枚の紙に書けてかつ人間が理解できる原因結果グラフは，節点数が40～50ぐらいまでである。

(2) テストケースの数

テストケースの数は，直接人手で作成したものと比較すると60～150%になった。AGENTシステムでは，まず，テストケースを機械的に作成した後，組み合わせが可能なテストケースをできるだけまとめている。一方，人は，テストケースを次々に拾い出し，その過程で不要と考えたものを除いていく。そのため，組み合わせの効果が大きい時は減少するが，人の拾い出し方が少ないとか排除するものが多いとテストケースの数は増加する。

さらに，正常処理と異常処理に対するテストケースとして分類すると次の事がわかった。

・正常処理に対するテストケースの数は減少する。

・異常処理に対するテストケースの数は増加する。

これは，正常処理については人間が機能を理解しているために充分なテストケースを揚げることができるが，異常処理に対しては漏れが多いためと考えられる。

(3) テストケース作成時間

テストケースの作成時間は，直接人手で作成する場合に較べ1～10倍程度かかる。これは，単に外部仕様を読んでテストケースを揚げるだけで済んでいたものが，原因結果グラフを書くための時間が余分にかかるためと考えられる。

原因結果グラフを書くためには，まず，機能を分割した後に，原因と結果の間の因果関係を明確にする必要がある。一方，外部仕様は，多くが自然語で書かれているためにあいまいなところが多く，明確な因果関係を求めるのに時間がかかる。さらに，原因結果グラフの論理関係は，原因の組み合わせがすべて可能であると考えてつけられる。一方，ソフトウェアでは，複数個の原因が1つの変数に関係するとか順序があるなどのために起こり得ない組み合わせがある。この起こり得ない組み合わせを制約条件で表わしているが，論理関係と制約条件の関係が複雑である。そのため，原因結果グラフの記述に誤りが入りやすく，その修正に時間がかかっている。

6. おわりに

機能テストのために，外部仕様を表わす原因結果グラフからテストケースを自動的に作成する，AGENTシステムについて述べた。AGENTシステムを試用してみたところ，次の様な効果があることがわかった。

(1) 正常処理に対するテストケースを削減できる。

(2) 異常処理に対するテストケースの漏れをなくすことができる。

(3) 外部仕様の不明な点を適出できる。

一方，AGENTシステムを効果的に利用するためには，今後，原因結果グラフの作成技法，構造テストとの有機的な結合などを検討していく必要がある。

〔参考文献〕

1) Myers, G.J.: Software Reliability, John Wiley & Sons, Inc. 1976 (「ソフトウェアの信頼性」 有沢誠訳 近代科学社)
2) Howden, W.E.: Functional Program Testing, IEEE Trans. Soft. Engi., Vol. SE-6, No. 2 (1980)
3) Elmendorf, W.R.: Functional Analysis Using Cause-Effect Graphs, Proc. SHARE XLIII, N.Y: SHARE 1974
4) Myers, G.J.: The Art of Software Testing, John Wiley & Sons, Inc. 1979 (「ソフトウェア・テストの技法」) 松尾正信訳 近代科学社)
5) 古川: 科能テストのためのテストケース作成手法について, 情報処理学会, 第21回全国大会予稿集, 1980
6) Goodenough, J.B. et al.: Toward a Theory of Test Data Selection, SIGPLAN NOTICES, Vol. 10, No. 6, 1975
7) Weyuker, E.J. et al.: Theories of Program Testing and the Application of Revealing Subdomains, IEEE Trans. Soft. Engi., Vol. SE-6, No. 3 (1980)
8) Howden, W.E.: Reliability of Path Analysis Testing Strategy, IEEE Tran. Soft. Engi., Vol. SE-2, No. 3, 1976
9) Miller Jr., E.F. etal: Automated Generation of Test case Datasets, SIGPLAN NOTICES, Vol. 10, No. 6, 1975
10) Boyer, R.S. etal.: SELECT-- A Formal System for Testing and Debugging Programs by Symbolic Execution, SIGPLAN NOTICES, Vol. 10, No. 6, 1975
11) Clarke, L.A.: System to Generate Test Data and Symbolicaly Execute Programs, IEEE Trans. Soft. Engi., Vol. SE-2, No. 3, 1976
12) Huang, J.C.: An Approach to Program Testing, Computing Surney, Vol. 7, No. 3, 1975
13) Stacki, L.G.: New Assertion Concept For Self-Metric Software Validation SIGPLAN NOTICES, Vol. 10, No. 6, 1975
14) 牛島和夫: Fortran プログラミングツール, 産業図書, 1979