U.D.C. 〔681.32.06.001.4:519.686〕:〔658.512.2.011.56:681.322〕

ソフトウェアテスト項目作成支援システム

Automated Generation System of Test Cases

野木兼六*　*Kenroku Nogi*
古川善吾*　*Zengo Furukawa*
保田勝通**　*Katsuyuki Yasuda*

ソフトウェアの社会的影響度が増大し，その信頼性向上がますます重要になっている。そこで日立製作所では，従来，個人の経験や直観に頼っていたテスト作業を合理化するために，系統的なテスト項目作成技法及びその支援システムを開発した。

本技法は，まず，ソフトウェアの機能仕様から，人手によって，機能図式と呼ばれる形式化された機能仕様を作成し，次に，この機能図式から，計算機によって，漏れや重複のないテスト項目を自動的に生成するものである。

これによって，未発見不良の削減，テスト項目の圧縮，個人差によるばらつきの防止などを図ることができる。

1　緒　　言

ソフトウェアのテストには，二つの問題がある。一つは，プログラミングなどと比較してテストの作業効率が悪いことであり，もう一つは，効果的なテスト項目の作成が難しいことである。前者については，近年，各種のテスティング モニタ，デバッガなどが開発されてきているが，後者については，二，三の技法が提案されているだけであり，実用的な支援システムはまだ存在しない。

テスト項目を系統的に作成するには，通常，入力条件に基づいて入力データ領域を分割し，各領域からテスト データを選択するという方法がとられる。しかし，実際のプログラムでは，入力条件のすべての組合せは膨大な数になってしまい，それだけのテストを実施することは現実的に不可能である。したがって，テスト項目の作成は，考えられるすべての組合せの中から，いかにして現実的な数で，かつ最も不良発見能力の高いテスト項目の集合を選択するか，という問題としてとらえる必要がある。同じ数のテスト項目でも，不良発見能力には大きな差があることが経験的に知られているからである。

系統的なテスト項目の作成には，大別して，二つのアプローチがある。一つは，プログラムの機能仕様に基づいてテスト項目を選択する機能テスト(ブラックボックス テスト)であり，もう一つは，プログラムの構造仕様に基づいてテスト項目を選択する構造テスト(ホワイトボックス テスト)である。これらは相補的な関係にあり，組み合わせて実施することが重要である[1),2)]。

今回開発した技法は，機能テストのための系統的テスト項目作成技法であり，機能仕様で定義されたプログラムの機能の組合せをテストするのに必要かつ十分なテスト項目を自動的に生成する[3)]。

2　技法の概要

従来の機能テストでは，開発者が機能仕様を読み，個人の経験や直観に頼って思いつくままにテスト項目を拾い出していた。このため，テスト項目の漏れや重複が多く，不良発生の大きな要因になっていた。これに対し本技法では，テスト項目の拾い出しを2段階に分けて行なう。すなわち，第1段階で，自然語などによって記述された機能仕様から，機能図式と呼ばれる形式化された機能仕様を作成し，第2段階で，

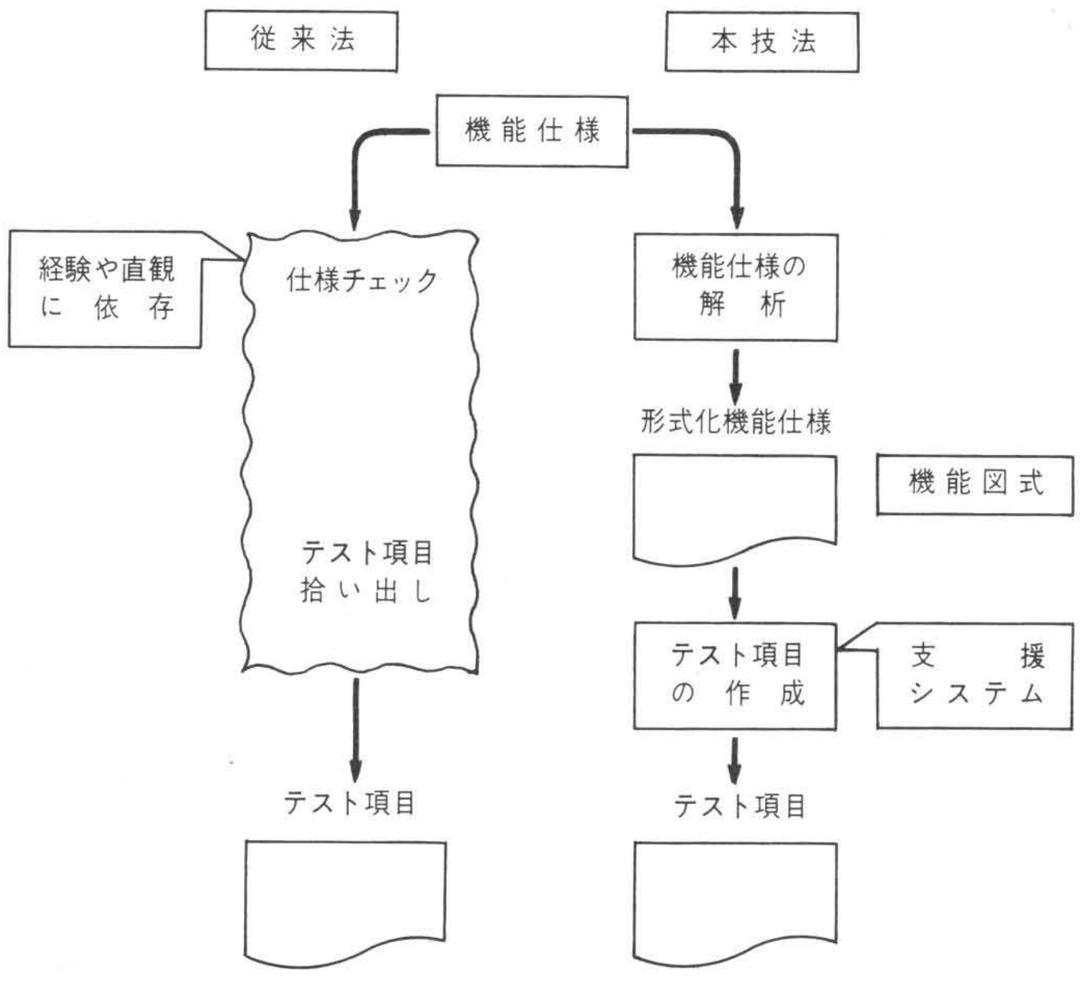


図1　本技法の概要　本技法では，まず機能図式を作成し，次にこれから漏れや重複のないテスト項目を自動的に生成する。

この機能図式を支援システムに入力し，テスト項目を自動的に生成する。本技法の概要を**図1**に示す。

これにより，次のような効果を期待することができる。

(1)　信　頼　性

現在，不良発生要因の60～80%は，テスト項目の漏れと言われている。本技法は，機能図式から漏れのないテスト項目を自動的に生成するため，未発見不良の削減を図ることができる。また，機能仕様を形式化する段階で，あいまいさや不完全さなど機能仕様の不備を発見することができる。

(2)　生　産　性

これまでの経験的な手法では，テスト項目の品質が分からないため，必要以上のテスト項目を作成する傾向があった。これに対して本技法は，機能図式から重複のないテスト項目を自動的に生成するため，テスト項目の圧縮を図ることができる。また，テスト用帳票などを自動的に作成するため，テ

* 日立製作所システム開発研究所　** 日立製作所ソフトウェア工場

スト文書作成作業の削減を図ることができる。

(3) 管 理 性

従来法では，開発者の経験年数の違いなどにより，テスト項目の品質が大幅に変動することが多かった。これに対して本技法は，機能図式からテスト項目を機械的に生成するため，個人差によるテスト品質のばらつきの防止を図ることができる。また，テスト進捗度の管理も容易になる。

3 機能図式

機能テストのための形式化された機能仕様の記法としては，原因結果グラフがよく知られている[1)]。原因結果グラフは，プログラムの入力条件(原因)と出力条件(結果)の間の因果関係を論理式で，また，入力条件同士の依存関係を制約条件で表現するものであり，図的な記法にその特徴がある。**図2**に，部品納入コマンドの機能仕様を原因結果グラフで表現した例を示す。この原因結果グラフは，組合せ論理で実現される静的なプログラムの機能仕様を表現するのには適しているが，順序論理で実現される動的なプログラムの機能仕様を簡潔に表現することができない。また，組合せ論理の場合でも，条件の組合せの網羅性をチェックすることが困難であるという欠点もある。更に，プログラムが大きくなるとその複雑さが急激に増大するという傾向があり，一度作成した原因結果グラフの変更が極めて困難であるという欠点もある。

そこで日立製作所では，原因結果グラフのこのような問題点を解決するいっそう強力な記法として，機能図式記法を開発した。機能図式は，機能仕様の動的な部分を状態遷移で表現し，静的な部分を論理関係(決定表あるいは原因結果グラフ)で表現する。具体的には，状態遷移の各状態での条件の組合せを表現するのに決定表あるいは原因結果グラフを使用する。どちらを使用するかは任意に選択することができる。**図3**に，現金自動支払機の機能仕様を機能図式で表現した例を示す。この機能図式は，動的なプログラムの機能仕様を表現することができるだけでなく，決定表により条件の組合せの網羅性をチェックすることもできる。また，複雑さが各状態に分散されるため，プログラムの大きさによる影響をあまり受けないという利点もある。

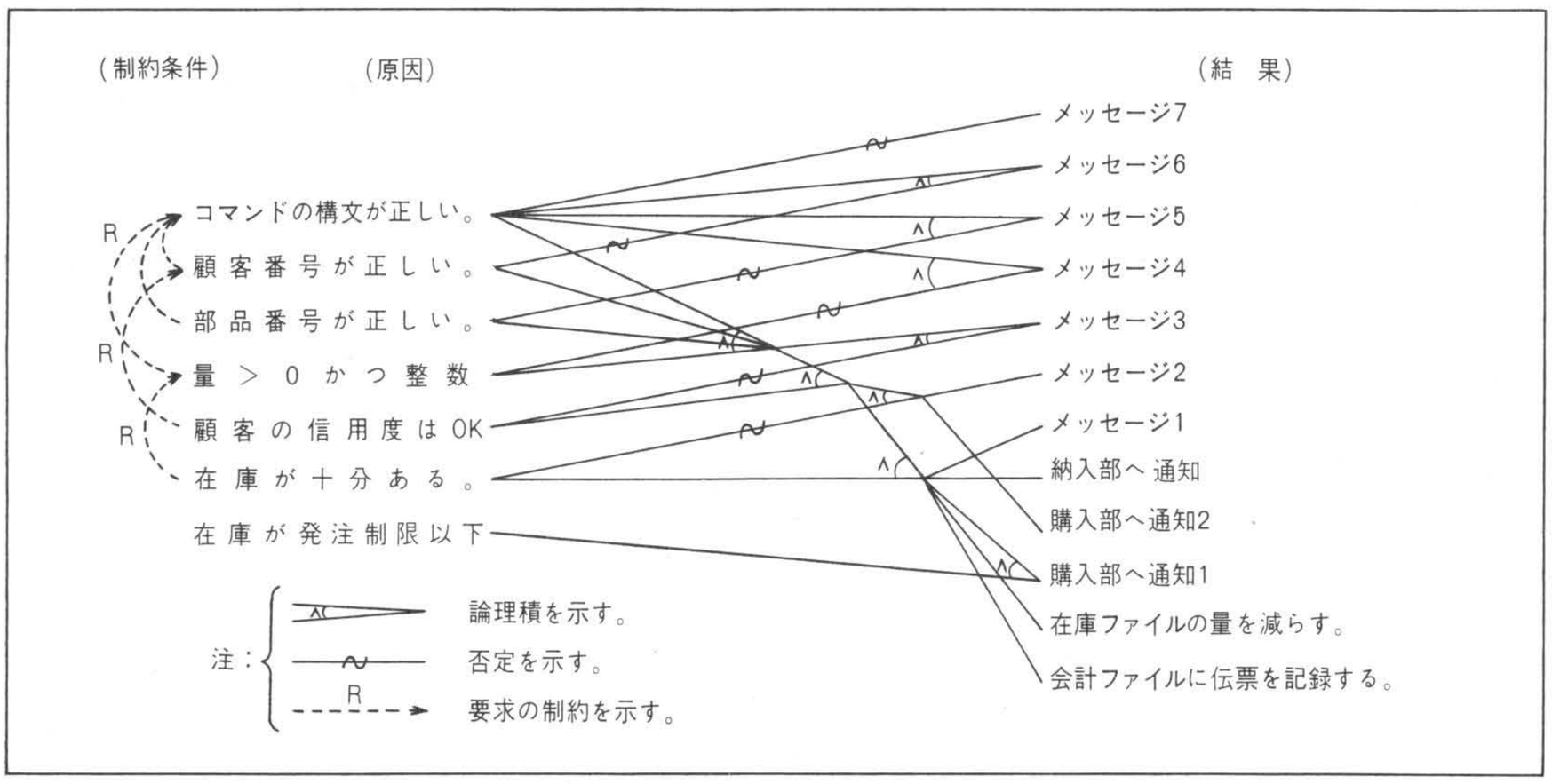
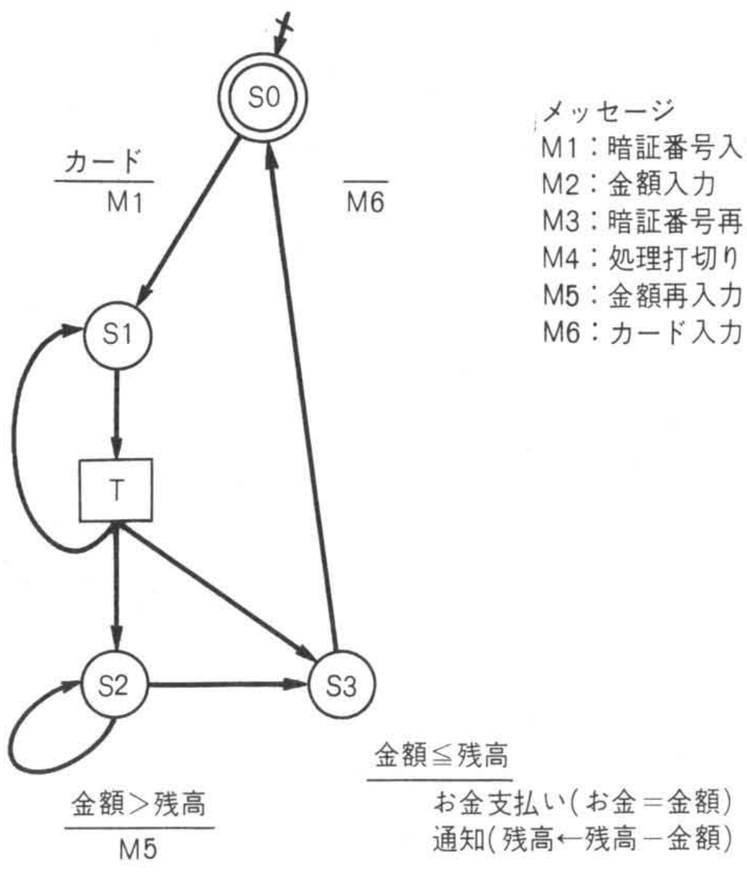


図2 原因結果グラフの例 原因結果グラフによる部品納入コマンドの機能仕様の例を示す。



(a) 状態遷移部

決定表T：

E 入力	暗証番号登録と一致	Y	N	N
	暗証番号3回誤り	N	Y	N
出力	M2	*		
	M3			*
	M4		*	
	登録取消し		*	
遷移先	S1			*
	S2	*		
	S3		*	

(b) 論理関係部

図3 機能図式の例 機能図式による現金自動支払機の機能仕様の例を示す。

4　支援システム

支援システムは，機能図式から漏れや重複のないテスト項目を自動的に生成する。支援システムの構成を**図4**に示す。各構成成分の概要は，以下に述べるとおりである。

(1)　入　　力

テキスト形式で記述された機能図式を入力し，状態遷移部と論理関係部に分離し，同時に，機能レビュー用ドキュメントとして，ソースリストのほかにマトリックス形式の決定表リスト，状態遷移表リストなどを出力する。**図5**に現金自動支払機の機能図式に対するソースリストを示す。

(2)　状態遷移の構造化

状態遷移部については，より強力なテスト基準で遷移経路を網羅するために，連結，選択，反復という3種類の構造だけから成る構造化状態遷移に変換する。

(3)　部分テスト項目の作成

論理関係部については，各状態での条件の組合せを効果的に確認するのに必要かつ十分な部分テスト項目を作成する。

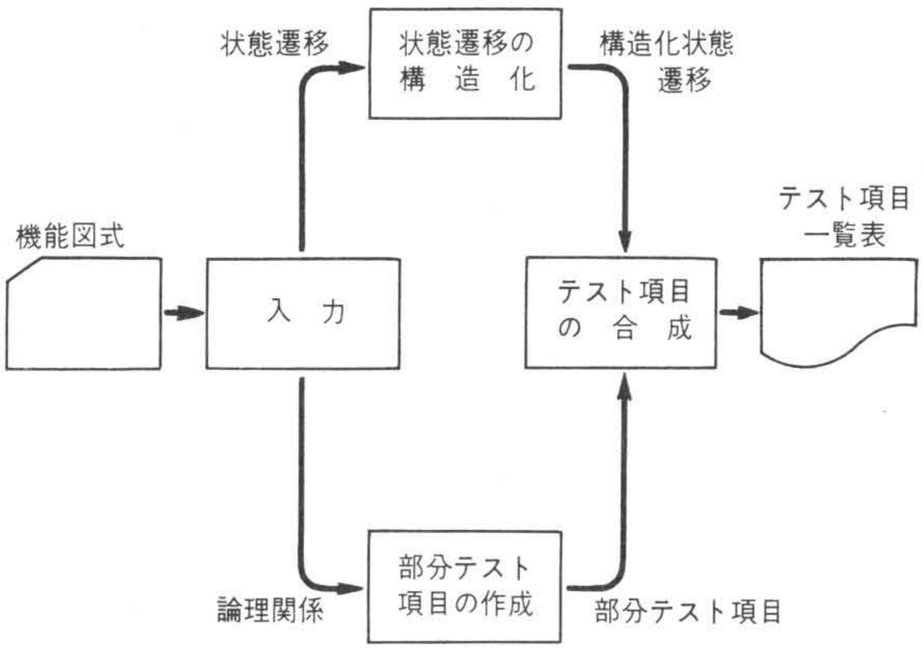


図4　支援システムの構成　　支援システムは，入力した機能図式を状態遷移部と論理関係部に分離し，構造化状態遷移と部分テスト項目を作成した後，テスト項目を合成する。

```
**SOURCE LIST**
  SEQ.    SOURCE LIST
          ----------1----------2----------3----------4----------
    1     Q+T/
    2 TITLE CD='ゲンキンジドウシハライキ'
    3 STATE (S0)
    4   NODE CAUSE C0='キャッシュ カード'
    5        EFFECT M1='アンショウ バンゴウ ニュウリョク'
    6   DECISION
    7        C01=(C0)/M1->S1
    8 STATE (S1)
    9   NODE CAUSE   AA='アンショウ バンゴウ トウロクズミ'
   10                AB='アヤマリ=3カイ'
   11        EFFECT  M2='キンガク ニュウリョク'
   12                M3='アンショウ バンゴウ サイニュウリョク'
   13                M4='ショリウチキリ'
   14                OC1='カード トウロクトリケシ'
   15   DECISION
   16                C11=(AA NOT AB)/M2->S2
   17                C12=(NOT AA AB)/M4 OC1->S3
   18                C13=(NOT AA NOT AB)/M3->S1
   19   CONST
   20                U11=AA AB EXCLUSIVE
   21 STATE (S2)
   22   NODE CAUSE   KA='キンガク<=ザンダカ'
   23                KB='キンガク>ザンダカ'
   24        EFFECT  M21='キンガク サイニュウリョク'
   25                FR='ツウチョウ ハッコウ'
   26                FR1='ザンダカ<-ザンダカ
                         ='オカネ シハライ'
```

図5　ソースリストの例　　現金自動支払機の機能図式に対するソースリストを示す。

TRANSITION TAIL	TRANSITION HEAD	KIND	NODE	MEANING	＊テスト＊ケース＊ID＊ 1	2	3	4
S0	S1	I	C0	キャッシュ カード		0	0	0
		O	M1	アンショウ バンゴウ ニュウリョク		0	0	0
S1	S1	I	AA	アンショウ バンゴウ トウロクズミ			X	
			AB	アヤマリ=3カイ			X	
		O	M2	キンガク ニュウリョク			X	
			M3	アンショウバンゴウ サイニュウリョク			0	
			M4	ショリウチキリ			X	
			OC1	カード トウロクトリケシ			X	
S1	S3	I	AA	アンショウ バンゴウ トウロクズミ		X		
			AB	アヤマリ=3カイ		0		
		O	M2	キンガク ニュウリョク		X		
			M3	アンショウ バンゴウ サイニュウリョク		X		
			M4	ショリ ウチキリ		0		
			OC1	カード トウロクトリケシ		0		
S3	S0	I	*			0		
		O	M7	カード ニュウリョク		0		
S1	S2	I	AA	アンショウ バンゴウ トウロクズミ			0	0
			AB	アヤマリ=3カイ			X	X
		O	M2	キンガク ニュウリョク			0	0
			M3	アンショウ バンゴウ サイニュウリョク			X	X
			M4	ショリウチキリ			X	
			OC1	カード トウロクトリケシ				
S2	S2	I	KA	キンガク<=ザンダカ				
				キンガク>ザンダカ				
				サイニュウリョク				

図6　テスト項目一覧表の例　　現金自動支払機の機能図式に対するテスト項目一覧表を示す。

(4) テスト項目の合成

構造化状態遷移から，選択についてはそれぞれの場合を確認し，反復については反復しない場合と反復する場合の二通りを確認するテスト経路を作成する。このテスト経路の各状態に遷移先がテスト経路の次の状態と一致する部分テスト項目を割り当てて全体のテスト項目を合成する。

図6に現金自動支払機の機能図式に対するテスト項目一覧表を示す。遷移(TRANSITION)は，遷移元の状態(TAIL)と遷移先の状態(HEAD)の対で示され，遷移元の状態での条件(NODE)と意味(MEANING)が入力(I)と出力(O)に分けて付加される。各テスト項目(この場合には4個)は列に示されており，空白はその遷移を通らないこと，○は成立，×は不成立を示す。テスト項目1は，どの遷移も通らないことを示しているが，これは初期状態(S0)と終了状態(S0)が一致しているために生成されたものである。

5 適用効果

本技法及び支援システムは，完成以来多くのシステムの機能テストに適用され，効果を挙げてきた。各種の条件下で適用されたので単純に結論を出すことはできないが，おおむね以下に述べるようなことが分かっている。

(1) 信 頼 性

条件の組合せを網羅するテスト項目の作成，特に，人間にとっては考えにくい異常ケースのテスト項目の強化，機能仕様の形式化段階での不良発見，などにより信頼性が向上した。

(2) 生 産 性

従来法によって，どの程度十分なテスト項目を作成していたかによるが，同等の品質を保証するという前提で比較した場合には，テスト項目，特に，正常ケースのテスト項目の圧縮によりテスト実施作業の削減が図れた。

(3) 管 理 性

従来法によるテスト項目の作成は，ほとんど経験者によって行なわれていたが，新人でも機能仕様を機能図式で表現することによって，品質の良いテスト項目を作成することができるようになり，新人の早期戦力化に効果があった。

6 結 言

ソフトウェアのテスト作業は，極めて属人的な色彩が濃く，ソフトウェアの開発作業の中で，最も強く合理化が望まれている部分である。今回開発した技法は，このようなテストに科学的な方法を導入することにより，経験や直観に頼らずに，漏れや重複のないテスト項目を系統的に作成することを可能にするものである。今後，更に適用の拡大を図り，ソフトウェアの信頼性向上に努めていく考えである。

最後に，本技法の開発に当たり，御協力をいただいた筑波大学教授 中田育男理学博士に厚く謝意を表わす次第である。

参考文献

1) G.J. Myers(松尾訳)：ソフトウェア・テストの技法，近代科学社(昭和55年)
2) 手工業的手法からの脱皮を目指すソフトウェア・テスト，日経エレクトロニクス，No.286(1982)
3) 古川，外：機能テストのためのテスト項目作成手法について，情報処理学会，ソフトウェア工学，16-2(1980)

論文抄録

ルール記述に基づくシステム制御方式

日立製作所 田代 勤・薦田憲久・他1名

計測と制御 22—9，786～790（昭58-9）

本稿は小特集「知識工学」の中で，制御方式の記述に「条件→結論」型のルールを使用する計算機制御について述べたものである。

ジョブショップ・製鉄所などの設備の自動運転，石炭ヤード・コンテナヤードなどの運用制御，自動倉庫のラック・クレーン割当てなどの制御では，(1)システムの建設が段階的に行なわれる，(2)扱う品目や量に従いシステム構成が変更される，(3)稼動率向上，省エネルギー化などのために，実稼動経験に基づきシステム改良が行なわれる，といったことから，制御方式が頻繁に変更されるという特徴がある。このような状況下で，制御方式をFORTRAN，PL/Iなどの汎用言語を用いて制御プログラム化する現在の計算機制御システムでは，プログラム開発がネックとなり頻繁に発生する制御方式の変更に迅速に追従することが難しくなってきている。

これに対処するため，システムの状態の組合せを条件とし，その条件が満足された場合に出すべき制御指示を結論として記述したルールによって制御方式を記述する方法が従来から試みられている。この代表的な例に，ルールをテーブル状に整理して与えるDecision Table方式，ルールを文字列表現で与えるRule Based Language方式，あいまい表現のルールを用いるFuzzy Control方式がある。これらの方式では，現状のシステム状態を満足するルールが自動的に選択され制御指示が決定されることから，ユーザーは処理手続を全く記述する必要がなく，プログラミングの専門知識を要しない，プログラム(ルールの集合)の段階的開発に強い，という特徴をもっている。しかし，大規模なシステムの制御に用いようとする場合，(1)ルールに指定すべき条件数が多数となる，(2)全ルール数がかなり多くなる，などの問題が発生し，適用対象の規模に限界がある。

これに対し，近年，ばらばらに記述した判断のルールを必要に応じて自動的に組み合わせ，所望の結論を得るという人工知能でのプロダクションシステムの手法を用いて制御指示を決定する方法が注目されている。この手法では，一階述語論理に近い変数を含むルールが使用でき，これによって，ルールの記述性，理解性が飛躍的に向上する。従来プロダクションシステムは主にコンサルテーションの場で用いられているが，制御に用いる場合，ルールの論理的完全性の保証，処理速度の向上が重要な課題となってくる。前者に対してはまだ十分な解決が得られているとは言えないが，後者に対しては，ワークメモリの検索をポインタを用いて行なう方式により実時間制御に必要な処理速度を確保したものが最近出てきている。

プログラム非専門家でも使いこなせる計算機制御装置という観点から，ルールのテスト・デバッグを支援するルール形制御用のプログラミングシステムの発展が待たれているところである。