U.D.C. 681.32.06.002:[658.52.011.56:681.322]

ソフトウェア開発支援システム(CASDシステム)

Computer Aided Software Development System

片岡雅憲* *Masanori Kataoka*
葉木洋一* *Yôichi Hagi*
野木兼六** *Kenroku Nogi*

ソフトウェアの大規模化，複雑化に対処するためには，従来の手工業的な開発方式では限界があり，コンピュータの高度利用による自動化技術を核として，ソフトウェア生産技術の抜本的革新を図る必要がある。日立製作所は，汎用コンピュータ用ソフトウェアの生産技術の革新を目的とした，総合的なソフトウェア開発支援システム(CASDシステム)を開発し，その一部を実用化した。

CASDシステムは，設計支援，プログラミング支援及びテスト支援の三つのサブシステムから構成されており，サブシステム相互が密接に接続された総合的，かつ一貫したシステムである。また，各々のサブシステムは高度の自動化機能をもつ治工具群から成っている。

CASDシステムは，ソフトウェアの信頼性，生産性向上の両面で，今後大きな効果を発揮するものと期待される。

1 緒　言

近年，コンピュータの利用形態の高度化・多様化に伴い，ソフトウェアは急激に大規模化，複雑化してきた。従来の手工業的な開発方式でこの変化に対処するには限界があり，ソフトウェア生産技術の抜本的革新が要求されている。

日立製作所は，汎用コンピュータ用ソフトウェアの生産技術の革新を目的とした，CASDシステム(Computer Aided Software Development System：ソフトウェア開発支援システム)を開発し，その一部については既に実用化した。

本論文では，CASDシステムの開発背景，ねらい及びシステム構成について概説するとともに，CASDシステムを構成する代表的な治工具を紹介する。

2 CASDシステムの概要

2.1 CASDシステムの開発背景

日立製作所では，従来からソフトウェア生産技術を生産管理技術及び生産支援技術の両面から改善してきた。前者は，所要工数計画，原価・工程・品質管理などの標準技術から成る[1]。後者は，構造設計法[2]と呼ばれる標準設計法，システム記述用高級言語，コンピュータの多重・遠隔利用技術，ライブラリの集中管理技術などから成る。

上記の生産技術の改善は，ソフトウェア開発の標準化及び効率向上に大きな効果をもたらしたが，ソフトウェア開発が手工業的である点を本質的に解決するには至らなかった。ソフトウェアの急激な大規模化・複雑化に対処するには，現在，手作業で行なわれている部分を極力自動化して，近代工業化を図る必要がある。従来もこの目的に沿って多くの治工具が開発されてきたが，自動化の対象範囲が部分的であったり，汎用性に欠けていたため，必ずしも十分な効果を発揮できなかった。

自動化を本格的に推進し，ソフトウェア開発を近代工業へ脱皮させるには，総合的かつ標準的な治工具システムを必要とする。CASDシステムは以上の背景のもとに開発したものである。

2.2 CASDシステムの開発方針

CASDシステムの最終目標はソフトウェア開発の近代化にあり，この目標達成のため，次に述べるような開発方針を設定した。

(1) ソフトウェア開発の各工程の作業を標準化し，この標準化を前提とした支援治工具により省力化する。支援治工具は，既存の個別治工具の機能を包含した高度なものとする。

(2) ソフトウェアの全工程を一貫する総合的なシステムとする。すなわち，CASDシステムに含まれる各治工具は，一貫した思想のもとに体系化し，相互に密接な関連をもつものとする。

2.3 CASDシステムのシステム構成

CASDシステムは，**図1**に示すように設計支援，プログラミング支援及びテスト支援の三つのサブシステムから構成され，ソフトウェア開発の全工程を対象としている。各サブシステムは相互に密接に接続されており，また各サブシステムを構成する治工具も相互に密接に接続され一貫したものとなっている。

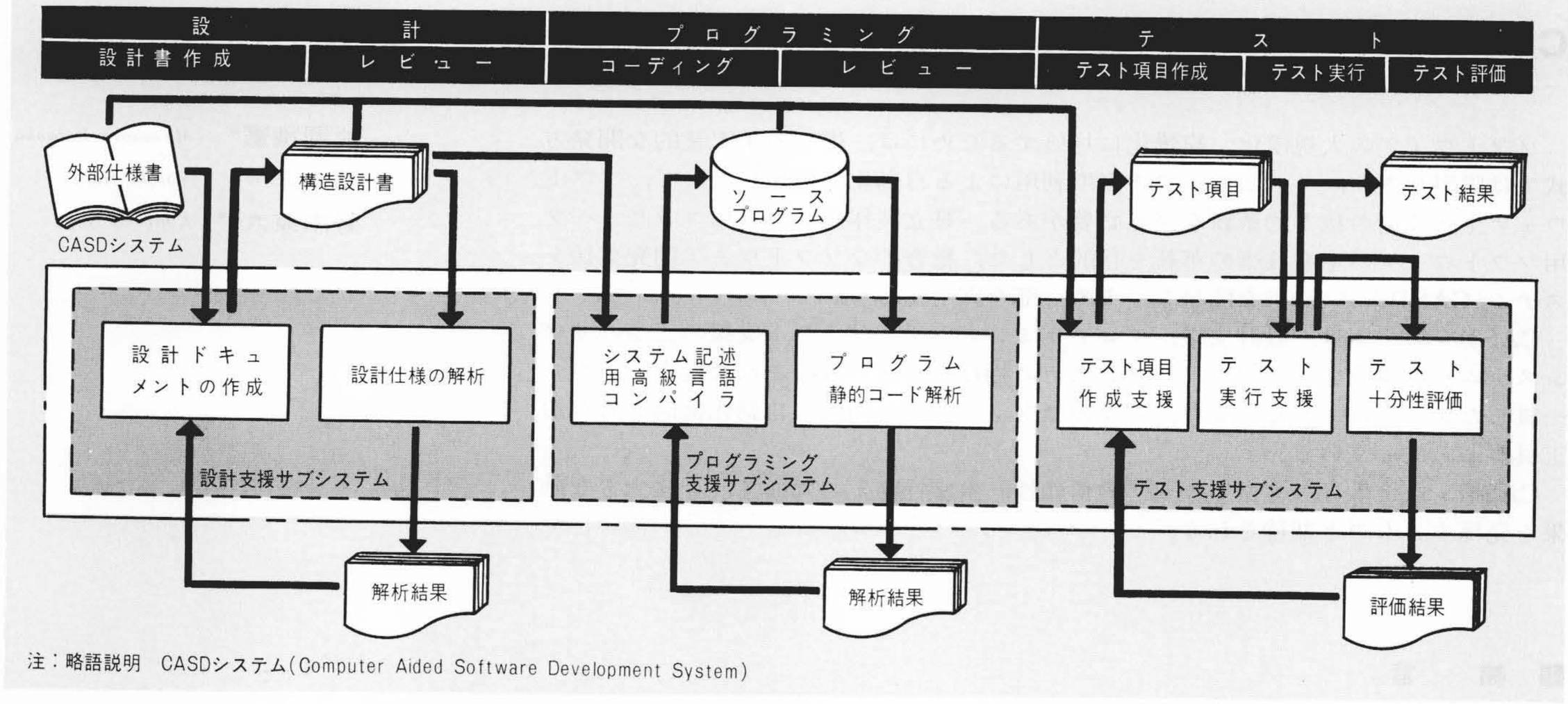
設計支援サブシステムは，設計ドキュメントを作成し，設計仕様を解析する。プログラミング支援サブシステムは，システム記述用高級言語によるプログラムの作成を可能とし，その結果を解析する。また，テスト支援サブシステムは，テスト項目の作成を支援し，被テストプログラムの実行を支援するとともに，テスト後にテスト十分性の評価を行なう。

3 設計支援サブシステム

設計支援サブシステムは，構造設計法[2]と呼ばれる標準設計法に基づいた設計支援治工具ADDS(Automated Design and Documentation System：構造設計支援システム)[3]で実現されている。ADDSは，MDL(Module Design Language：モジュール仕様記述言語)と，**図2**に示す処理系から構成され，その特長は次に述べるとおりである。

(1) 従来，人手により自然語や図表であいまいに記述されていた設計仕様を形式化し，MDLにより正確に記述可能とする。

* 日立製作所ソフトウェア工場　** 日立製作所システム開発研究所



注：略語説明 CASDシステム(Computer Aided Software Development System)

図1 CASDシステムのシステム構成 ソフトウェア開発工程とCASDシステムの三つのサブシステム(設計支援，プログラミング支援，テスト支援)の関係を示す。

(2) 設計仕様情報を設計仕様情報データベースに格納し，随時，修正，変更を可能とする。

(3) 設計仕様の形式的な誤りを自動的に検出し，設計レビューを支援する仕様解析結果を出力する。

(4) ラインプリンタ，レーザビームプリンタ及び端末上に，多種類の設計ドキュメントを自動出力する。特にレーザビームプリンタ上には，ハードウェアの高品質印字特性を利用して，高度の視覚化を図った図表形式ドキュメントを出力する。

ADDSが出力する設計ドキュメントを**表1**に示す。

4 プログラミング支援サブシステム

プログラミング支援サブシステムは，プログラムをHPL(Hitachi Programming Language)と呼ばれるシステム記述用高級言語でコーディングし，コーディング結果の効率良いレビューを可能とすることを目的とする。このサブシステムは，**図3**に示すようにHPLコンパイラと，SCAN(Static Code Analysis：静的コード解析)システムの二つの治工具から構成される。

コーディングレビューは，プログラムの初期不良を早期に検出するとともに，プログラムの論理を十分に理解する上で極めて重要な作業である。しかし，従来当作業は手作業で行なわれていたため，手間がかかる，個人間の技量差による影響が大きい，などの問題があった。SCANシステムは，**図3**に示すようにHPLコンパイラが出力する静的コード解析情報を入力として，コーディングレビューを支援する各種レポートを出力するもので，次に述べるような特長をもつ。

(1) プログラムの制御構造を解析し，図形式で出力する。

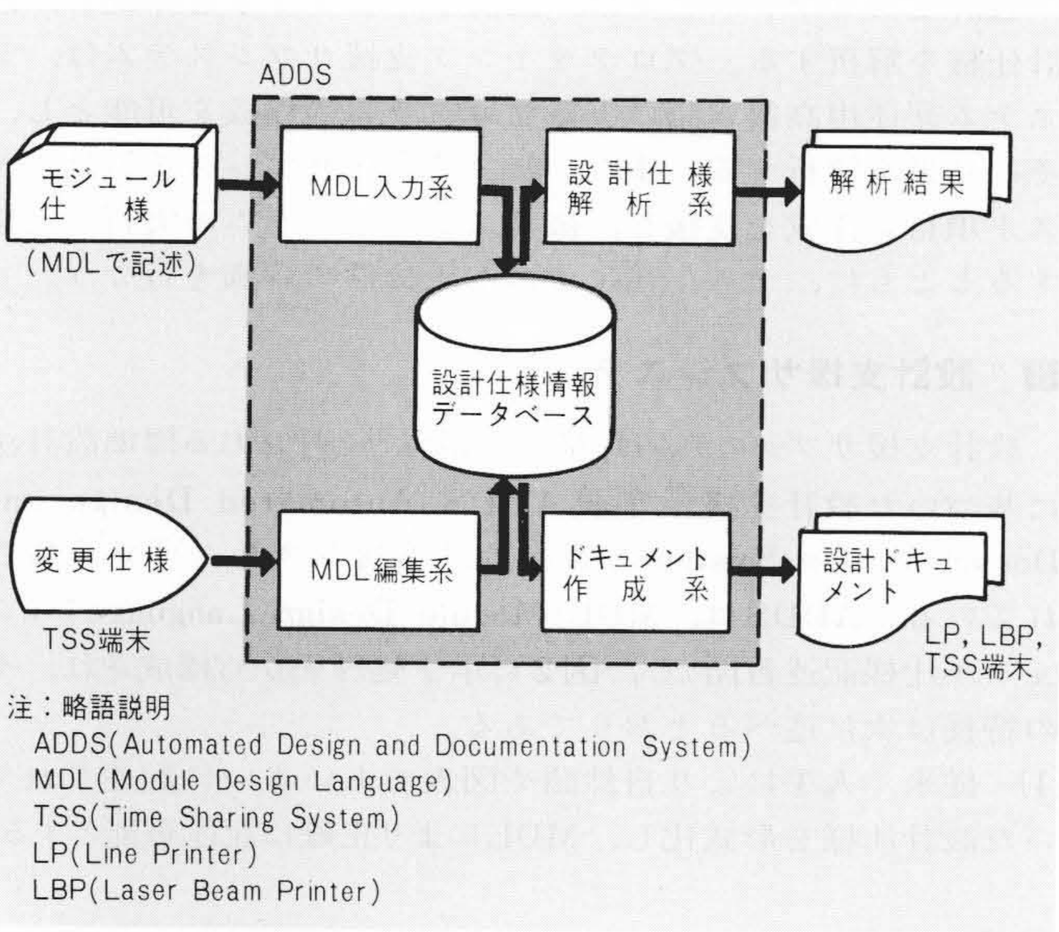
(2) データ構造の解析，すなわちプログラムのデータ定義部を解析し，テーブル仕様書(データの構造と使用法を規定する仕様書)を出力するとともに，個々のデータの流れ(生成，参照，更新，削除の順序関係)を解析する。

(3) モジュール間の制御の移行関係，モジュールとデータとの相互関係を解析する。

(4) 上記(1)～(3)の解析結果が，ADDSにより作成された設計情報と整合しているかどうかをチェックする。

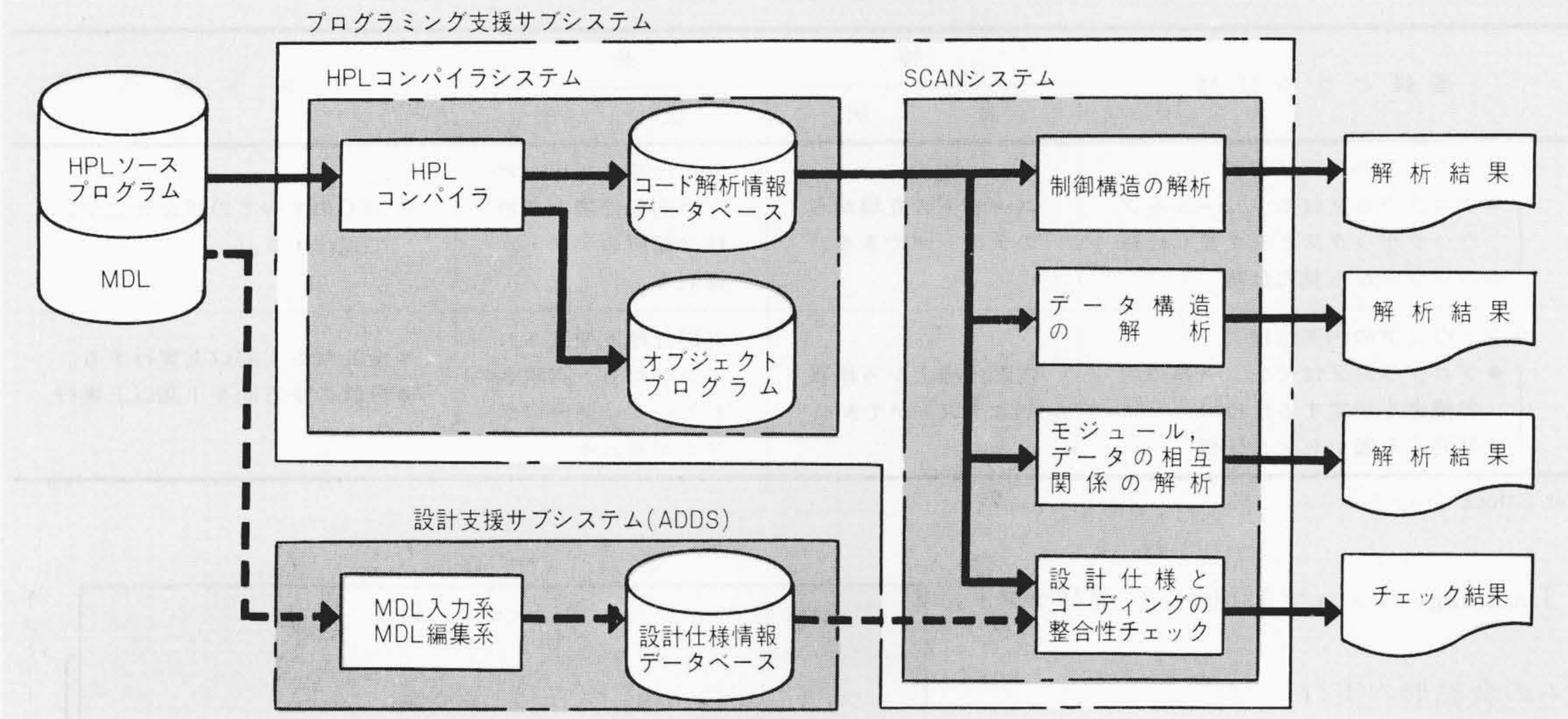
表1 ADDSが出力する設計ドキュメント ADDSは多様な設計ドキュメントを図表形式で出力する。

No.	ドキュメント名	形式	内容
1	機能階層構造	図・表	各モジュールの階層構造を示す。
2	モジュール仕様	表	各モジュールの仕様を示す。
3	データフロー	図	モジュールの入出力と処理手順を図示する。
4	モジュール関連	図・表	モジュール間の制御関係を示す。
5	モジュール一覧	表	全モジュールの概要一覧を示す。
6	機能一覧	表	各モジュールの機能一覧を示す。
7	変更歴一覧	表	各モジュールの変更履歴一覧を示す。



注：略語説明
ADDS(Automated Design and Documentation System)
MDL(Module Design Language)
TSS(Time Sharing System)
LP(Line Printer)
LBP(Laser Beam Printer)

図2 ADDS処理系の構成 ADDSはMDLで記述された設計仕様を入力として，これを解析し，また，設計ドキュメントを作成する。



注：略語説明
HPL(Hitachi Programing Language)
SCAN(Static Code Analysis)システム

図3 プログラミング支援サブシステムの構成 HPLソースプログラムを静的にコード解析し，コーディングレビュー時のレビュー支援情報を出力する。

5 テスト支援サブシステム

ソフトウェアのテスト作業は，信頼性向上及び生産性向上の両面から改善する必要がある。前者の面からは，潜在する不良を着実に摘出するテスト技術を必要とし，後者の面からは，現在，ソフトウェア開発作業の工数の約半分を費やしているテスト作業の効率を向上させる必要がある。以上の観点からテスト支援サブシステムは**図4**に示す構成とし，次の方針のもとに開発を行なった。

(1) 設計仕様の厳密な解釈を行なわせる。

本来，テストとは設計仕様とプログラムとの整合性を検証することを目的としている。したがって，厳密なテストを行なうためには設計仕様の厳密な解釈に基づいたテスト項目が必須であり，このための系統的なテスト項目作成技術を開発する。

(2) 明確なテスト基準を設定する。

プログラムの実行順序の組合せは天文学的な数となり，そのすべてをテストすることは不可能である。この中から実用上十分なソフトウェア品質を保証し，かつ実施可能なテスト項目を抽出するためのテスト基準を設定する。テスト基準は**表2**に示すように外部仕様と内部仕様の両面から設定する。

(3) テスト作業の省力化を推進する。

テスト作業をできるだけ自動化し，省力化する。

(4) テスト十分性の評価を行なう。

テスト実行後，これが**表2**の内部仕様テスト基準の面から十分であるかどうかを評価する。

5.1 テスト項目作成支援

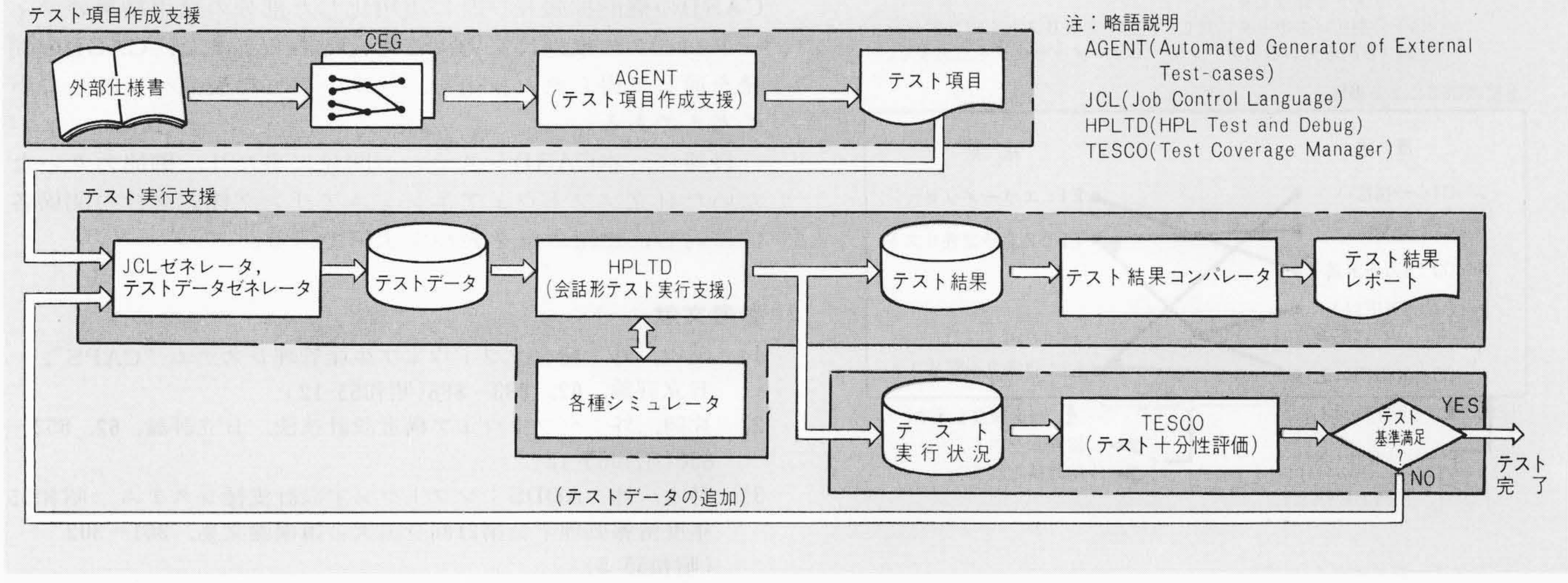
AGENT(Automated Generator of External Test-cases：テスト項目作成支援治工具)は，外部仕様に基づくテスト項目を自動的に作成する治工具である。AGENTを用いるには，まず外部仕様書(ユーザーマニュアルなど)に基づいて外部仕様を厳密に解釈し，これをグラフ表現したCEG[4](Cause and Effect Graph：原因結果グラフ)を作成し，次にCEGの情報をAGENTに入力する。

AGENTの利用によりテスト項目が系統的に作成できる。またCEGの作成過程やAGENT内での自動チェックにより，外部仕様のあいまいな点や不完全な点が摘出される。

図5に外部仕様の例とそのCEGによる表現を，**図6**にAGENTの出力例を示す。

5.2 テスト実行支援

テスト支援サブシステムでは，被テストプログラムの実行を効率良く行なうために，以下の支援機能をもつ(**図4**参照)。



注：略語説明
AGENT(Automated Generator of External Test-cases)
JCL(Job Control Language)
HPLTD(HPL Test and Debug)
TESCO(Test Coverage Manager)

図4 テスト支援サブシステムの構成 テストは，外部仕様からテスト項目を作成し，実行する。実行結果は，テスト十分性評価で内部仕様面から評価し，テスト基準を満足するまでテストデータを追加し，再実行を繰り返す。

表2 外部仕様テストと内部仕様テスト テストは，外部仕様と内部仕様の両面から行なう必要がある。

No.	テスト分類	基礎となる仕様	特長		テスト基準
			長所	短所	
1	外部仕様テスト	ソフトウェアの外部仕様 ●プログラム又はモジュールをブラックボックスとして見た仕様 ●ユーザーから見た仕様	ユーザーの立場からのテストができる。	外部仕様には現われない内部仕様上の特殊な箇所のテストが漏れる。	●CEGのすべての組合せについて実行する。
2	内部仕様テスト	ソフトウェアの内部仕様 ●プログラム又はモジュールの内部構造を規定する仕様 ●開発する者から見た仕様	内部仕様上から網羅的にテストができる。	外部仕様上規定されていながら，実現されていない部分のテストが漏れる。	●全命令を1回以上実行する。 ●分岐の全方向を1回以上実行する。

注：略語説明 CEG(Cause and Effect Graph)

(1) JCL(Job Control Language：ジョブ制御言語)及びテストデータの作成
(2) 被テストプログラムの会話形の実行
(3) 各種シミュレータの提供
(4) テスト結果の自動照合

上記(1)の面からはJCLジェネレータ及びテストデータジェネレータがある。(2)の面からはHPLTD(HPL Test and Debug)と呼ぶ治工具により，HPLで記述された被テストプログラムを会話形で実行し，制御することを可能としている。また，HPLTDは各種テスト環境の設定を支援し，(3)の各種ハードウェア及びソフトウェアのシミュレータの組込みを容易にしている。一方，(4)の面からはテスト結果コンパレータがあり，多様なテスト結果を多様な方式で比較することを可能にしている。

5.3 テスト十分性評価

TESCO(Test Coverage Manager)は，テスト実行後にテストが十分に行なわれたかどうかを，**表2**の内部仕様基準に基づき評価する。**図4**に示すように，TESCOはテスト実行中の被テストプログラムの実行状況を監視・記録して，これを解析することによりテストされていない命令や分岐条件を摘出する。テスト作業者は，テストされていない部分に対してはテストデータを追加することにより，テスト基準に沿った系統的なテストを実施できる。

〔外部仕様の例〕
1万円以上の取引データに対し，次の処理を行なう。
(1) 取引額10万円以上で一括払いのデータに対し，得意先一覧リストを作成する。
(2) 分割払いのデータに対し，入金予定表リストを出力する。
(3) 1万円未満の取引データは，エラーメッセージを出力する。

上記のCEGによる表現

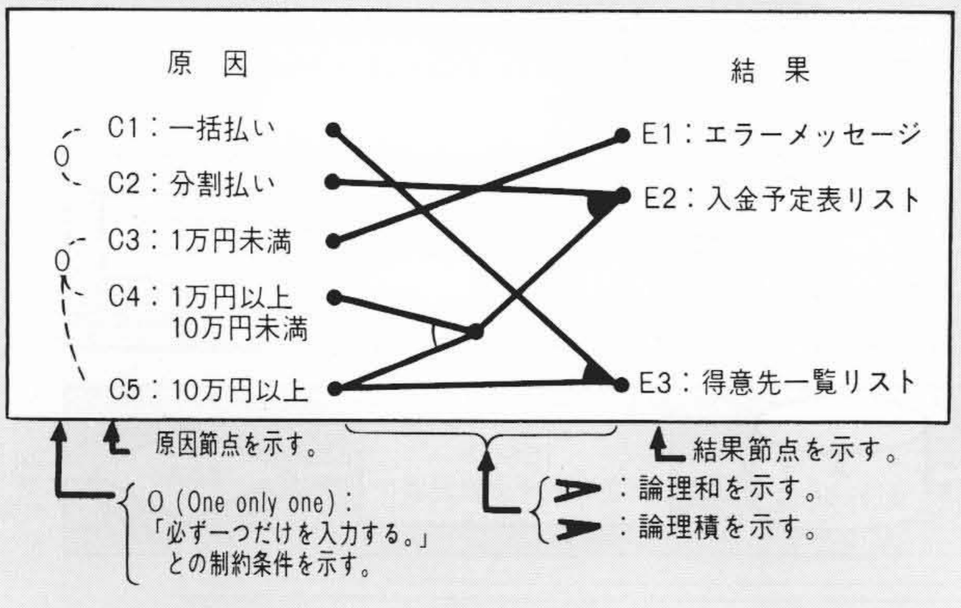


図5 外部仕様の例とそのCEGによる表現 外部仕様を原因と結果に分け，両者の論理関係をグラフ表現したものがCEGである。

```
     ***** TEST  CASE  SPECIFICATIONS *****
** TEST  CASE      1 **
  * CAUSE   NODE#  NAME      C  DESCRIPTION
                    1  C1     F ¬(イツカツ  バライ)
                    2  C2     T   ブンカツ  バライ
                    3  C3     F ¬(¥<10,000)
                    4  C4     T   10,000 <= ¥ < 100,000
                    5  C5     F ¬(¥ >= 100,000)
  * EFFECT  NODE#  NAME      C  DESCRIPTION
                    8  E1     F ¬(エラー  メツセージ)
                    9  E2     T   ニユウキン ヨテイヒヨウ リスト
                   10  E3     F ¬(トクイサキ  イチラン  リスト)
** TEST  CASE      2 **
  * CAUSE   NODE#  NAME      C  DESCRIPTION
                    1  C1     T   イツカツ  バライ
                    2  C2     F ¬(ブンカツ  バライ)
                              T   ¥ < 10,000
```

テストケース(テスト項目)1の内容を定義している。
図5の節点名と対応
¬：NOT記号を示し，( )内の意味を否定する。
T：節点の状態を成立させることを示す。
F：節点の状態を成立させないことを示す。

図6 図5のCEGに対するAGENT出力例 テスト項目1は次のテストを行なうべきことを表わしている。
(1) 分割払いで，1万円以上10万円未満のデータを入力する。
(2) 結果は入金予定表リストだけが出力される。

6 結 言

CASDシステムは，ソフトウェア開発の全工程を対象としていることによる総合性と，これを構成する各サブシステムが相互に密接な関連をもつことによる一貫性に特長があり，ソフトウェア開発の近代工業化の観点から，大きな効果を発揮するものと期待される。

今後，更に，ソフトウェア開発の近代化を促進するために，CASDの機能拡張及び既に実用化した部分の適用結果のフィードバックの推進に努力する考えである。また，CASDの開発を通じて得られた技術を，ソフトウェア製品へ反映していく考えである。

最後に，本CASDシステムの開発に当たり，御協力をいただいた日立ソフトウェアエンジニアリング株式会社の関係各位に対し，感謝の意を表わす次第である。

参考文献

1) 芝田，外：総合ソフトウェア生産管理システム“CAPS”，日立評論，**62**，883～888(昭和55-12)
2) 片岡，外：ソフトウェア構造設計技法，日立評論，**62**，853～856(昭和55-12)
3) 野木，外：ADDS：ソフトウェア設計支援システム，昭和55年度情報処理学会第21回全国大会講演論文集，301～302(昭和55-5)
4) G.J. Myers著，有澤訳：ソフトウェアの信頼性，近代科学社(昭和52-10)