解 説

3.1 システムソフトウェアを対象にしたCASEの現状と動向†

前 澤 裕 行††

1. はじめに

計算機ハードウェアの高速化と大容量化にともないソフトウェアの大規模化，複雑化，多様化が急速に進み，ソフトウェア開発の効率化が課題となっている．また，計算機システムの事故は大きな影響を与えるため，高度な信頼性が要求されている．このため，高信頼のソフトウェアを効率よく開発するための生産技術がますます重要となっている．

ソフトウェアの生産技術はいくつかの区切りを経て発展してきた．1960年代に名人プログラマによる手作りの時代が終わり，属人性を排した均質化のための技術としての構造化プログラミング技法が生まれた．70年代には，ライフサイクルモデルに基づいて，要求定義に代表される上流工程の重要性が認識され，種種の計画，設計技法とその支援ツールが開発された．ソフトウェアエンジニアリング誕生の時代である．80年代は，実用化のために，それまでに開発された方法論，技法，ツールの統合化を行う CASE (Computer Aided Software Engineering) の時代に入っている．現在では商用の CASE 製品も数多く登場している．

一口に CASE と言っても対象とするソフトウェアの性質やユーザによって内容はさまざまである．事務処理用，制御用，機器組み込みマイコン用，などソフトウェアの種類によって CASE も異なる．ソフトウェアの規模に応じても CASE の使い分けが必要である．ベテラン用と初心者用も異なる．CASE の選択に当たっては，対象となるソフトウェアの性質と開発組織の特徴に合ったものを選択することが必要である．

† CASE for System Software Development by Hiroyuki MAEZAWA (Hitachi Ltd., Systems Development Laboratory).

†† (株)日立製作所システム開発研究所

ここでは，オペレーティングシステム，通信ソフトウェア，データベースなどのシステムソフトウェアを対象とした CASE の現状と動向について述べる．システムソフトウェアは事務処理や制御用の各種応用ソフトウェアが動くための土台となるものであり，構造は複雑でしかも高い信頼性が要求される．また，過去開発した応用ソフトの動作を保証するための互換性も要求される．

2. システムソフトウェア CASE への要件

2.1 システムソフトウェア開発における問題

システムソフトウェアの開発は，多人数で長期間にわたって行われる．スケジュールと分担を決めて開発が進められるが，期間中に人の移動により分担の変更が生じたり，一工程の小さな遅れが全体工程を狂わせたりする．開発規模が大きく多人数のため，人と人，物と物の間のインタフェースに係わる不良が発生する．たとえば，仕様変更の連絡が不徹底であったり，出来上がったソフトウェアを組み合わせると動かないという具合である．開発者の熟練度はまちまちであり，設計品質にも差がでる．また，プロジェクトのどこでどんな問題が発生しているのかの把握が難しく，把握したときは手遅れということも生じる．

ソフトウェアの性質からみると，大規模でしかも構造は複雑であり，高速性が要求される．プログラム全体の構造を把握するのが難しく，個々の設計も難しい．設計の途中で仕様変更を余儀なくされることも多い．信頼性を得るためにレビュに多くの時間を費やす．テストの設計は難しく，テストの十分性の判定を誤ると大きな事故の原因となる．また，実際の利用環境に近い状態でテストを行うためには準備に多くの時間が必要となる．

システムソフトウェアは，機能の追加や改造が何回も繰り返される．改造の担当者は，設計にあたって，

前任者の作ったソフトウェアの構造を理解するために多くのドキュメントやソースコードを読まねばならない．また，互換性を保証するための制約の見極めも必要となる．

このような特徴と問題からシステムソフトウェア用の CASE には次の要件を満たすものが必要となる．

2.2 CASE の要件

システムソフトウェアの大規模で複雑かつ，高信頼性を要求されるという性質から，これに適した設計，レビュ，テストのための技法が必要である．

（1） 設 計 支 援

種々の熟練度の設計者が混在した開発体制で複雑な構造のソフトウェアを設計するためには，構造化が必要である．大きなソフトウェアを小さな単位に分割して，その間の関係を明確に定義することにより，複雑な構造はより簡単な構造へと分解することが可能となる．また，分割する際の基準を設けることにより，設計における属人性を排除することができる．ドキュメントの記述内容を曖昧性のないものにするために並列性，実時間性などを記述するための手段が必要である．

（2） レビュ支援機能

設計レビュは，それが要求を満たしているか，正しい動作が保証されているかの二つの面から行う必要がある．また，静的構造だけでなく動的構造についてのレビュがシステムソフトウェアにおいては特に重要である．ドキュメントを用いて行う従来のレビュ方法に代わり，プロトタイピングやシミュレーションを用いたレビュが必要である．

（3） テスト支援機能

テストケースの設計機能，テストデータの作成機能，テストの十分性評価機能などのほかにシステムソフトウェア固有の機能が必要である．OS のテストでは特権モードでの実行比率が高いため実計算機または仮想計算機を用いる．このため OS の開発システムと，テスト用のシステムが異なるためテスト，デバッグの効率が他のソフトウェアに比べて悪いという問題を解決することが必要である．

（4） 機能拡張のための支援機能

長期にわたって継続的に行われる機能拡張に対して，母体となるソフトウェアの理解を支援する機能が必要である．既存のソースコードからドキュメントを生成する機能，変更時の波及箇所や量の特定を支援する機能が必要である．また，特定の基準に従って仕様書やソースコードを書き換える適合化機能などが必要である．

システムソフトウェアの長時間，多人数による開発のためには，プロジェクトの状態を可視化するための機能が不可欠である．このために次のような機能が必要である．

（5） 工 程 管 理

工程管理はプロジェクトの進行にともない，作成された仕様書，プログラムの量を測り，計画との突き合わせを行って，問題を早期に検出することが目的である．設計，製造を支援するツールと連動し，進捗のデータが自動的に採取できることが望ましい．問題の発生時に解決方策を決定するための人や開発用の機器を再配分するための支援機能も必要である．

（6） 品 質 管 理

品質の尺度には信頼性，効率性，保全性，有用性，柔軟性などがあるが，システムソフトウェアにおいて最も重要なのは信頼性である．信頼性向上のためには，不良の検出を徹底することはもちろんであるが，何よりも不良を作り込まないことが重要である．このためには，製造工程のみならず，設計工程に対しても品質の尺度を定義し，工程ごとに評価を行うことが大切である．

（7） 構 成 管 理

システムソフトウェアの開発においては，設計変更が多数の担当者によって並行してなされる．このような仕様書やプログラムに対してなされる種々の変更に対応してその時点時点での内容を互いの関連を明確にして保存することが，設計，製造における整合性の保証には不可欠である．

3. CASE の現状と動向

3.1 上流工程の CASE

上流工程の CASE は，ソフトウェア開発過程で仕様の定義からシステムの構造設計を支援する．現在のところ，開発方針立案などを行う Upper Upper CASE，要求定義などの Upper CASE ともにシステムソフトウェア向きに特化して商用になっているものはない．汎用の CASE を流用するか，個々の会社で自家用に構築している．

システムソフトウェアを対象とした上流工程の CASE として以下があげられる．

（1） 設 計 支 援

大規模，多重性，実時間処理というシステムソフトウェアの特性から，構造化，リアルタイム処理，状態

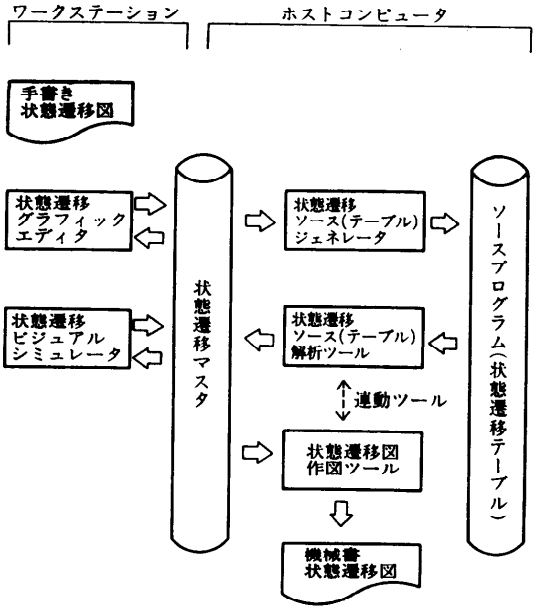

図-1 状態遷移設計支援 CASL 構成

遷移などの設計を支援する機能が必要である.

リアルタイム用の設計 CASE として, 構造化分析に基づく Hatley と Ward の SA 手法を用いたツール[1)~3)]がある. また, リアルタイムシステムの記述を設計段階まで持ち込んだ考え方に Darts のタスクデザイン[4)] がある.

状態遷移図に基づくものとしては, statechart[5)], CASL[6),7)] などがある. CASL のシステム構成を図-1 に示す. CASL は, 状態遷移図を作成するグラフィックエディタとビジュアルシミュレータをワークステーション上に実現している. 状態遷移図の作成・検証後に, ホストコンピュータにデータを送る. ホスト側では, 状態遷移図からソースプログラムを生成する自動コーディングツールや, 保守用ドキュメントを生成する状態遷移図自動出力ツールを使って, プログラムやドキュメントの作成を行う.

(2) 検 証 支 援

仕様をレビュー・検証するため, 図式記述法, シミュレーション及びプロトタイピング機能が必要である. 特に並行処理に対する検証が重要である.

並列プロセスやリアルタイムシステムの検証のためにシミュレーション, プロトタイピング[8)] を行うツールとして, 状態遷移図に基づく GIST[9)], 時間論理に基づく ENVISAGER[10)], ペトリネットに基づく PROT[11)], 関数型言語に基づく PAISLey[12),13)], 論理型言語に基づく P-Flots[14)] などがある.

また, 上述の CASL のシミュレータには,

- 状態遷移動作表示機能 (ウオークスルー)
- プロセス間通信制御機能
- データトレース機能

があり, 状態遷移図の動作検証を効率的に支援する.

(3) 再利用支援

上流工程の再利用としては, 設計に関する知識と仕様データの再利用が必要である.

知識を再利用するツールは, 一般にエキスパートシステムとしてとらえることができる. ソフトウェア作成に関する知識を利用するツール[15),16)], OS の性能改善に関する知識を利用するツール[17)] などがある.

生産物再利用ツールとして, 仕様書を再利用するもの[18)] や, 仕様から Ada の部品を取り込む PSDL[19),20)] がある. また並列プログラム部品を時制論理を用いて結合する方式[21)]も提案されている.

3.2 中・下流の CASE

(1) プログラミング支援

汎用大型 OS は数百万ステップからなる膨大なプログラムで構成されているが, その入力や修正を行うエディタは, プログラム開発時に最も頻繁に使用されるツールであり, エディタの使い勝手がプログラム開発効率にあたえる影響は大きい[22)].

エディタには, 行エディタ, 画面エディタのほかに, プログラミング言語の文法規則に従って構文要素単位のプログラム編集を行う構造エディタ[23)]がある. 構造エディタの例としては, ALOE[24)], KBEmacs[25)], PARSE[26)] などがある. また, ハードウェアの発達を背景に, PAD, NS チャート, HCP チャートなど構造化プログラミング[27)]に対応する図式[28)]で表現されたプログラムを編集する図式エディタも普及してきている[29)].

SEWB[30)]は統合型の CASE であり, 下流工程用に PAD エディタ, プログラム生成系, プログラム解析系, PAD テスタなどを備えている (図-2).

PAD エディタは, 連接・反復・分岐の三つのシンボルを用い, 概要から詳細へ段階的に記述する方式を採用している. また, PAD 図の文法を内蔵し, 画素の大きさ・結合・配置を自動的に決定し入力を誘導する文法誘導型編集方式を採用している.

プログラム生成系は, 設計仕様をCなどの特定言語で記述したソースプログラムに変換する機能をもつ.

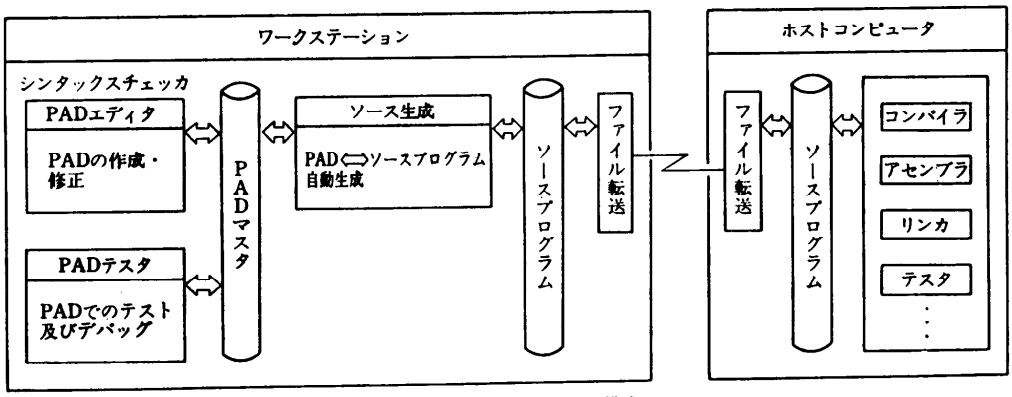

図-2　PAD 支援ツールの構成

プログラム解析系は，生成系と対をなすもので，ソースプログラムの構造を解析して設計仕様を作る逆変換系である．また，PAD テスタは PAD によるテストを支援するもので，実行したボックスの色を刻々と変えたり，変数の内容を実行に合わせて同時に表示する．

膨大な量の設計情報の入力を支援する方法として，専業入出力サービスを利用した運用も考えられている[31),32)]．

（2）　テスト支援

テスト工程はソースコードの解析，テストデータの作成，カバレージ分析，テスト環境の準備など，種々の作業を含む．それに対応してテスト支援ツールも，静的解析ツール，記号実行ツール，テストデータ生成ツール，カバレッジ分析ツール，ドライバ・スタブ生成ツール，シミュレータなど，さまざまなものがある[33)]．

OS を構成するソフトウェアは，ほとんどが特権モードで実行されるため，テストを実計算機または仮想計算機を独占して行う．このため，OS の開発システムとテストシステムが別のシステムとなり作業効率が悪いとか，OS のテストを行う実計算機または仮想計算機上では，高度なテスト支援機能が実現できないという問題があり，一般の応用プログラムのテスト環境とは異なるテスト環境が必要である．

OS 用のテスト支援ツールとしては，遠隔地から

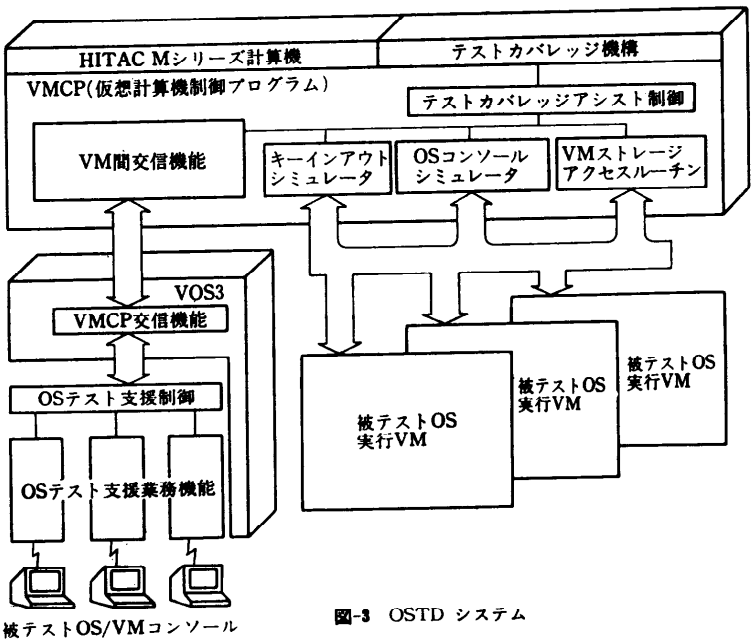

図-3　OSTD システム

OS のテスト・デバッグを行うことを可能にする WAVE システム[34)]，OS のテストの自動化を目的とした ART-IV システム[35)]，OSTD[36)]などがある．

OSTD は，特権プログラム比率が高い OS のテスト・デバッギングに必要な諸機能を体系的に提供するシステムである．OSTD は，図-3 に示すように，テスト制御用仮想計算機（VM：Virtual Machine）と被テスト用 VM 間の VM 間更新機能・被テスト OS コンソールと OS コンソールのシミュレーション機構などをもち，① OS のテストとデバッグを 1 台の端末で会話形式で実行可能にする，②仮想計算機能を用いて複数の OS を同時に実行させ，被テスト OS の監視，実行制御を可能にする，③ TSS 機能を用いて

ダンプ情報解析，端末操作情報，ストレージダンプ収集を可能にする，④テスト十分性評価を OS のテストに適応可能にするなど，種々の機能を備えている．これらの機能により，1台の端末だけを用いて OS のテスト・デバッグ作業が連続的に実行でき，合理的で無駄のない作業を可能にしている．

3.3 保守用の CASE（リエンジニアリング）

ソフトウェアのライフサイクルの中で，保守工数の占める割合は，一般に 50% 以上といわれている．システムソフトウェアの分野ではソフトウェアの寿命が長いため，この割合はさらに高い．このため，いわゆる 3R's (Reverse Engineering, Re-structuring, Re-engineering) の技術が重要である．

Reverse Engineering は既存のソフトウェアを分析して，構造や仕様の理解を支援する技術である．Re-structuring は悪い構造のソフトウェアを良構造化する技術である．Re-Engineering は既存ソフトウェアを新しい仕様に改造するために，既存ソフトウェアの理解と新ソフトウェアへの改造を支援する機能である[37]．

システムソフトウェアを対象とした 3R's のツールは現時点では十分ではない．Reverse Engineering に分類されるツールとして，ソースコードを解析してモジュールやテーブルの構造をドキュメント出力するツールが開発されている程度である[38]．

汎用ツールとしては，研究段階であるが，知識処理により，ソースコードの意図の理解を支援するツールの開発が行われている．その一つである Programmer's Apprentice[16] はソースコードをもとにプログラマの意図を理解して，コードの修正作業を関連箇所の自動変更により支援する．

3R's のツールの開発は事務処理ソフトウェアを対象とした分野で最も盛んである．構造化プログラムへの自動変換ツール[39]，データ名を統一するツール[40]，標準インタフェースへの自動変換ツール[41]などが開発されている．今後，システムソフトウェアの分野でもこの種のツールが開発されることが期待される．

3.4 工程間に横断的な CASE

計画，設計から保守にいたる開発工程全体に関わるものとして，CASE プラットフォームと管理用 CASE がある．

（1） CASE プラットフォーム

これは，種々のツールを統合化するための土台となるものである．ツールの対話部分を標準化することによりユーザインタフェースの統一を行い，仕様書やプログラムを一元的に管理してツール間の受け渡しを円滑にする．

ESPRIT（欧州戦略情報技術研究開発計画：European Strategic Program for Research and development in Information Technology）の PCTE[42] (Portable Common Tool Environment) プロジェクトはプラットフォーム用の標準インタフェース規格を定めている．Emeraude[43] はこの規格に準じて開発されたプラットフォームの例である．PCTE プラットフォームは，①仕様書やプログラムを管理する OMS (Object Management System) 機能，②個々のツールの実行，複数のツールの合成，ツール間のコミュニケーションを管理する機能，③分散化環境の管理，④ユーザインタフェースの統一機能などをもつ．

プラットフォームのインタフェースの標準化の他の例としては Ada プログラミング環境 APSE におけるインタフェース CAIS (Common APSE Interface Set) がある．

リポジトリ（Repository：情報資源倉庫）[44]は，プラットフォームの一部であり，ソフトウェア開発の過程で生み出された種々の情報を蓄積して再利用するための仕掛けである．蓄積する情報としては，仕様書，プログラム，データディクショナリなどの開発情報のほかに，ビジネスモデルや企業戦略などの計画情報，進捗状況，品質評価データなどの管理情報，さらには開発のための設計規格，ノウハウなどの技術情報などがある．リポジトリは，単にこれらの情報を格納するだけでなく，情報の間の関係，たとえば参照関係，生成関係などを含めて蓄積することができるため，ツールの統合化において大きな役割を果たすものとして期待されている．上述の PCTE の OMS はリポジトリの一種である．

（2） 管理 CASE

システムソフトウェアの開発においては品質，特に信頼性の管理が重要となる．

SQE[45]，SQMAT[46]，ESQUT[47] などの品質管理ツールは信頼性，操作性，ポータビリティなどの項目について開発の各段階で採取したデータから評価値を計算し，その結果をグラフや表で表示することにより，品質の可視化を支援する．

SQE は，システムソフトウェアの信頼性の管理を目的として開発された．設計品質，テスト品質，総合品質の三つの評価を行うサブシステムから構成されて

おり，開発の全工程をカバーしている（図-4）．設計品質評価部は，開発規模，処理内容，開発形態，単位規模あたりの設計工数などから品質を推定する機能をもち，信頼性設計の際の目標値設定に利用される．テスト品質評価部は設計時に設定した目標値と摘出実績を比較してグラフ出力する．また，不良実績の累積値の推移を成長曲線に当てはめて残された不良の件数を推定する機能ももつ．総合評価部は，検査段階で使用するもので，設計品質評価部とテスト品質評価部から出力されたデータをもとに総合評価指数を計算し，レーダチャート形式で出力する．工程管理ツールの例としては CAPS[48]，IKKS[49] などがある．これらは工程ごとの工数標準値の推定によるスケジューリング支援や実績データの採取，予・実績データの比較表示などを行う．

システムソフトウェアの開発では，このほかに，構成管理，原価管理などの機能も必要となる．

4. おわりに

CASE は薬である．何にでも効く万能薬は便利ではあるが効き目は遅い．重体の患者には，体質や症状にあった特効薬が必要である．特効薬は，適用範囲は狭く，副作用の危険性はあるが，正しく使えば，大きな効果を発揮する．CASE がより有効であるには，対象ソフトウェアの性質だけでなく，開発者の特性，組織のポテンシャルなどに合わせて特化したものであり，さらに正しく使われる必要がある．

システムソフトウェアの開発を対象とした CASE の場合，現在でも設計用，テスト用，管理用のツールの一部に特化がみられるが，多くの場合，より汎用的な技術やツールの流用となっている．もちろん，汎用のツールも必要であるが，特定の効果を狙ったツールも必要である．

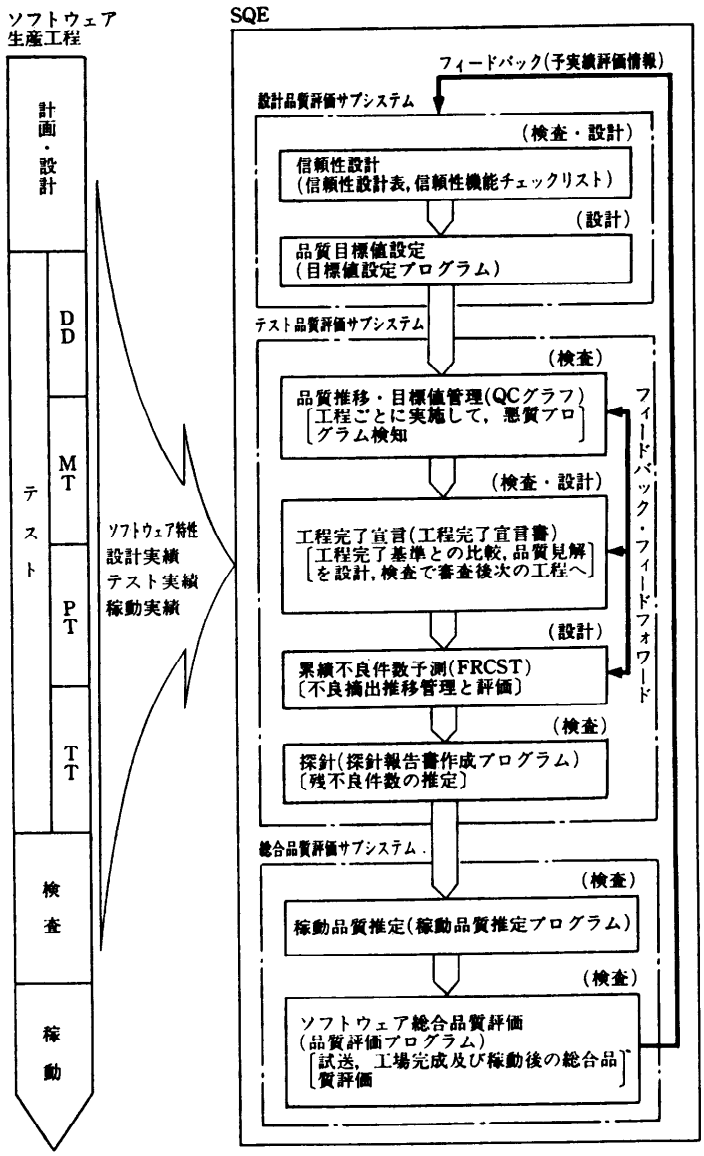

図-4　SQE の構成

たとえば，システムソフトウェアの性質をとらえた再利用およびエンジニアリングのツールは一層の研究開発が必要である．また，管理機能と設計，製造機能の連携の強化も必要である．多人数が同時に働くオフィス環境として，CASE と OA との融合化も必要となる．これらの機能が充実して，はじめて CASE の定着化が加速される．

参 考 文 献

1) Ward, P. T.: The Transformation Schema: An Extension of the Data Flow Diagram to Represent Control and Timing, IEEE Trans., Vol. SE-12, No. 2, pp. 198-210 (1986).
2) Forte, G.: リアルタイム CASE ツール，組み込みシステムをねらい機能強化，日経エレクトロニクス 5-29, No. 474 (1989).
3) Hatley, D. J. and Pirbhai, I. A.: Strategies for Real-Time System Specification, Dorset House Publishing (1987).
4) Gomma, H.: Software Development of Real-Time Systems, Communications of ACM, Vol. 29, No. 7, pp. 657-668 (1986).
5) Harel, D.: On Visual Formalism, Communication of the ACM, Vol. 31, No. 5, pp. 514-530 (1988).
6) 金子他：状態遷移ビジュアルシミュレータの検討，信学技法，SSE 89-28 (1989).
7) 中谷他：状態遷移シミュレータ，信学技法，SSE 88-160 (1988).
8) 伊藤，本位田：プロトタイピング支援ツール，情報処理，Vol. 30, No. 4, pp. 387-395 (1989).
9) Balzer, R. M. et al.: Operational Specification as Basis for Rapid Prototyping, ACM, SIGSOFT, SE Notes, Vol. 7, No. 5, pp. 3-16 (1982).
10) Diaz-Gonzales, J. P. et al.: Prototyping Conceptual Models of Real-Time System: A Visual Perspective, HICSS '89, pp. 358-367 (1989).
11) Bruno, G. and Marchetto, G.: Rapid Prototyping of Control System Using with High Level Petri Nets, 8th ICSE, pp. 230-235 (1985).
12) Zave, P.: An Overview of the PAISLey Project, ACM SE Note, Vol. 9, No. 4, pp. 12-19 (1984).
13) Zave, P. and Schell, W.: Salient Features of an Executable Specification Language and Its Environment, IEEE Trans., Vol. SE-12, No. 2, pp. 312-325 (1986).
14) 田村他：並行処理ソフトウェアシステムの設計向きプロトタイピング手法とそのツール，情報処理学会論文誌，Vol. 28, No. 9, pp. 923-935 (1987).
15) Neighbors, J. M.: The Draco Approach to Constructing Software from Reusable Components, IEEE, Trans. Vol. SF-10, No. 5, pp. 564-574 (1984).
16) Waters, R. C.: The Programmer's Apprentice; A Session with KBEmacs, IEEE Trans., Vol. SF-11, No. 11, pp. 1296-1320 (1985).
17) 麻生川他：ACOS-4 性能改善エキスパートシステムの開発，情報処理学会研究会報告，OS 35-2 (1987).
18) 千吉良他：システム仕様書の再利用によるソフトウェア開発技法，日立評論，Vol. 69, No. 3, pp. 47-52 (1987).
19) Luqi and Berzins, V.: Rapidly Prototyping Real Time System, IEEE Software, pp. 25-36 (1988).
20) Luqi and Berzins, V.: A Prototyping Language for Real-Time Software, IEEE Trans., Vol. SF-14, No. 10, pp. 1409-1423 (1988).
21) 内平，本位田：時制命題論理を用いた部品からの並列プログラム合成，日本ソフトウェア科学会第3回大会論文集，pp. 145-148 (1986).
22) 野木他：プログラミングツール，昭晃堂(1989).
23) 原田賢一：構造エディタ，共立出版 (1987).
24) Medina-Mora, R. et al.: An Incremental Programming Environment, IEEE Trans., Vol. SE-7, No. 5, pp. 472-482 (1981).
25) Waters, R. C.: The Programmer's Apprentice: A Session with KBEmacs, IEEE Trans., Vol. SE-11, No. 11, pp. 1296-1320 (1985).
26) 田中他：計算機誘導型構造エディタ，日立評論，Vol. 68, No. 5, pp. 393-398 (1986).
27) Dijkstra, E. W.: Notes on Structured Programming, Academic Press (1972).
28) 二村良彦：構造化プログラム図式，コンピュータソフトウェア，Vol. 1, No. 1, pp. 64-77 (1984),
29) Visual Programming, IEEE Computer, Vol. 18, No. 8 (1985).
30) 葉木他："SEWB" の開発思想と機能，日立評論，Vol. 70, No. 2, pp. 101-108 (1988).
31) 梶原他：PAD に基づく処理手順設計—ツールの概要とその運用—，情報処理学会第 35 回全国大会講演論文集，pp. 1167-1168 (1987).
32) 鈴木他：ソフトウェア構造設計支援システム "ADDS"，日立評論，Vol. 66, No. 3, pp. 185-188 (1984).
33) 玉井他：ソフトウェアのテスト技法，共立出版 (1988).
34) 佐藤他：VM 機能を持つ TSS システムの実現法，情報処理学会計算機システムの制御と評価研究会資料 18-5 (1983).
35) 隈本他：自動テストの一手法，情報処理学会第 26 回全国大会講演論文集，pp. 587-588 (1983).
36) 久保他：汎用 OS テスト支援システム OSTD，情報処理学会第 33 回全国大会講演論文集，pp. 365-372 (1986).
37) Chikofsky, E. J. et al.: Reverse Engineering and Design Recovery: A Taxonomy, IEEE Software, pp. 13-17 (1990).
38) 開発進む「Σ ツール」ソフト開発分散環境が目前に，日経コンピュータ，pp. 115-124 (1987. 5. 11).
39) プログラムの保守・開発費用を節減する再構造化ツール，日経コンピュータ，pp. 81-90 (1986.

4. 11).
40) Tools to Rejuvavate Your Old System, EDP Analyzer, Vol. 22, No. 4, pp. 1-14 (1984).
41) 内藤他：事務処理ソフトウェア標準化支援システムの開発，情報処理学会第 39 回全国大会講演論文集，pp. 1607-1608 (1989).
42) PCTE Interfaces: Supporting Tools in Software Engineering Environment, IEEE Software, Vol. 6, No. 6, pp. 15-23 (1989).
43) Campbell, I.: Emeraude Portable Common Tool Environment, Information and Software Technology, Vol. 30, No. 4 (1988).
44) Maclure, C.: CASE is Software Automation, Prentice Hall (1989).
45) 橋本他：ソフトウェア品質評価システム "SQE" 日立評論，Vol. 68, No. 5, pp. 55-58 (1986).
46) 山崎他：SQMAT 支援ツールの開発，情報処理学会第 37 回全国大会講演論文集，pp. 750-751 (1988).
47) 平山他：モジュール設計段階における品質評価の一手法，情報処理学会第 36 回全国大会講演論文集 pp. 941-942 (1988).
48) 芝田他：総合ソフトウェア生産管理システム "CAPS"，日立評論，Vol. 62, No. 12, pp. 37-42 (1980).
49) 寺門他：ソフトウェア工程管理システム IKKS，情報処理学会第 39 回全国大会講演論文集，pp. 1409-1414 (1989).

(平成 2 年 5 月 28 日受付)