解　説

ＣＡＳＥの歴史と展望

藤尾　好則

１．背　　景

近年，コンピュータのハードウェア／ソフトウェア技術やそれらを結びつけるネットワーク技術の発展はめざましく，その利用は企業活動から家庭に至るまで，広い範囲にわたっている。利用範囲の広がりに伴って，ソフトウェア開発の要求は多様化，複雑化，高度化してきている。しかし，ハードウェアの性能向上に比べて，ソフトウェア開発における生産性の向上は低い。

その原因は，人力のみで生産性を上げるには限界にきていること，日々のシステムを維持・管理するための保守費用に多くの工数を費やしていることである。

また，開発作業の成果物であるソフトウェアの品質が悪いことである。このため，品質を作り込むための仕掛が必要であり，ＣＡＳＥツールはその仕掛の１つといえる。

２．定　　義

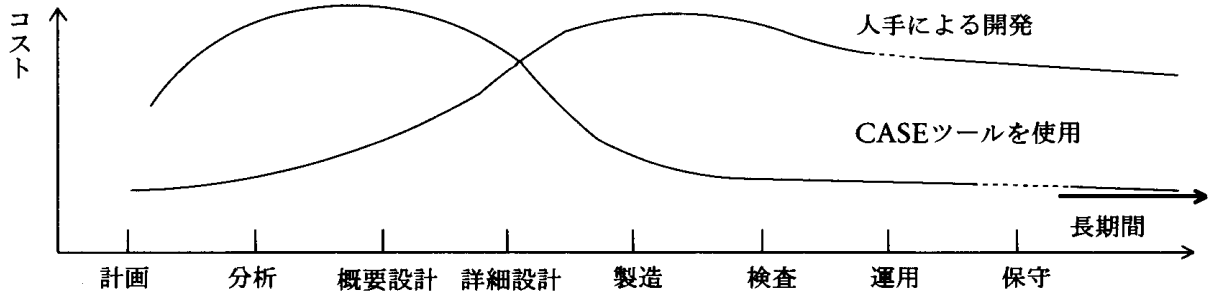
ＣＡＳＥ (Computer Aided Software Engineering) の目標は，「ソフトウェア開発の生産性向上，ソフトウェア製品の品質向上」である。具体的には，「統合化されたツールの集合によるソフトウェア・ライフサイクル全体の自動化」を目指している。ＣＡＳＥツールとはソフトウェア開発，保守の仕事を自動化する手助けとなるようなソフトウェア・ツールである。

ＣＡＳＥは，システムの開発工程に沿った開発手順や標準をまとめた『開発方法論』，ＣＡＳＥをコンピュータ上で実現する『ＣＡＳＥツール』とＣＡＳＥを導入する場合の『適用技術（適用のノウハウ）』を巧く組み合わせて，はじめて有効に活用できる。

ＣＡＳＥツールの効果を開発費用の点からとらえると図のようになる。

ＣＡＳＥツールを適用した開発は，人手による開発に比べて計画，分析，概要設計など上流工程では多くの開発工数を必要とするが，下流工程では製造，検査などが自動化されているため工数は少なくてよい。

初めてＣＡＳＥツールを適用したときは方法論，ツール操作の学習などに時間を費やすが，繰り返しＣＡＳＥツールをシステム開発に適用すれば，習熟し，適用技術のノウハウが蓄積されて上流工程の工数も減少してくると考えられる。



ライフサイクル

さらに，ソフトウェア工学／情報工学を推進し，構造化技法を実用化する，機能検証によりソフトウェアの品質が向上する，ソフトウェアの保守を支援する，開発期間が短縮される，再利用を促進するなどの効果が挙げられる。

３．ソフトウェア開発のプロセス

ソフトウェア開発プロセスはソフトウェア製品を作るために用いられる，ツール，技法と慣習をセットにしたものである。

カーネギメロン大学のＷ．Ｓ．Humphray 教授は，次に示す５段階のプロセスの発展レベルと６つの改善ステップを提案している。ＣＡＳＥの適用は『③定義されたレベル』から可能であると考えられる。従って，組織としては現在どのレベルにあるかを認識して，常に６つの改善ステップを実施して発展レベルの段階を上げる努力が必要である。

【５段階の発展レベル】

①初期レベル＝無秩序が常な組織。プロジェクトの成功失敗はプロジェクトの性格や作業者の腕前に依存している。

②反復可能なレベル＝全てのプロジェクトが安定して管理されている。プロジェクト管理者の腕前に依存している。

③定義されたレベル＝作業プロセスの一貫した定義が組織として行われ，その作業プロセスを一貫して全てのプロジェクトで実現している。

④管理されたレベル＝各作業段階のプロセスを全て計測評価し，公式な品質保証グループが責任を持っている。

⑤最適化されたレベル＝プロセス自体の改善を行うための積み重ねと改善が組織体に組み込まれている。

【６つの改善ステップ】

①ステップ１＝現在の開発プロセスの状況を認識する

②ステップ２＝望ましいプロセス像を作る

③ステップ３＝プロセス改善に必要な活動の優先度を確立する

④ステップ４＝必要な活動を遂行できるように計画を作る

⑤ステップ５＝資源を確保して計画を実行する

⑥ステップ６＝開発ステップ１からはじめる

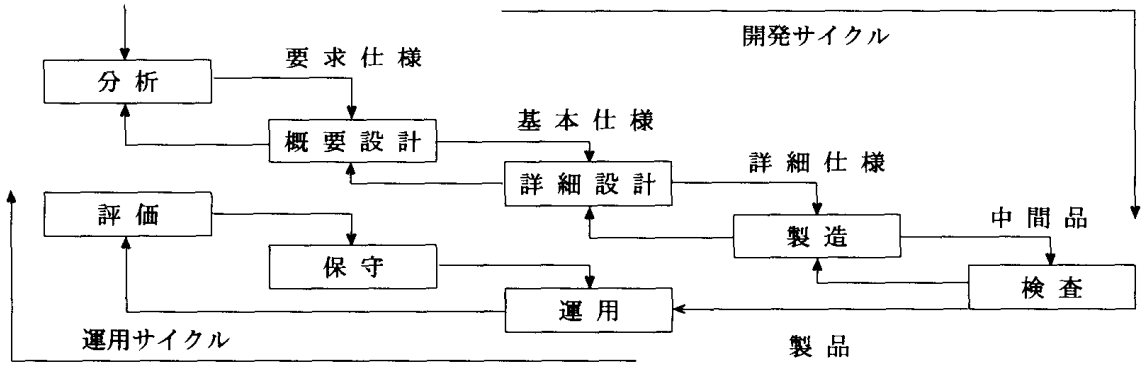
ソフトウェアを工業的に生産するための開発プロセスは次の３種類のモデルがある。

(1) ウォータフォールモデル

Royce によって発表され，1970 年代から '80 年にかけて，ほとんどのソフトウェアはこのモデルに沿って開発されてきた。日本では日本電気のＳＴＥＰＳ，日立のＨＩＰＡＣＥ，富士通のＳＤＥＭのベースになっている。

ソフトウェアの開発から保守までのライフサイクルを図に示すように，いくつかの工程に分割する。各工程では定められた形式で仕様やプログラムを作成して，その正しさを検証してその工程を終了する。

ＣＡＳＥツールの狙いの一つは，この一連の仕様の作成をコンピュータで支援することである。

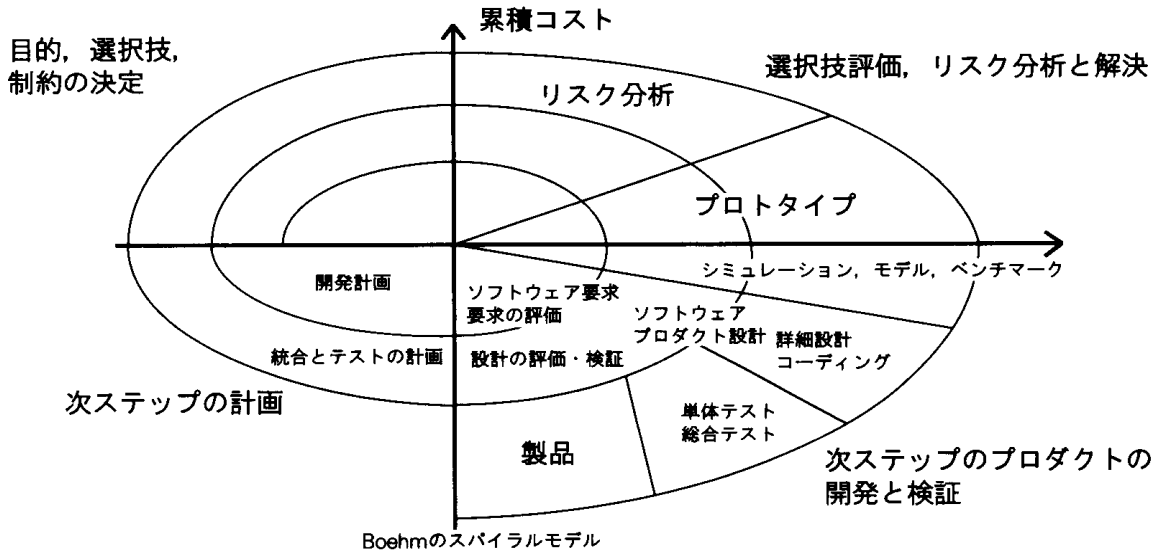


(2) スパイラルモデル

スパイラルモデルは Boehm が提唱したものである。

ウォータフォールモデルでは上流工程でユーザの要求を十分取り込めず仕様の変

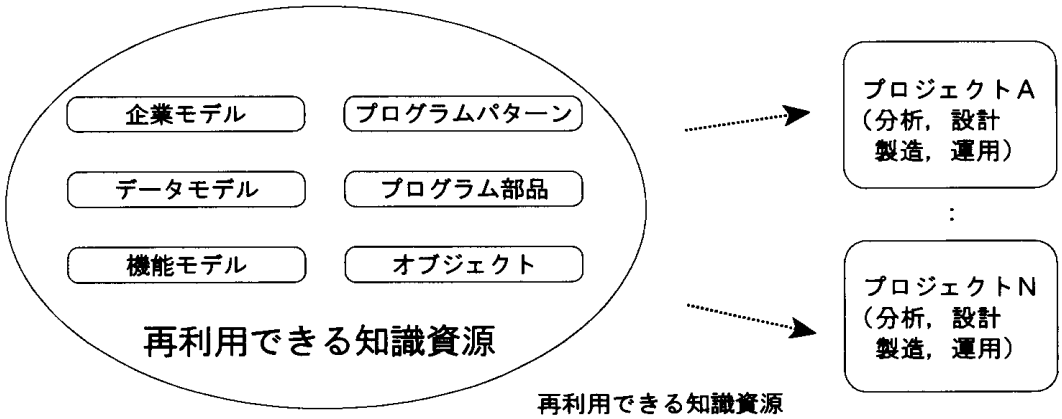
更が発生すると，下流工程でプログラムの修正が発生し，多くの費用が必要になる。この欠点を補うためプロトタイプを作り，これをシミュレーションしたり，仮に実行して，ユーザとシステムの要求を詳細化／洗練して本番用の実規模のシステムに作り上げて行く。すなわち，「リスク分析」，「プロトタイプ作成」，「評価・検証」，「計画」の繰り返しにより，プロトタイプを実システムに成長させて行くものである。



⑶　再利用モデル

1980 年代に，ソフトウェアの生産性を確実に向上させる方法として，プログラムの再利用が叫ばれた。生産性が最も向上するのは「作らないこと」である。この考えをソフトウェア資源の再利用だけでなく，知識資源にまで拡張したのが再利用モデルである。

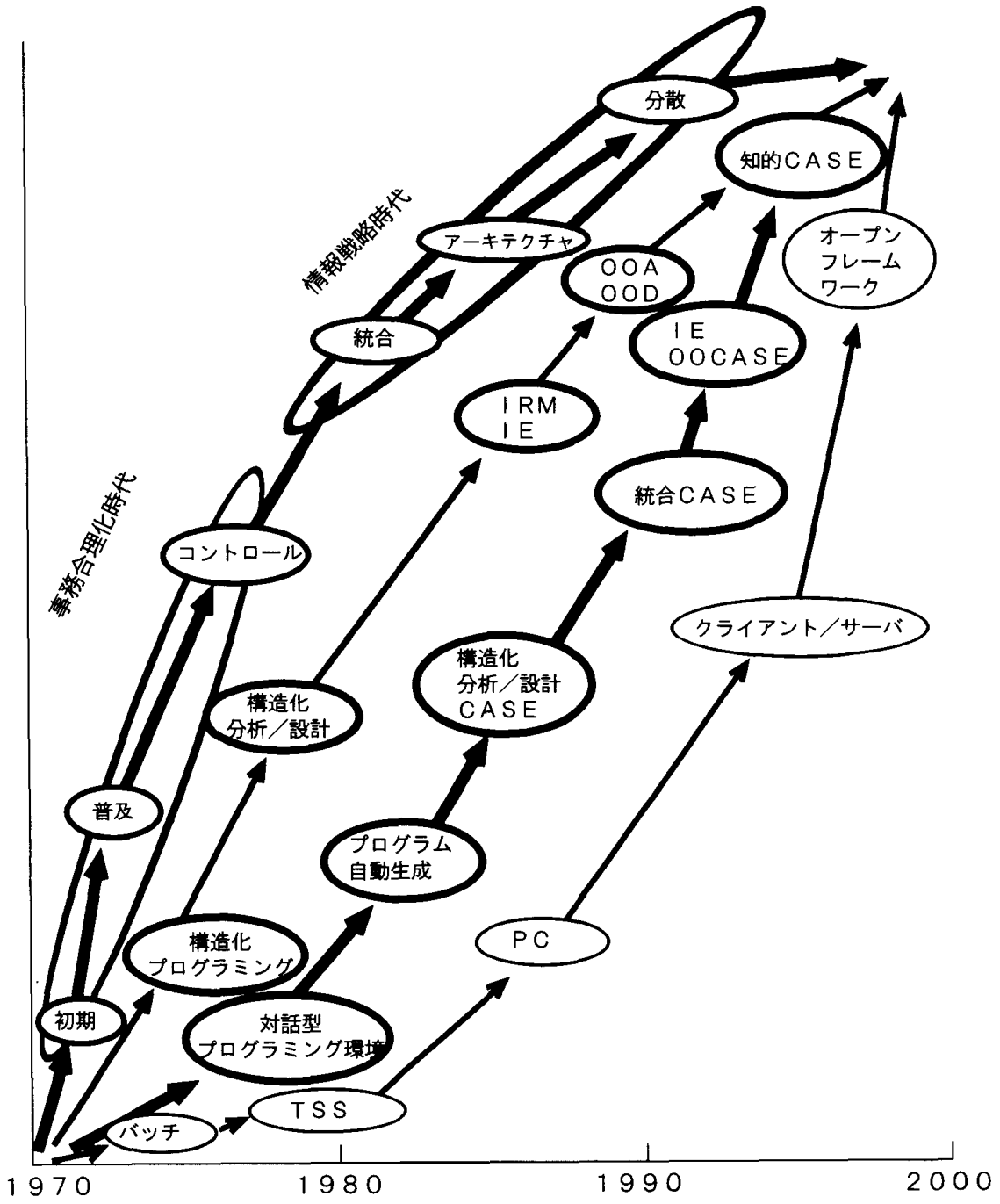
再利用できるものとしては「企業モデル」，「データモデル」，「機能モデル」，「プログラム部品／パターン」，「オブジェクト」などがある。これらの知識を機械で管理し，必要な都度参照，引用して設計し，プログラムを作成する。このため再利用するユニットとこれらを取り込む仕組みが重要である。



４．変遷と動向

ＣＡＳＥを取り巻く環境，特に，情報システムのシステム化の狙い，開発技法やＣＡＳＥツール，ツールの動作環境に関する変遷がＣＡＳＥに影響を及ぼしている。

これまでの経緯と今後の動向の流れを次の図に示す。



４．１　ＣＡＳＥの基盤となる方法論，技法の発展

ＣＡＳＥは方法論や技法をもとに構築されており，ここではＣＡＳＥの発展 の中で影響を与えている主な方法論と技法を順に記述する。

⑴　構造化技法

構造化技法法は，1968 年に Dijkstra がＧＯＴＯ文の有害性を説いて，プログラムを「連接」「選択」「繰り返し」の３つの組み合わせで構成されることを提唱したのが始まりである。その後，約 20 年間発展を続け，1980 年代の中頃には成熟期に達したとみられる。この技法は多くのＣＡＳＥで使用されている。

①　構造化プログラミング (SP: Structured Programming)

論理的なユニットを「順次」「選択」「繰り返し」の３種類の構造の組み合わせとしてプログラムを作成する。

構造化されたプログラムはＧＯＴＯ文が排除されるため見やすいプログラムとなり，保守しやすいものとなる。また，論理的に検証しやすいため，バグが少なく品質が向上する。

②　構造化分析技法 (SA: Structured Analysis technology)

システムの要求を記述する技法で，データフローに着目してユーザ要求を分析し，データフロー図 (DFD: Data Flow Diagram) の階層構造として表現する技法である。E. Yourdon と T. DeMarco らによって集大成され，1970 年代終わりから 1980 年代にかけて米国を中心に普及した。

構造化分析技法は主機能とその入出力データの流れを把握し，主機能の下位レベルの機能と入出力データの流れを調べることを繰り返してシステムを分解し，設計が開始される前にシステムの要求を定義する。この成果は複数のデータフロー図，複数のミニスペック，及びデータ辞書から構成される。

特徴は，何を行うか (What) を表現し，どのように処理するか (How) は表現しない。視覚的にわかりやすい図形表現を用い，トップダウンに階層化した仕様書を作成するなどが挙げられる。

③　構造化設計技法 (SD: Structured Design technology)

主な概念は，L. Constantine によって確立され，G. J. Myers，W. P. Stevens が論文を発表して有名になった。

構造化プログラミング手法をシステム設計に持ち込んだもので，ソフトウェア

システムのインプリメンテーションから独立した部分のモジュール，副モジュールや手続きをトップダウンで設計するソフトウェアの設計法である。成果物は構造図，モジュール仕様，データ辞書の３つで構成される。

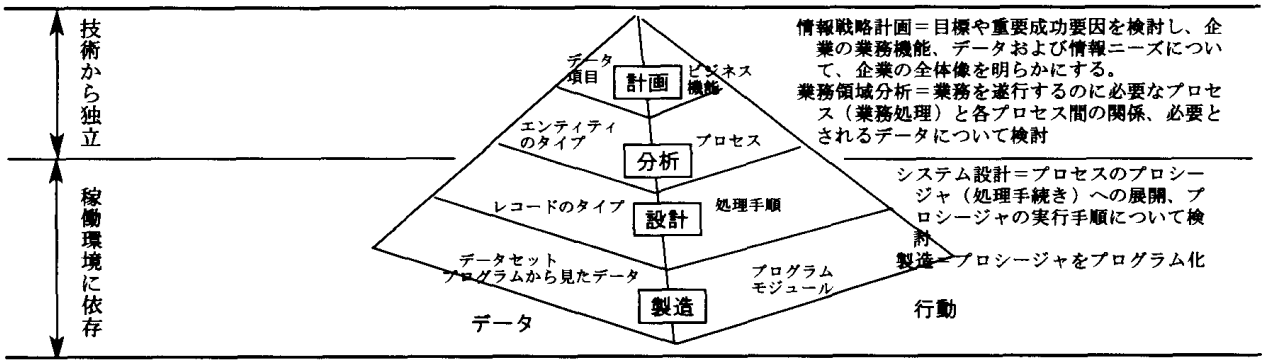
この手法には，モジュールがどのくらいブラックボックスとして情報隠ぺいされているかといった「モジュール化の度合い」を最大にしたり，モジュール間の結合度や情報の流れを最小にする手順が示されている。

⑵ インフォメーション・エンジニアリング (IE=Information Engineering)

James Martin が提唱している開発技法で，「企業全体または企業の複数部門を対象にした情報システム化を実現するために情報戦略計画，業務領域分析，システム設計，製造の４段階を構造化技法／エンティティ関連などの技法を体系的に組み合わせてトップダウンに進める技法」である。

ＩＥの特長として，経営学でいう戦略経営理論とシステム分析を行う際の構造化分析／設計を統合させた点にある。すなわち企業の目標や目標達成のため重要要因 (CSF = Critical Success Factor) は何か，など計画段階で分析する。さらに企業活動をデータモデル，プロセスモデルとして構築，このモデルの枠組みの中で分析，設計を行い，最終的なソフトウェアの論理設計へと作業を進めていく。

データ重視の考え方が強く，自動化ツールの利用を前提にシステムの開発・保守の手順が考えられている。



J. Martin のピラミッド

⑶ オブジェクト指向技法

我々の周囲の世界をさまざまなオブジェクト（対象物，もの）の集まりとしてとらえ，その振る舞いに着目してモデル化する。すなわち，日頃の慣れた見方で，現実の世界を眺めて，いくつかのオブジェクトとその間のメッセージの伝達にまとめることである。

基本要素はオブジェクトであり，単一の実体の中にデータ構造とその振る舞いの

両方を持つこと（カプセル化）を特徴とする。内部の詳細情報はできるだけ隠し，外部には機能特性（インタフェース）のみを見せる。

さらに，新しいクラス（導出クラス）での差分以外は既存のクラス（基本クラス）と同じとすることができる継承の概念。

「円」と「線」の2つのオブジェクトに「強調」という操作があるとき，「強調」というメッセージを「円」オブジェクトに送れば円を黒色にし，「線」オブジェクトに送れば線を太くする。このように異なるオブジェクトが同一メッセージに異なる反応をする多相性。

この「もの」に着目した方法は，クラスの再利用性が可能であるばかりでなくユーザと設計者との間で問題の把握がしやすいという長所がある。

① オブジェクト指向プログラミング

オブジェクト指向言語は，Algol から Simula を経てオブジェクト指向言語として開発された Smalltalk-80（Xerox 社）や C 言語にオブジェクト指向機能を付加した Objective-C（Stepstone 社），C++(AT & T 社）がある。また，Lips 上のオブジェクト指向言語としては Loops（Xerox 社），Flavors（Symbolics 社，Lisp Machine 社）などがある。

この言語はプログラムの簡易性，保守性の向上を目指している。

② オブジェクト指向分析／設計

P．Coad と E．Yourdon によるオブジェクト指向分析手法であるコード／ヨードン法は，実世界の「もの」から，「オブジェクト」を切り出し，「もの」のもつ機能や役割からオブジェクトが行うサービスを定義し，オブジェクトというカプセルを定義し，継承を考慮したクラス階層の定義を行う。

同じく Shlaer と Mellor によるシュレア／メラー法は，従来の構造分析，意味データモデルといった手法をオブジェクト分析に適用し，情報モデル，状態モデル，プロセスモデルの3つのモデルで表現する。

また Booch のオブジェクト指向設計では，ソフトウェアの複雑さは対象システムの階層構造として現れると考え，この複雑さに対処するため，分割，抽象化，階層化の手法を用いる。この手法を使い，システム要求仕様を満たすような形で少しずつ設計モデルを作っていく。

4．2　ＣＡＳＥツールの発展

⑴ 部品化，パターン化に基づくプログラミング開発環境

データ辞書とプログラム部品を再利用する支援ツールを中心とした汎用コンピュータ向けのソフトウェア開発支援環境。

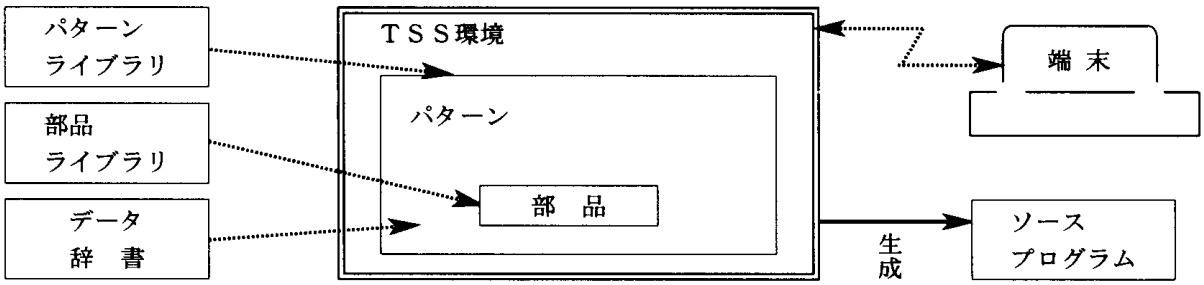
このソフトウェア開発支援環境はコンピュータのＴＳＳ環境で動作し，統一された操作性のあるメニュー画面から呼び出して使う。プログラム開発効率化に成果を上げた。

［代表製品］

ＮＥＣ＝ＳＯＦＰＩＡ（ＤＤＡ，ＰＳＡ，ＰＭＡ）

日　立＝ＥＡＧＬＥ２　　ＩＢＭ＝ＣＳＰ

富士通＝ＡＤＡＭＳ　　住商コンピュータサービス＝Ａｌｉｃｅ



ＣＡＳＥツール

⑵ 構造化分析／設計技法に基づく上流ＣＡＳＥ

ワークステーションは高速のＣＰＵと大容量のメモリを搭載し，要求分析や構造設計など，これまで手書きで済まされていた仕様やダイヤグラムの作成を支援する上流ＣＡＳＥツールである。米国を中心に登場した。

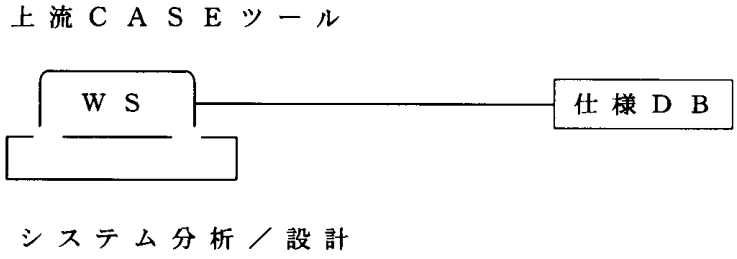
このツールで用いられるダイヤグラムには，構造図（システムやプログラムの構成を表す図），プロセス／データ関連図（データの作成と使用側の依存関係を表す），データフロー図（プロセス間をデータが流れ加工されていく様子を表す），データ構造図（データの順次，繰り返し，選択構造を表す），データモデル（実世界における実体間の関係をデータベースに反映させるモデル），アクション図（プログラムの概要や詳細な制御構造を表す，構造化されたフローチャート）などがある。

［代表製品］

Cadre Technologies 社＝ Teamwork

Interactive Development Environment 社＝ Software through Picture

Index Technology 社＝ EXcelerator



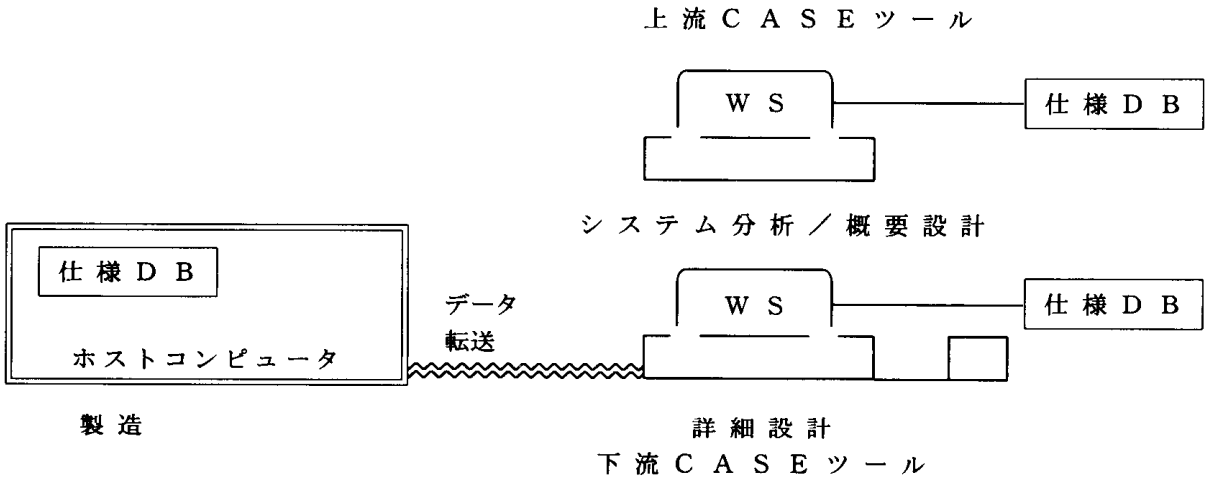
⑶ 分散開発環境での構造化プログラミング技法に基づくＣＡＳＥ

ＴＳＳの開発環境をワークステーションへ移し，設計支援系はワークステーション，生成系はホストコンピュータという分散開発環境として整備したＣＡＳＥである。

［代表製品］

ＮＥＣ＝ＩＤＬＴＯＯＬ，ＰＷＢ　　富士通＝ＹＰＳ／ＡＧＰ

日　立＝ＳＥＷＢ２



⑷ 統合型ＣＡＳＥ

上流工程から，下流工程までソフトウェアの開発ライフサイクル全体を一貫して支援するＣＡＳＥ (Integrated-CASE) である。

統合型ＣＡＳＥの特徴は，特定の開発方法論に基づき，この方法論の一連の作業である情報・機能モデルの設計からプログラム作成までの一連の作業を支援し，自動化している点である。次の４つの点で「統合」されている。

①開発ライフサイクルの各工程を支援するツールを用意して，全工程を一貫支援する。

②リポジトリは各工程で発生する情報を管理する情報資源倉庫である。リポジトリにより支援ツール間のデータ管理の統合や，ソフトウェア資産の再利用ができる。

③マンマシンインタフェースを統一して，ＣＡＳＥプラットホームにより操作性を統一する。

ワークステーションでは標準化が推進されておりＵＮＩＸ系はＯＳＦのＭｏｔｉｆが多い。

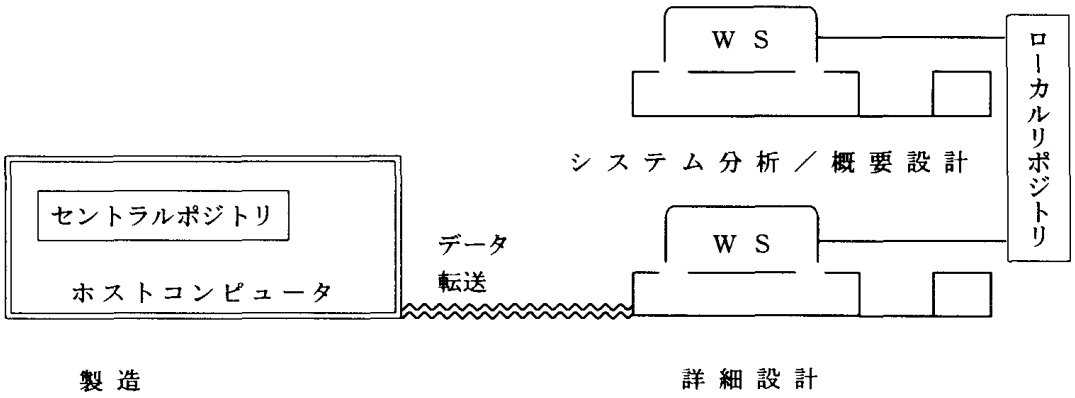
④ホストとワークステーションを連携し，ホストにある既存ソフトウェア資産を直接利用してワークステーションで分散開発ができる。

［代表製品］

ＮＥＣ＝ＣＡＳＥＷＯＲＬＤ　　Texas Instruments ＝ＩＥＦ

日　立＝ＳＥＷＢ３　　Knowledge Ware ＝ＩＥＷ，ＡＤＷ

富士通＝ＳＤＡＳ統合ＣＡＳＥ　Andersen Consulting ＝ FOUNDATION



⑤　知的ＣＡＳＥ

ソフトウェア開発には，これまで人手で行われているユーザから要求を正確に引き出す要求獲得や，要求された機能をこれまでの経験より具体的なプログラムに分割したり，パターン／部品を組み合わせてプログラムを設計する創造的な作業がある。このような創造的で経験や試行錯誤を必要とする知的作業を，蓄えられた知識データベースや推論機能によって自動化する。

さらに音声による自然言語で要求を入力したり，分析結果をアニメーションで表示できるような知的な機能をもったＣＡＳＥである。

5．課　題

ＣＡＳＥの変遷と動向で述べてきたように，現在は基幹情報システムを構築するためのＣＡＳＥとして，統合型ＣＡＳＥが位置づけられ活用されている。また知的ＣＡＳＥは今後発展が期待されるツールである。さらに，小規模，非定型処理を行う部門情報システムやエンドユーザが情報システムを構築するためのビジュアルかつパソコン上で動作する手軽なＣＡＳＥが出現してきた。

ＣＡＳＥの普及に伴い，下流工程は自動化されるためプログラマは高度なプログラム技術を必要とする分野（ＯＳ，ネットワークなど）の専門家になるか，さらに上流工程のシステム分析者，設計者への転向が必要になってきている。

しかし，上流工程におけるシステム計画，分析，設計はコンピュータのハードウェア／ソフトウェアの知識に加えて，情報システム構築の方法論とＣＡＳＥツールに精通していることが必須である。一方，対象システムの問題点の解決には行政／経営的なアプローチすなわち，業務知識に精通していること，かつ戦略的思考法などを駆使した計画立案やシステム概念の構築が必要である。

今後，工学的及び行政／経営的観点の両面からのアプローチに基づく，ＣＡＳＥを活用した情報システム構築法の研究と，この研究成果を大学の教育へ適用して行くことが課題であろう。

参考文献

⑴　原田実監修，「ＣＡＳＥのすべて」，オーム社，1991
⑵　J. Martin, “Information Engnieering, Book I”, Prentice-Hall International Editions, 1989
⑶　Boehm, B. W., “Software Engineering Economics”, Prentice-Hall, 1981
⑷　W. S. Humphrey 著，藤野喜一監訳，「ソフトウェアプロセス成熟度の改善」，日科技連，1991