解　説

1. CASE 環境の概要†

藤野　喜一††

1. はじめに

最近の情報化社会の発展にともない，ソフトウェアの開発量が大幅に増加してきており，その生産性を向上させることが急務になっている．また，開発すべきシステムの多様化，大規模化などの進行によって，ますます複雑になり，かつ高信頼性が要求されている．これらの高度に複雑なソフトウェアの開発には，もはや人海戦術では限界があり，ソフトウェア生産技術の改革が必要になりつつある．

これまで，構造化プログラミングにはじまり，データ抽象化，ジャクソン法，オブジェクト指向などの方法論や，これら方法論を支援するツールの研究開発が進められてきた．さらに，ソフトウェア生産の管理技術やツールについても，多くの研究がなされてきた．最近では，ソフトウェア開発の手順をモデル化し利用しようとするプロセスモデルの研究もさかんである．これらの研究成果のいくつかは，実際のソフトウェア開発に適用されているが，まだ多くの成果をあげているとはいえないし十分普及しているともいえないのが実状である．

一方，最近の C & C（コンピュータと通信）の発展には目を見張るものがあり，高機能高性能ワークステーションが低価格で入手できるようになってきた．コンピュータネットワークも急速に発展，整備され世界規模のネットワークが運用，活用されてきている．

このようなソフトウェア改革に対する「ニーズ」と，そのためのソフト／ハードの両面にわたる「シーズ」とが合致した「CASE 環境」への大きな期待がある．CASE 環境とは，ソフトウェア生産（開発，管理，保守，普及など）へのコンピュータの応用である CASE に関わる環境すべてを指す．

CASE 環境の歴史はまさにソフトウェア生産技術の歴史である（図-1）．

CASE 環境の最初の段階は，高級言語活用の時代である．マシン語によるプログラム開発方式の非生産性，低品質を改善するために，さまざまな高級言語が提案された．現在，使用されている高級言語の多くはこの時代に提案されたものである．コンパイラがこの時代の CASE 環境の主要素と考えられる．

次に方法論活用の時代となった．コンピュータの普及が開始されるにともないコンピュータへの要求が増大し，大規模なソフトウェアの開発が積極的に行われるようになった．このようなソフトウェア需要の拡大，ソフトウェアの巨大化により，“ソフトウェア危機”が認識され，ソフトウェアの作り方のあるべき姿の追求が始まった．第一の成果が，プログラムを構造化して構成するべきであるという構造化プログラミングの考え方である．その後，構造化の設計技法や分析技法が提案された．その流れは，下流から上流，個別から全体へである．すなわち，製造工程のための方法論から設計工程，要求分析工程の方法論へ発展すると同時に，ソフトウェアライフサイクルという考え方にみられるように，開発工程ごとに作業と生産物を規定し，これに基づいて技術の体系化を図る考え方が主流となってきた．このように多くの技法が開発されたものの，ソフトウェア開発ツールは，依然，テキストエディタやデバッガが中心であった．

そして，ツール活用の時代に突入した．開発された技法を実際のソフトウェア開発に適用する試みが増すにつれて，技法に対応したツールが開発されるようになった．ワークステーションの普及，日本語やウィンドウなどのユーザインタフェースの向上とともに，ツールが積極的に使用されるようになった．しかし，さまざまなツールが独立に個別に開発されたため，ツール間でデータ，操作性，対象作業が異なるという問題が表面化してきた．このため，ツールの統合化（統合

† Overview of CASE Environment by Kiichi FUJINO (NEC Corporation).
†† 日本電気(株)

	50年代	60年代	70年代	80年代	90年代
CASE 環境	高級言語活用		方法論活用	ツール活用	知識活用
対象工程	プログラミング		設計 テスト	要求定義 管理	統合化
対象メディア	テキスト 表		図	マルチメディア	
ツール	アセンブラ コンパイラ		エディタ デバッガ	図形エディタ 統合化ツール	知識エディタ
技法	モジュール化		抽象化 構造化プログラミング 構造化設計・分析	オブジェクト指向	
言語	マシン語 COBOL	FORTRAN LISP	PL/I C	PROLOG 4GL	C++
コンピュータ利用環境	バッチ処理		TSS	分散処理	ワークステーション

図-1 CASE 環境の歴史

CASE) へと技術開発が進められている.

本解説では，このような CASE ツール統合化段階における CASE 環境の状況を研究開発動向と標準化動向を中心にその概略を説明する．さらに，CASE 環境の発展は，究極には CASE 環境を対象，目的などに応じて構築するメタ CASE 環境につながっていくと考えられる．このメタ CASE とその実現のために必要な技術を以降で簡単に説明することにする.

2. CASE 環境について

狭義の意味での CASE は，ソフトウェア開発の上工程（要求分析，設計など）を主に支援するツール群といえるが，上工程の結果が正しく下工程に反映される必要があることや，複数の技術者が共同で作業するのを支援するために，より広義の意味でのソフトウェア開発支援環境の必要性が認識されてきた．CASE 環境とは，ソフトウェア開発の全作業をコンピュータによって支援する統合的なシステムであるといえる.

このような広義の意味での CASE 環境は多くの要素からなるが，大きく技術ツール群，管理ツール群，支援ツール群に分けられる[1]．Jones[1] によれば，必要なツール全体で，だいたい 100 ほどの異なった機能をもつツールが必要になり，その内訳は，技術が 60，管理が 20，支援が 20 という分類になる．また，狭義の CASE ツール群は，そのうち，約 10〜15 を満たすにすぎない，という.

技術ツール群は，ソフトウェアの開発や保守に携わる技術者のいろいろな作業を支援するもので，現在市販されている多くの CASE ツールがこれに分類される．従来，ソフトウェア開発の多くの作業は，紙と鉛筆でなされてきた．そこでは，一般的な文章のほか，図面や，表，フォームシートなどが使われてきた．技術ツールは，これらの文書化作業をコンピュータによって支援し，さらに，結果の正しさの保証，ソースプログラムとの整合性検査などの機能を含んでいる．技術ツールの支援作業を表-1 にリストアップする.
ツールとしては，さらにコンパイルやリンカなど，通常に使われるものが加わる.

管理ツールは多くの技術者が共同で行うソフトウェア開発作業の管理的側面を支援するもので，表-1 に示す作業が対象となる.
現在，すでに各種の管理ツールが市販されているが，その多くは個人的なツールで，共同作業管理支援までにはいたっていない.

支援ツールは，技術ツールと管理ツールを結合して互いに連携をもたせるためのもので，表-1 に示すような要素からなる.

表-1 に示した以上の機能の多くは，対象ソフト，対象業務，プロジェクト規模，要員レベルなどによって適切なものを選択する必要があり，CASE 環境といっても非常に多くのバラエティがあるといえる.

一方，CASE 環境を CASE システムの概念的な構成で考えると，ユーザインタフェース系，オブジェクトベース系，解析系の三つに分けてとらえることができる.

CASE システムのユーザである開発者が作成する開発文書の入出力を行うのがユーザインタフェース系であり，ユーザインタフェース系を介して得られたさまざまな情報を保持するのがオブジェクトベース系である．解析系は，オブジェクトベース系に保持された

表-1 CASE 環境[1]

技術ツール	管理ツール	支援ツール
要求分析	規模見積り	GUI ツール*1
データフロー分析	スケジュール管理	電子メール
機能分割設計	作業分割計画	ボイスメール*2
設計仕様書作成	コスト見積り	通信機能*3
文書の版構成管理	品質見積り／評価	共通データベース*4
標準設計の再利用	プロジェクト予算管理	ネットワーク*5
新しいコード作成	マイルストーン管理	従来環境との
既存コード再利用	コスト追跡	連携機能*6
既存コードの解析修正	誤り追跡	セキュリティ機能
既存コードの再構造	生産性／品質尺度測定	
未使用コードの削除	要員管理	
設計レビュ	製品出荷管理	
コードインスペクション	その他	
デバッグ		
テストケース作成		
テストタイプライアン管理		
誤り解析		
ユーザマニュアル作成		
オンラインヘルプ		
ユーザ訓練，教育		
その他		

*1 GUI (Graphical User Interface) ツール：各種ツールを共通のユーザインタフェースで使えるようにする
*2 技術者間の連絡のために必要
*3 ツール間のデータ交換のため
*4 開発成果（設計結果，プログラム，テストデータほか），管理情報，ノウハウなどを統合的に管理するため
*5 分散開発環境のためのネットワーク
*6 CASE 以外の一般のツール，従来環境との間の連携機能

情報をもとに解析，変換などを行い，さまざまな付加価値をつけるためのものである．

3. CASE 環境の研究開発動向

ここでは，CASE 環境の研究動向を，2.でのべたユーザインタフェース系，オブジェクトベース系，解析系の三つとこれらのインテグレーションという点から概説する．

3.1 ユーザインタフェース系

ユーザインタフェース系は，開発者が直接，CASE システムと接する部分である．したがって，開発で使われるさまざまな表記やマシン・ソフトウェア環境などと密接に関係している．開発者はグラフィックスや言語などのメディアを介して CASE システムに入力し，CASE システムはグラフィックスや紙などのメディアを介して開発者に出力する．

現在，最も広く利用されている入出力メディアとして，グラフィカル・ユーザインタフェース (GUI) がある．GUI は，(マルチ) ウィンドウシステムを基盤とし，ウィンドウシステムをベースに "look & feel" のガイドラインとツールキットを提供する汎用 GUI がよく利用されている．この汎用 GUI の例としては，Motif, OPENLO OK, プレゼンテーションマネジャなどがある[4]．

また，日常使われている図形は，(1)表系 (行列系)，(2)網図系，(3)領域系，(4)座標系，(5)シンボル系，(6)絵画系に分類される[3]．

このうち，ソフトウェア開発では，表系，網図系，領域系，座標系がよく使われている．このようなソフトウェア開発でよく使われるダイアグラム，表，イメージなどを提供する CASE 向け GUI としては，Athena, Interview, 鼎などがある[4]．たとえば，鼎では，テキストエディタのほかに，テーブルエディタ(表系)，ダイアグラムエディタ(網図系，領域系)，木構造エディタ (座標系)，ビットマップエディタ(絵画系) を提供している[5]．

一方，入力メディアとして言語を対象としたものは，単なるテキスト処理（テキストエディタ）から，プログラミング言語や仕様化言語などの形式言語処理，さらに自然言語処理を含めたものまで多岐にわたる．特に，日本語処理においても，日本語フロントエンドのように構文情報の解析を行う汎用的な処理系[19]と，さらに開発仕様の獲得までを目的とした処理系[18]がある．

3.2 オブジェクトベース系

ユーザインタフェースを通して入力された情報を蓄積するのがオブジェクトベース系の役割である．オブジェクトベースは，ユーザインタフェースを通して得られた情報の構造や，情報の利用形態によって，影響を受ける．CASE システムでも，一般の情報処理システムと同様に，CODASYL 型のネットワークデータベースや，関係データベースなどが広く利用されている．ソフトウェア開発では，さらに，開発された情報を単に保持するだけでなく，版管理，構成管理などが要求される．版管理を対象としたシステムとして，ファイル単位の版管理を行う SCCS (Source Code Control System)[14] や RCS (Revision Control System)[15] がある．また，構成管理を対象としたシステ

表-2 CASE 環境

ユーザインタフェース系
GUI
汎用 GUI
CASE 向け GUI
言語
テキスト
形式言語
日本語
オブジェクトベース系
汎用データベース
CODASYL 型
関係型
版管理
SCCS
RCS
構成管理
Make
解析系
PSA, SREM, REVS…
インテグレーション
CASE アーキテクチャ

ムとしては，UNIX の Make が知られている．その他，多くの版管理・構成管理に関する研究開発が行われている[16),17)].

パーソナルコンピュータやワークステーションでは，ユーザインタフェース系の入出力メディアの多様化が著しい．このため，オブジェクトベースに格納される情報構造も，従来のテキストベースの論理情報のほかに，マルチメディア化への対応が現在強く要求されている．

3.3 解 析 系

ユーザインタフェース系とオブジェクトベース系だけでは，単なる開発文書システムあるいはプレゼンテーションシステムにすぎない．開発者が入力した開発情報に，さまざまな付加価値をつけるのが解析系である．解析系は，CASE 構成要素の中でも，最も対象ドメインに依存したものである．対象とするシステムの特徴（ビジネスアプリケーション，リアルタイムなど），開発工程，開発パラダイム，実現方式，開発マシン環境などによって，多種多様の解析システムが存在する．

解析系では，以下のようなアプローチが考えられる．

- 高度な検索
- 検　　証
- シミュレーション（静的特性，動的特性）
- 変換，自動生成

たとえば，要求分析工程を対象とした解析系として，ISDOS の PSA (Problem Statement Analyzer) や SREM[6], REVS (Requirements Engineering and Validation System)[7] などが古くから知られている．

ユーザインタフェース系とオブジェクトベース系が開発情報の構文や構造を扱っているのに対して，解析系は開発情報の意味を扱っているといえる．知識工学の成果をソフトウェア開発に応用した研究が数多く行われており，その成果が期待されている[8)]．

3.4 CASE 環境のインテグレーション

CASE 環境を三つの視点に分解して捉えてきた．これらの視点に用意された個々の機能が有機的に結合されてはじめて，一つの CASE 環境として実現される．有機的に結合した CASE 環境の方向性を示すために検討されているのが CASE 環境アーキテクチャである．たとえば，Penedo らは，CASE アーキテクチャを（1）ハードウェア，実 OS レイア，（2）環境サポートレイア，（3）ツール・キャパシティレイア，（4）環境適合，プロジェクトユーザサポートレイアの4つのレイアとして捉えている[9)]．

また，Wasserman は，CASE 環境のアーキテクチャを，開発工程や開発技法に対応した水平ツールと，ライフサイクル全体に共通する支援機能をレイアとして捉えた垂直ツールとの視点から捉えている[10)]．現時点では，これらのアーキテクチャは，概念的な枠組として議論されている段階であるが，今後の重要課題の一つである．

4. CASE 環境の標準化動向

CASE 環境の標準化について，CASE ツール統合化，仕様記述言語などに着目して簡単にその状況を説明する．

4.1 CASE ツール統合化

CASE 環境を具体的に統合化するには

- ツールの統合化
- 開発方法論，技法の統合化
- 対象領域の統合化

などいろいろなアプローチがある．そのなかでも CASE 環境で使用する道具（ツール）の統合化が最も重要かつ早急に進められるべき課題である．

CASE ツールの統合化には，大きく三つのテーマが考えられる．

（1） ツール間インタフェース

現在開発されている CASE ツールのほとんどが，ソフトウェア開発工程を部分的にしか支援していな

い．したがって，CASE の最終目標であるソフトウェア開発の一貫支援を実現するためには，各 CASE ツール間でのインタフェースを統一し，ファイルやデータの交換を円滑にする必要がある．

特に，事務処理分野ツールでは，各ツール間のアーキテクチャを規定・公開することで，ツールの互換性やツール体系のオープン化を図っている場合が多い．Σツールでもそのようなアーキテクチャが規定されている．

(2) 共通ソフトウェア DB

各 CASE ツールの成果物（出力）を一元的に管理するためのソフトウェア DB（リポジトリとも呼ばれる）を設定し，これに対するアクセス・ルールを規定すると，上記(1)を含めて，各 CASE ツール間でのデータ交換，データ管理の統合化が図れる．多くの米国製の CASE ツールは，ソフトウェア DB へのアクセス用ライブラリや関数を提供し，他の市販ツールとの統合を可能としている．

(3) ユーザ・インタフェース

複数ある CASE ツールを一つの CASE 環境として統合化するためには，各ツールのユーザ・インタフェースを統一することも大切である．現在，UNIX の標準化活動の一つとして，WYSIWYG (What You See Is What You Get) 方式の GUI (Graphical User Interface) の標準化活動が活発に進められている．CASE ツールもこれらの標準的な UI を利用することで，ツールの操作性が向上するだけでなく，ツール自体の生産性も向上するようになる．また，利用者にとっては，CASE ツールの導入・利用が容易になる．

4.2 仕様記述

一般の CASE 環境には，通信システムの標準的な形式仕様記述言語である SDL, LOTOS などに相当するものは今のところない．たとえば，上流 CASE でよく使われるデータフロー図に対しそのプロセスの記述方法（記述言語）については，特に標準化されていないのが現状である．各ツールで独自に規定していたり，汎用言語を用いたり，さまざまである．しいていえば，上流 CASE の道具であるデータフロー図，E—R (Entity-Relationship) 図，状態遷移図などが CASE ツールにおける仕様記述の道具として共通化されている，といえる．ただし，各種の方言は存在する．

詳細仕様を記述する方法としては，各種のチャート記法があり，JIS, ISO などで標準化活動が進められている．チャート記法としては，HCP や YAC II, PAD, SPD などを採用していることが多い[20]．

4.3 標準化のための活動

CASE 環境の標準化については，現在 ISO, IEEE などでの活動，欧州における IPSE (Integrated Project Support Environment) などがある．しかし，現時点では CASE ベンダによる自社開発 CASE ツールを取り巻く環境での標準化が一歩進んでいる状況である．また，日本では，昭和60年に開始した Σ プロジェクトにより，Σ システム（Σ ツール，Σ WS, Σ OS, Σ ネットワークの総称）という我が国の標準的なソフトウェア開発環境が構築された．

最後に，CASE 環境の標準化には，Ada の APSE (Ada Programming Support Environment) や UNIX の標準化活動などのように，(狭義の) CASE 環境というよりは，広義の CASE 環境ともいうべきソフトウェア開発環境としての標準化が必須であると考えられる．そして，その結果を狭義の CASE 環境にも随時，適用していく必要があると考える．

5. 今後の方向

現在多くの，CASE ツールが使用されているが，まだ十分な効果をあげているところは多くはないように見受けられる．これは，使う側が十分にはまだ慣れていないなどの理由もあるが，さらに，現在の CASE の機能自体にも不十分なところが多いためともいえる．今後は，CASE の個別のツールがより高機能になっていくであろう．そのためには，ベースとなる方法論がもっと研究されるべきである．現在の多くの CASE ツールは，データフロー図やモジュールのコール関係図などに象徴される構造化分析／設計法をベースにしているが，最近のオブジェクト指向設計や，多くのウィンドウシステムにみられるイベントドリブンな設計に対しては，より適切な方法論とツールが必要となろう．

また，CASE 環境構築には非常に多くの要素技術が必要となり，そのすべてを単一のシステムでまかなうことは難しくなっている．むしろ，いろいろなツールを組み合わせて CASE 環境を構築する傾向が，今後ますます強くなってくるであろう．すなわち，CASE 環境の構築は，ソフトウェア開発という業務を対象としたシステムインテグレーション作業になってくるであろう．そこでは，異なったいろいろなツール

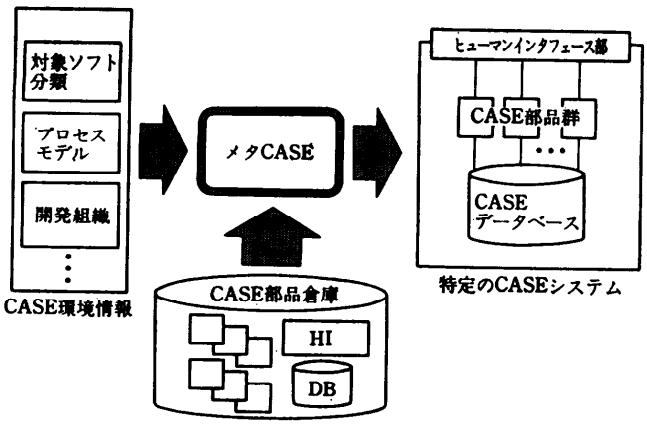

図-2 メタ CASE による CASE 構築環境

を有機的に組み合わせる技術が重要になる．3.の研究開発動向や4.の標準化動向で述べた技術の多くは，この組合せ技術に関連している．

これらの CASE 要素技術，要素の組合せ技術を使って実際の CASE 環境を構築する場合，まず対象業務分析（ソフトウェア開発作業分析）から始まり，そこで必要なコンピュータネットワークの設計，作業標準の設計，人間・機械系の設計，必要なツール群の選択，ツールの組合せ設計，独自に必要なソフトウェアの開発等々を進めることになるが，これはまさしく CASE 環境構築そのものが，CASE 環境の支援対象でもある．このような，CASE 環境構築のための CASE 環境を，仮にメタ CASE と呼ぶことにする．図-2 にメタ CASE の概念を示す．メタ CASE は，支援対象が CASE 環境に限定されており，より専門的な高度な機能の提供が可能になるであろう．たとえば，ソフトウェア特性，組織，要員特性，予算などをもとに，最も適切な CASE 環境，作業標準規定，必要な組織形態，教育訓練計画などを提案し，実現するための支援機能を含むであろう．

メタ CASE のベースとなる技術として，プロセスモデリングやプロセスプログラミングの研究がある．そこでは，ソフトウェア開発のプロセスを形式的にある種のプログラムで記述し，その結果，そのプロセスにあった開発環境を生成する．CASE 要素技術や要素組合せ技術の進展と，現実のソフトウェア開発作業の分析によるノウハウの蓄積が進むことにより，プロセスモデリングの研究成果はより実際的な場で活用できるようになるであろう．

さらに，将来的には，CASE 環境を含むソフトウェア開発組織（仮にソフトウェアファクトリィと呼ぶことにする）の全体の設計・実現の技術が必要になろう．そこには，研究開発部門，営業支援部門，システムインテグレータ部門，ソフトウェア開発部門，リリース管理部門，ユーザ教育部門などがあり，その間で必要な情報交換が緊密になされ，そのためのコンピュータネットワークによる支援環境（広義の CASE 環境）が完備される．このような環境の実現には，適切なソフトウェアファクトリィを構築するための技術，すなわちソフトウェアファクトリィ・エンジニアリングが必須となる[2]．

ソフトウェア開発は，現在のコンピュータハードウェア技術，通信技術などの急速な進展の成果をまだ十分に活用しているとはいいがたい．これらのC＆C技術を十分有効に活用することによって，今後の CASE 環境は，より一層，使いやすくて強力な機能をもたせることができるであろう．

6. おわりに

CASE 環境についての概略を歴史，現状，将来の観点からおのおの述べてきた．ここで述べなかったが，非常に重要なことは，CASE 環境の将来にとって重要なものは，「知識」であるという点である．ソフトウェアの品質向上のためのさまざまな工夫，ノウハウや高度なソフトウェアを実現するための構造，方式などの専門知識，などの多岐にわたる「知識」を効果的に蓄積し活用することが今後重要となろう．先に述べたメタ CASE，すなわち CASE 環境構築のための CASE 環境もさまざまな「知識」の集大成をしなければ効果的に実現できない．この意味で1.で述べたツール活用の時代の次には，「知識」活用の時代に向かっていくことであろう．技法によりソフトウェア生産の指針を与え，ツールにより生産性向上，品質向上の具体的な成果を達成したのち，技法やツールの高度化へと進展する．すなわち，専門化した高度な知識の活用により生産ノウハウの蓄積・体系化が図られよう．CASE 環境は「知識」を付加していくことで大きく成長し，成熟していくことが期待できる．

謝辞 本解説の作成にあたり，日本電気(株)ソフトウェア生産技術開発本部川越恭二，小山田正史，

岡田雄一郎の各氏，ならびに NEC アメリカ紫合 治氏のご協力に感謝します．

参考文献

1) Jones, C.: Why Choose CASE?, International Edition of BYTE Magazine (Dec. 1989).
2) Fujino, K.: Concept of Software Factory Engineering, NEC R & D, No. 94 (July 1989).
3) 吉川編：コンピュータグラフィックス論，日科技連 (1977).
4) マルチメディア時代のユーザ・インタフェース，日経コンピュータ別冊ソフトウェア，日経 BP (July 1989).
5) 暦本他：X ウィンドウ上のマルチメディアインタフェース構築環境：鼎，情報処理学会プログラミングシンポジウム (1989. 1).
6) Alford, M.: A Requirements Engineering Methodology for Real-Time Processing Requirements, IEEE Trans. on SE. Vol. SE-3, No. 1 (Jan. 1977).
7) Bell, T. E., Bixler, D. C. and Dyer, M. E.: An Extendable Approach to Computer-Aided Software Requirements Engineering, IEEE Trans. on SE. Vol. SE-3, No. 1 (Jan. 1977).
8) 玉井：ソフトウェア開発への知識工学の応用，情報処理，Vol. 28, No. 7 (July 1987).
9) Peneo, M. H. and Riddle, W. E.: Guest Editors' Introduction Software Engineering Environment Architectures, IEEE Trans. on SE, Vol. 14, No. 6 (June 1988).
10) Wasserman, A. I.: CASE in the '90's, ソフトウェア・ツール・シンポジウム '90 講演資料 (Jan. 1990).
11) Barstow, D. R., Shrobe, H. E. and Sandewall, E.: Interactive Programing Environments, McGraw-Hill (1984).
12) 斎藤：ソフトウェア開発環境，情報処理，Vol. 28, No. 7 (July 1987).
13) 松本：ソフトウェアエンジニアリングデータベース SEDB/OKBL のデータモデルについて，情報処理学会論文誌, Vol. 30, No. 9 (Sep. 1989).
14) Rockhind, M. J.: The Source Code Control System, IEEE Trans. on Software Engineering, SE-1 (4) (Dec. 1975).
15) Tichy, W. F.: RCS: A Revision Control System, Integrated Interactive Computing Systems, North-Holland (1983).
16) Leblang, D. B. and Chase, R. P.: Computer Aided Software Engineering in a Distributed Workstation Environment, Proc. of ACM SIGSOFT/SIGPLAN Sympo. on Practical Software Development Enviroment (Apr. 1984).
17) Babich, W. A.: Software Configration Management, Addison-Wesley (1986).
18) 辻井，上原：ソフトウェア工学と自然言語処理，情報処理，Vol. 28, No. 7, pp. 913-921 (July 1988).
19) 杉山他：例文からの文法獲得に基づく日本文インタフェース構築ツール「ゆい」，電子情報通信学会研究会資料，NLC 89-8 (1989).
20) たとえば，「コンピュータソフトウェア事典」，丸善 (1990).
21) 藤野：CASE 環境の現状と動向，情報処理学会 CASE 環境シンポジウム予稿集 (1989. 3).

(平成 2 年 4 月 20 日受付)