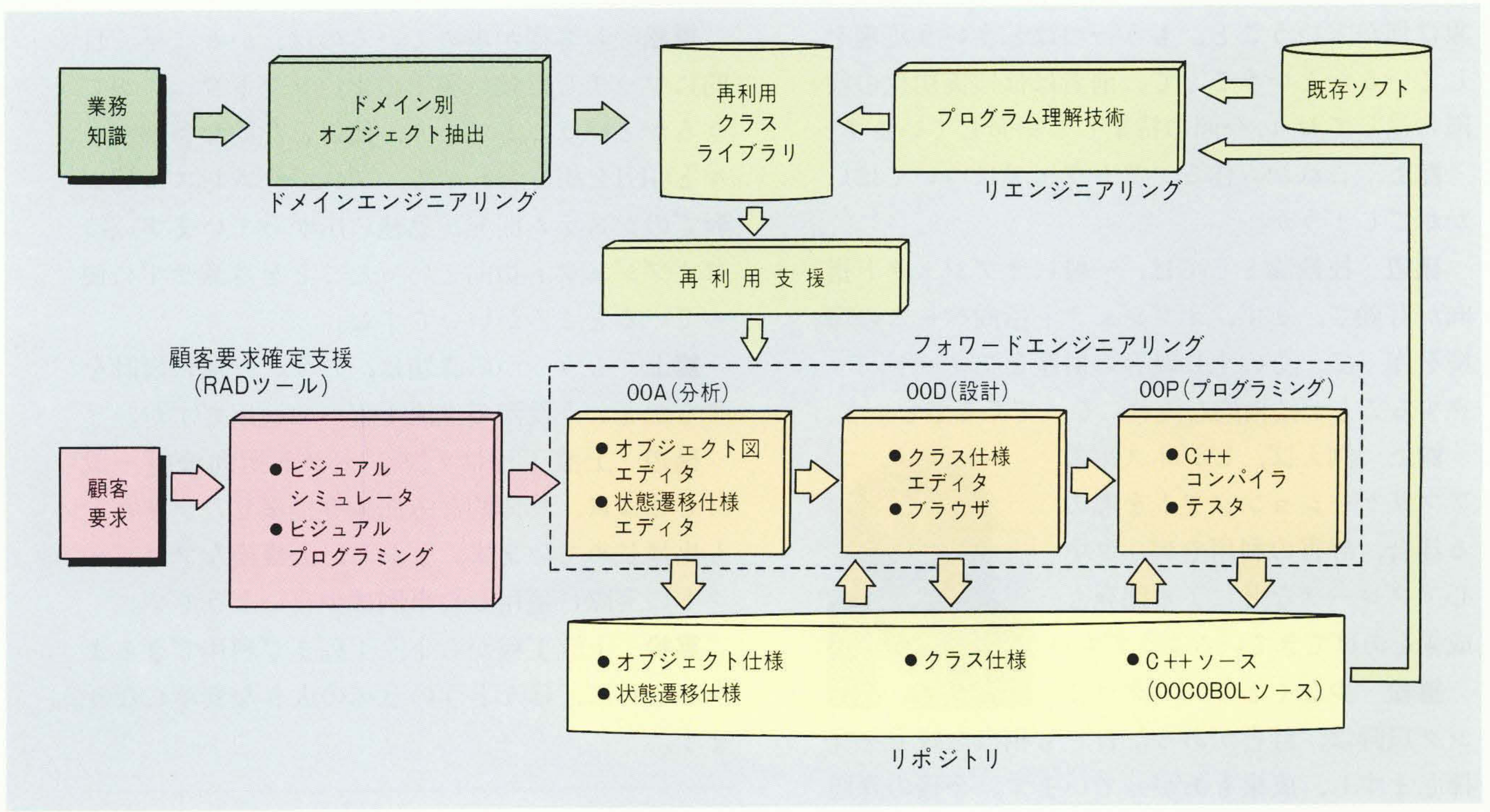
ソフトウェア生産技術の展望

―ソフトウェアエンジニアリングの現状と将来―

Current and Future Aspects of Software Engineering

増位庄一*　*Shôichi Masui*
葉木洋一**　*Youichi Hagi*
梶山雄三***　*Yûzô Kajiyama*
今城哲二****　*Tetsuji Imajô*



注：略語説明ほか　⇨ 開発作業の流れ
RAD (Rapid Application Development), 00A (Object-Oriented Analysis), 00D (Object-Oriented Design), 00P (Object-Oriented Programming), 00COBOL (Object-Oriented COBOL)

オブジェクト指向CASEの概念　オブジェクト指向は再利用を前提としたソフトウェア生産のための新しいパラダイムである。顧客要求確定支援のためのRADツール，ドメインエンジニアリングによるクラスライブラリの準備によって飛躍的な生産性の向上が期待できる。

ソフトウェアを効率よく生産するということは，コンピュータ出現以来の永遠の課題である。ソフトウェア生産技術者は，その課題にこたえる種々の要素技術を提供することに努力してきた。今，それらの要素技術が組み合わされ，パラダイムシフトを伴う新しいソフトウェア生産技術が生みだされようとしている。オブジェクト指向という従来とは異なるパラダイムに基づくソフトウェア生産支援システム(CASE：Computer Aided Software Engineering)である。

そこでは，RAD(Rapid Application Development)と呼ばれる要求仕様の早期確定と，スパイラル型のソフトウェア開発手順を含む新しい生産方式の実現も意図されている。さらに，ソフトウェア生産ツールの統合化を目指しての世界標準づくりも進行中である。このようなダイナミックな展開を見せているソフトウェア生産技術について，その現状と将来を展望する。

* 日立製作所 システム開発研究所　** 日立製作所 ビジネスシステム開発センタ　*** 日立製作所 情報システム事業部
**** 日立製作所 ソフトウェア開発本部

1　はじめに

ソフトウェアとは何か。それはコンピュータというハードウェアを目的に沿って望みどおりに動作させるための手順書である。コンピュータの基本動作である命令語は限られている。したがって，望みどおりに動作させるためには，ソフトウェア開発者がその望みをコンピュータのわかる命令語に分解してコンピュータに教える必要がある。この分解作業がソフトウェアの設計開発である。

ソフトウェアの需要は年々増大し，設計開発に必要な人員も急速に増大してきた。この状況の改善を図り，ソフトウェアを工業的な生産物として位置づけ，その効率的生産を目指す技術がソフトウェアエンジニアリング[1]である。だれもが容易にソフトウェアの生産作業に参加できるようにするために，ソフトウェア生産の手順の確立と効率的支援のためのツール開発を行うことがソフトウェアエンジニアリングの主な課題である。

ここでは，ソフトウェアエンジニアリングの現状と技術動向について述べる。

2　ソフトウェアエンジニアリングの現状

課題解決のための第一のアプローチは高級言語化である。ソフトウェア生産者がCOBOLやC＋＋などの高級言語を用いて記述した要求を，コンパイラが最適の命令語に自動的に分割する。このアプローチでは，要求を表現しやすい言語を準備することに力点が置かれてきた。第4世代言語[2]は成果のひとつである。これにより，コンピュータに自分が使っていることばで要求を伝えることができるようになった。

第二のアプローチはソフトウェア生産のパラダイムの確立である。要求をより小さな要求の組み合わせに分解していくという構造化技法[3]はひとつの成果である。また，「もの」を中心にソフトウェアを組み立てようとするオブジェクト指向は，最も新しいパラダイムの一つである。

第三のアプローチは，生産環境の整備[4]である。エディタ，ブラウザなど設計開発や正当性検証を支援する各種ツールの開発とそれらの統合化，連携化が図られてきた。また，ワークステーションの普及とGUI(Graphical User Interface)の発展により，ビジュアルな生産環境が安価に提供されるようになり，個々人の開発効率が大きく向上した。

第四のアプローチは，ソフトウェアの部品化あるいは再利用である。アプリケーションプログラムの汎(はん)用的な部分を部品としてあらかじめ用意しておき，同種のプログラミングの際にそれを利用するということはすでに実効をあげつつある。特にGUI関係は，部品の標準化および編集操作のビジュアル化が容易であることから，対話型GUI構築ツールUIBT(User Interface Building Tool)という形で製品化されている。また既存のCOBOLプログラムを解析し，データの名称統一を図るリエンジニアリングツールも実用化フェーズにある。

これらの**図1**に示すような開発努力により，ソフトウェアの生産性は初期に比べて飛躍的に向上している。しかし，この向上率はソフトウェアの需要の爆発的増大に対してはまだ小さく，バックログと呼ばれる開発待ちソフトウェアの増加を止めることはできなかった。

3　ソフトウェアエンジニアリングを巡る技術動向

ソフトウェアエンジニアリングの周辺環境は大きく変化している。対象とするハードウェアは，小型化傾向(ダウンサイジング)により，従来の大型プロセッサと端末から，ワークステーションやパーソナルコンピュータをネットワークで結合したクライアントサーバシステム(CSS：Client Server System)の形態へと変わりつつある。それに伴い，ソフトウェアの開発環境も，ワークステーション上でGUIを駆使した図式を中心としたものに変化している。これにより，エンドユーザー自身が作りたいものをみずから開発するエンドユーザーコンピュー

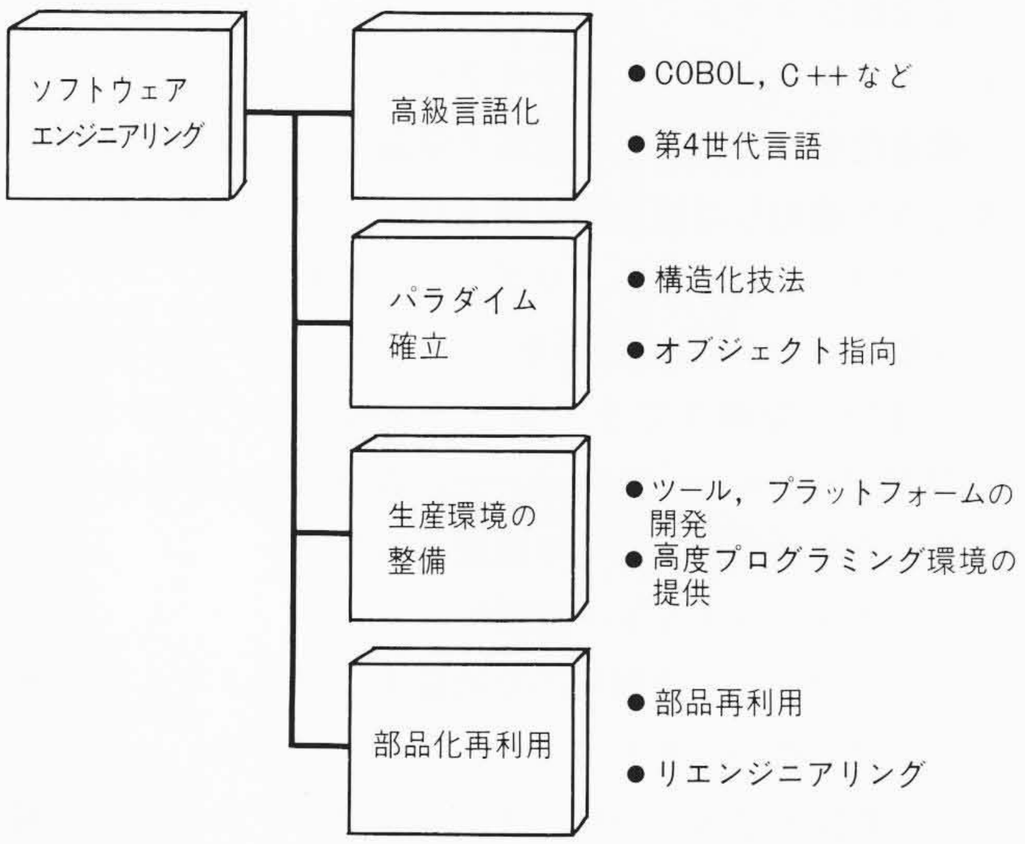


図1　ソフトウェアエンジニアリングの要素技術　　ソフトウェアを人間にわかることばで記述し，自動的に命令語に変換し，その動作を簡単にテストするとともに，考え方，生産物を再利用することがソフトウェアエンジニアリングの理想である。

ティングが可能となり，特に入出力画面などはユーザーが好みに合わせて自由に変更できるまでになってきた。一方，ハードウェアのオープン化は，ソフトウェア生産ツール自身のオープン化を促進し，ユーザーが自分の気に入ったツールを自由に組み合わせることを可能にした。このため，ツール間のデータ交換のための国際標準フォーマットCDIF(CASE Data Interchange Format)や，ソフトウェア生産活動から生じるドキュメント，プログラムを蓄積するためのリポジトリ(情報資源倉庫)としてのPCTE(Portable Common Tool Environment)[5]などが提案されている。

オープン化を支える標準化活動は，ISO(International Standard Organization)などの国際的標準化機関，学会，業界団体，コンソーシアムなど各所で取り組まれている。OMG(Object Management Group)が主導するオブジェクト指向アプリケーションのための分散処理環境CORBA(Common Object Request Broker Architecture)，OSF(Open Software Foundation)の開発したCSSのための分散処理環境DCE(Distributed Computing Environment)，ISOが制定したソフトウェアの品質保証システムのガイドラインISO9000-3[6]はその例である。

4 アプリケーション開発支援体系CAPSDF

図2に示すアプリケーション開発支援体系CAPSDF[7](Computer-aided Application System Development Framework)は，最新の成果を取り入れたソフトウェア生産支援システムである。

(1) ソフトウェアライフサイクルの一貫支援

構造化をベースとした構築方法論にライフサイクルの考え方を適用した統合的ソフトウェア開発環境の提供

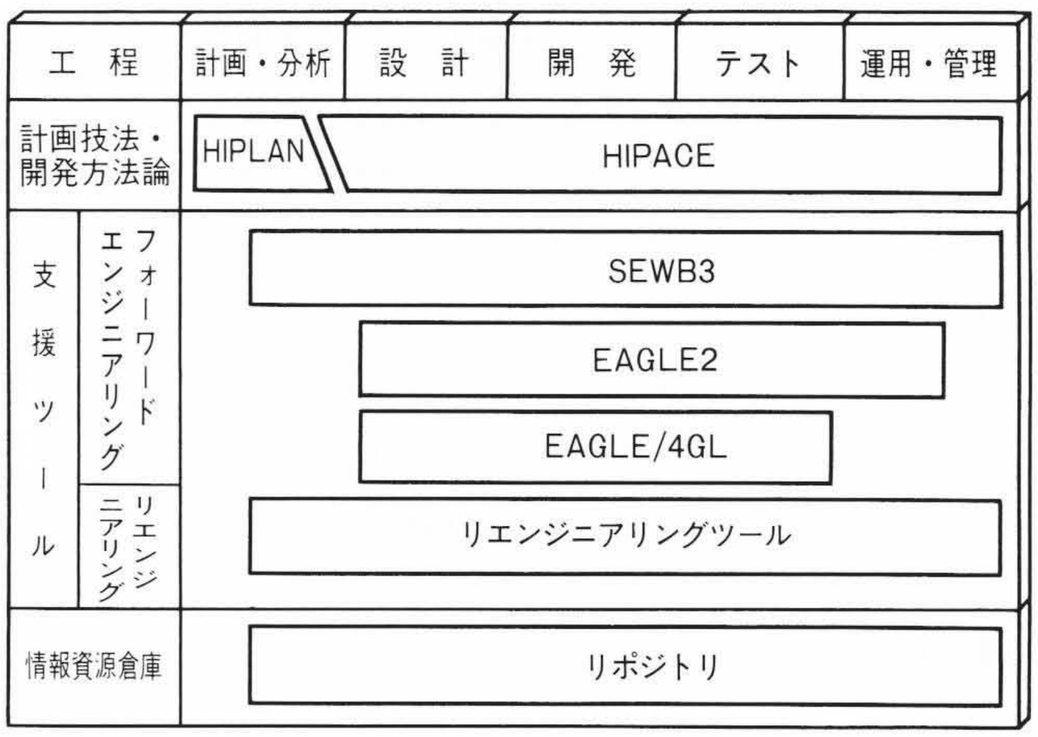
(2) プラットフォーム，リポジトリの整備

多数のツール間の連携をとり，ソフトウェアの分析から運用・管理までを一貫して支援するCASEツールSEWB3(Software Engineering Workbench 3)および再利用のためのリポジトリ機能の提供

(3) リエンジニアリング支援[8]

ソフトウェアの再利用のため従来のソフトウェアをオブジェクト指向に変形するラッパリング技術，ソフトウェアの意図や意味を分析しオブジェクト化のための情報を取り出すプログラム理解技術など，既存のソフトウェア資産を再活用するためのリエンジニアリングツールの提供

ソフトウェア構成



ソフトウェアの開発工程とツールサポートの関係

注：略語説明
CAPSDF (Computer-aided Application System Development Framework)
HIPLAN (Hitachi Integrated Planning Procedure for Information Systems)
HIPACE (Hitachi Phased Approarch for High Productive Computer System's Engineering)
SEWB3 (Software Engineering Workbench 3)
EAGLE2 (Effective Approach to Achieving High Level Software Productivity 2)
EAGLE/4GL (EAGLE/4th Generation Language)

図2 アプリケーション開発支援体系CAPSDF CAPSDFは，今までのソフトウェアエンジニアリングの成果の集大成である。

(4) 充実したプログラミング環境

EAGLE/4GL(Effective Approach to Achieving High Level Software Productivity/4th Generation Language)などの第4世代言語，C++などのオブジェクト指向言語，資産の多いCOBOLなどの各種言語とそのための高度なプログラミング環境の提供

5 オブジェクト指向技術

オブジェクト指向[9]は，**図3**に示すようにオブジェクトという実体と，そのオブジェクト間のメッセージによって所望の機能を実現するプログラミングパラダイムであり，

(1) 人間の思考との親和性がよい。

(2)　継承機能，カプセル化機能などプログラムの開発保守，再利用に有利な機能を提供できる。
などの優れた特長を持っている。

日立製作所はオブジェクト指向に関してもプラットフォーム，ツール，言語およびデータベースの開発からアプリケーションシステム，方法論の確立に至るまでの包括的な取組みを進めている。オブジェクト指向標準言語C＋＋とそのクラスライブラリ[10]の開発，ルールベースとオブジェクト指向を組み合わせた知的アプリケーション開発ツールES/KERNEL2[11](Expert　System/KERNEL2)をさらに発展させたオブジェクト指向システム構築ツールObjectIQの提供など，オブジェクト指向の実用化に向けた先進的製品をすでに提供している。

適切なオブジェクトのクラスライブラリの準備は，ソフトウェアの生産性をコーディングおよびデバッグが不要となることで一けた以上高める。また，クラスライブラリの持つ拡張性はプログラミング活動自身の再利用化を可能とし，作れば作るほど，作らなくて済むという状況を生み出す。このため，クラスの適切な設計法の確立，有能なライブラリアンの育成，大規模ライブラリの管理方法の開発など課題の解決を急いでいる。効果的なクラスライブラリは，当面，百科事典ではなく専門事典的色彩を持ったものになる。特定の分野(ドメイン)のソフトウェア分析を行うドメインエンジニアリングは，分野を限定することによって再利用性の高いクラスライブラリを得ることを目的としたアプローチで，その試行にも取り組んでいる。

6　次世代のソフトウェア生産を担う技術

ソフトウェアエンジニアリングは，ここしばらくCAP-SDFに代表されるウォータフォール型の開発方法論に加えて，クラスライブラリの整備によるプログラムレス開発方法論の確立を目指すことになる。それに加えて，オブジェクト指向の優れた抽象化能力は，ソフトウェアの仕様，動作を早期に決定し，その具体化を行うためのRAD(Rapid Application Development)技術の発展を促す。RAD技術とは，要求仕様の早期確定，ソフトウェア開発期間の飛躍的削減を可能とする技術の総称であり，次世代のソフトウェア生産を担う技術として期待されている。

オブジェクト指向に時間経過に伴う自動起動などの自律化機能を付加したエージェント指向も次世代技術として関心が高まっている。エージェントには，能動的プログラム部品としての役割が期待されており，事務処理分野の真のオートメーション実現への切り札となる。

またグループによる分散開発を可能にするため，CSCW(Computer Supported Cooperative Works)技術のソフトウェア生産への適用[12]が期待されている。これ

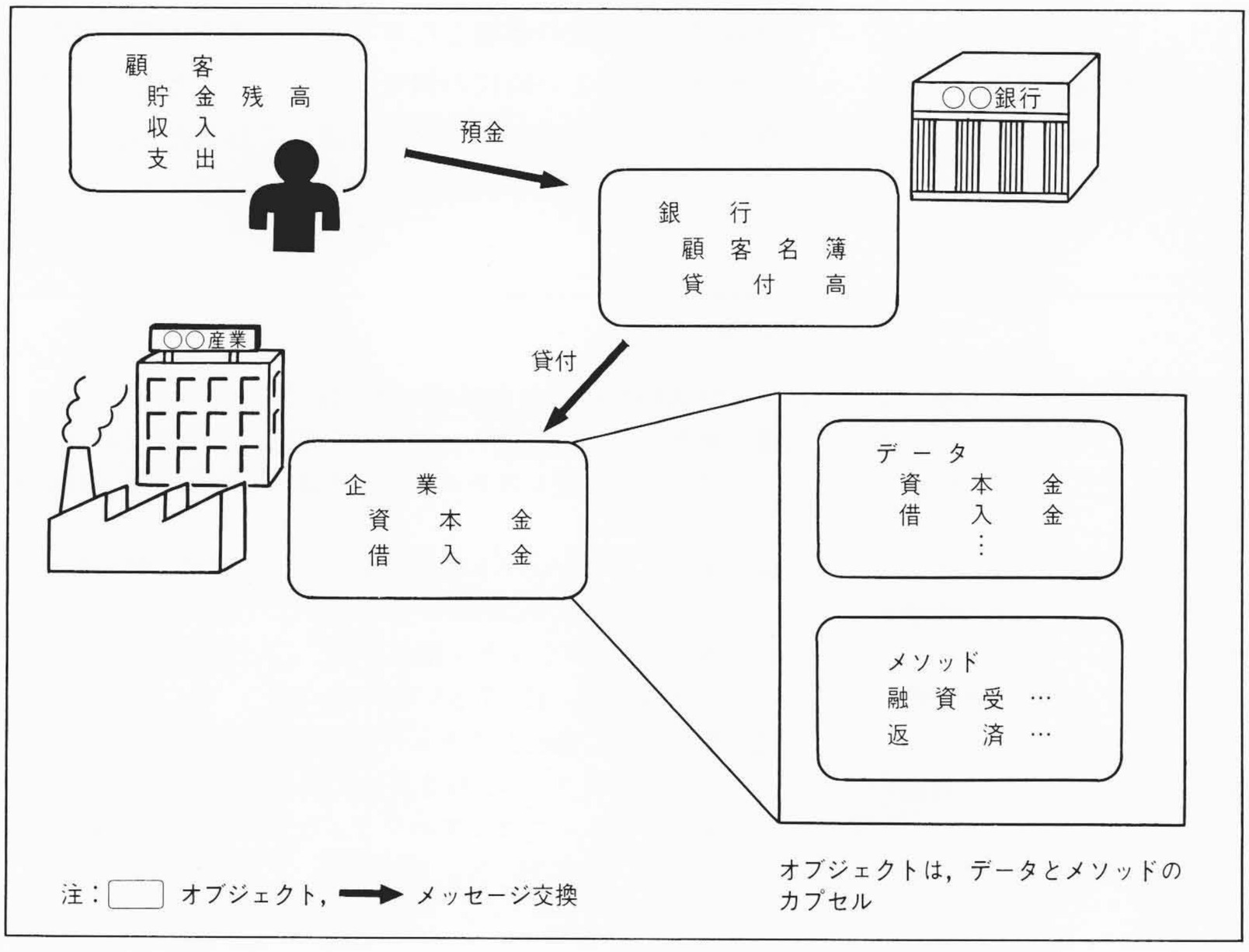


図3　オブジェクト指向の概念
オブジェクトとその間のメッセージ交換によって処理が進められるオブジェクト指向は，次世代のソフトウェア生産パラダイムとして有力視されている。

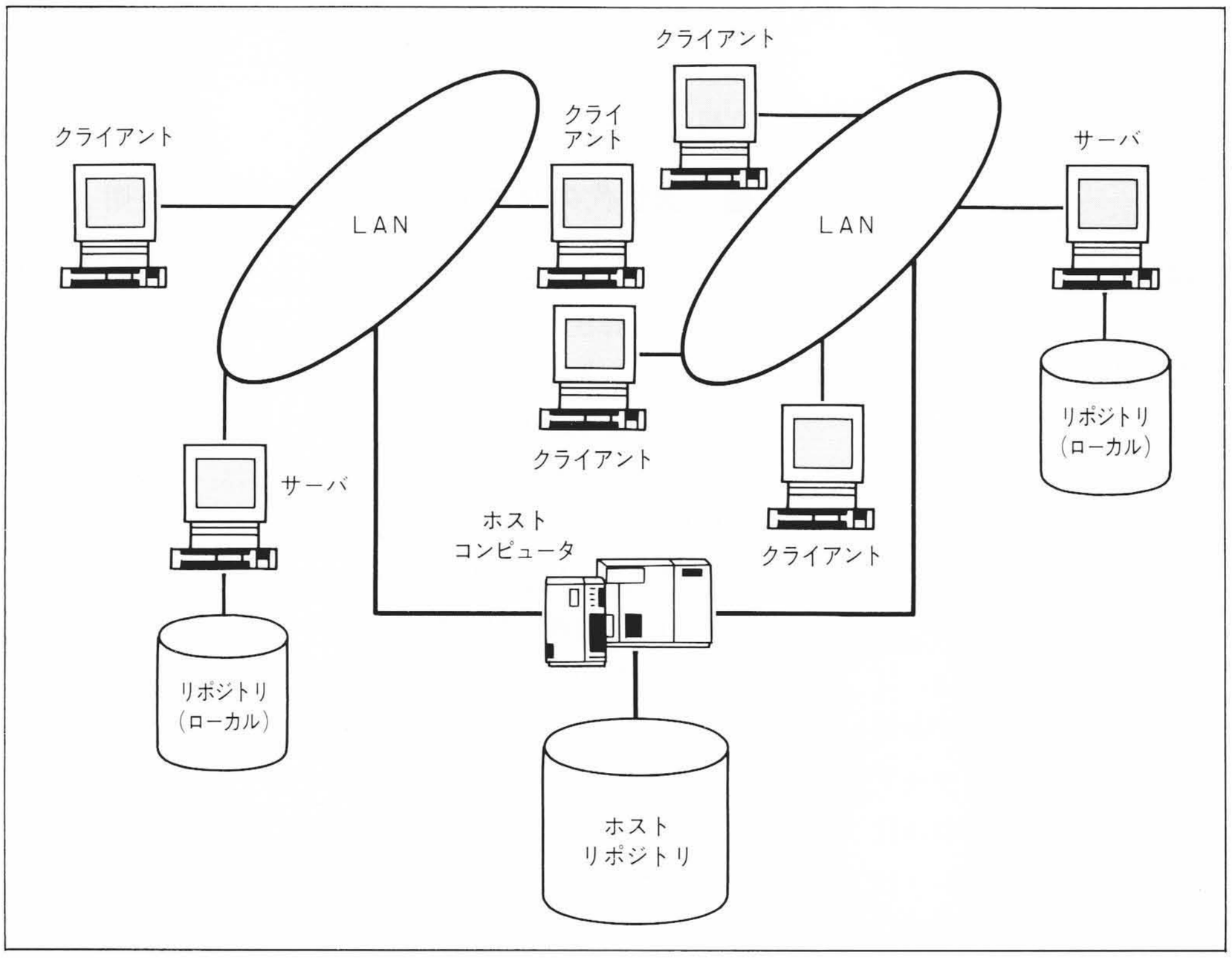


図4 CSSアーキテクチャ
ワークステーション，パーソナルコンピュータがLANで結合され，クライアント，サーバとして働くCSS(Client Server System)上のアプリケーション開発は，CSCWなどの新しい技術の導入によって実現できる。

は**図4**に示すようなCSSアーキテクチャ上のアプリケーションを，多人数で分散開発によるための大規模ソフトウェア開発手法の要素技術であり，その協調化のための手法に関心が集まっている。

7 おわりに

ソフトウェアエンジニアリングの現状と将来について概観した。この10年の進歩は予想をはるかに越えたものであり，今後の10年でもここで述べた以上の変化が生じるに違いない。ソフトウェアエンジニアリングは，「だれもが容易に」を合いことばに進展してきた。今後は「みんなで仲よく」という形で進歩が導かれるであろう。しかし，その後はどうなるであろうか。ひとつの可能性として，少数精鋭の復活，すなわちコンピュータのプログラムはごく少数の卓越したコンピュータウィザードだけが担当すればよい時代の到来がありうることを，ソフトウェア生産技術者の夢として指摘しておきたい。

参考文献

1) Fairley, R. E.：Software Engineering Concept, McGraw Hill(1985)
2) 葉木，外：システム開発支援ソフトウェア“EAGLE”，日立評論，**68**，5，373～378(昭61-5)
3) 藤野：CASE環境の概要，情報処理，Vol.31，No.8(1990)
4) 前澤：システムソフトウェアを対象にしたCASEの現状と動向，情報処理，Vol.31，No.8(1990)
5) 松本：CASE環境統合のためのインタフェースの標準化現状，情報処理，Vol.31，No.8(1990)
6) 飯塚：ソフトウェアの品質保障－ISO/DIS9000-3，対訳と解説－，日本規格協会(1990)
7) 西尾，外：日立製作所のアプリケーション開発支援体系“CAPSDF”，日立評論，**75**，11，715～720(平5-11)
8) 渡部，外：ソフトウェア資産を有効活用するリエンジニアリング支援システム，日立評論，**75**，11，741～744(平5-11)
9) 米澤：オブジェクト指向計算の現状と展望，情報処理，Vol.29，No.4(1988)
10) 森，外：オブジェクト指向言語C＋＋の開発支援環境，日立評論，**75**，11，755～760(平5-11)
11) 堺原，外：知的システムの構築を容易にするアプリケーション開発ツール，日立評論，**75**，11，761～766(平5-11)
12) 垂水：グループウェアのソフトウェア開発への応用，情報処理，Vol.33，No.1(1992)