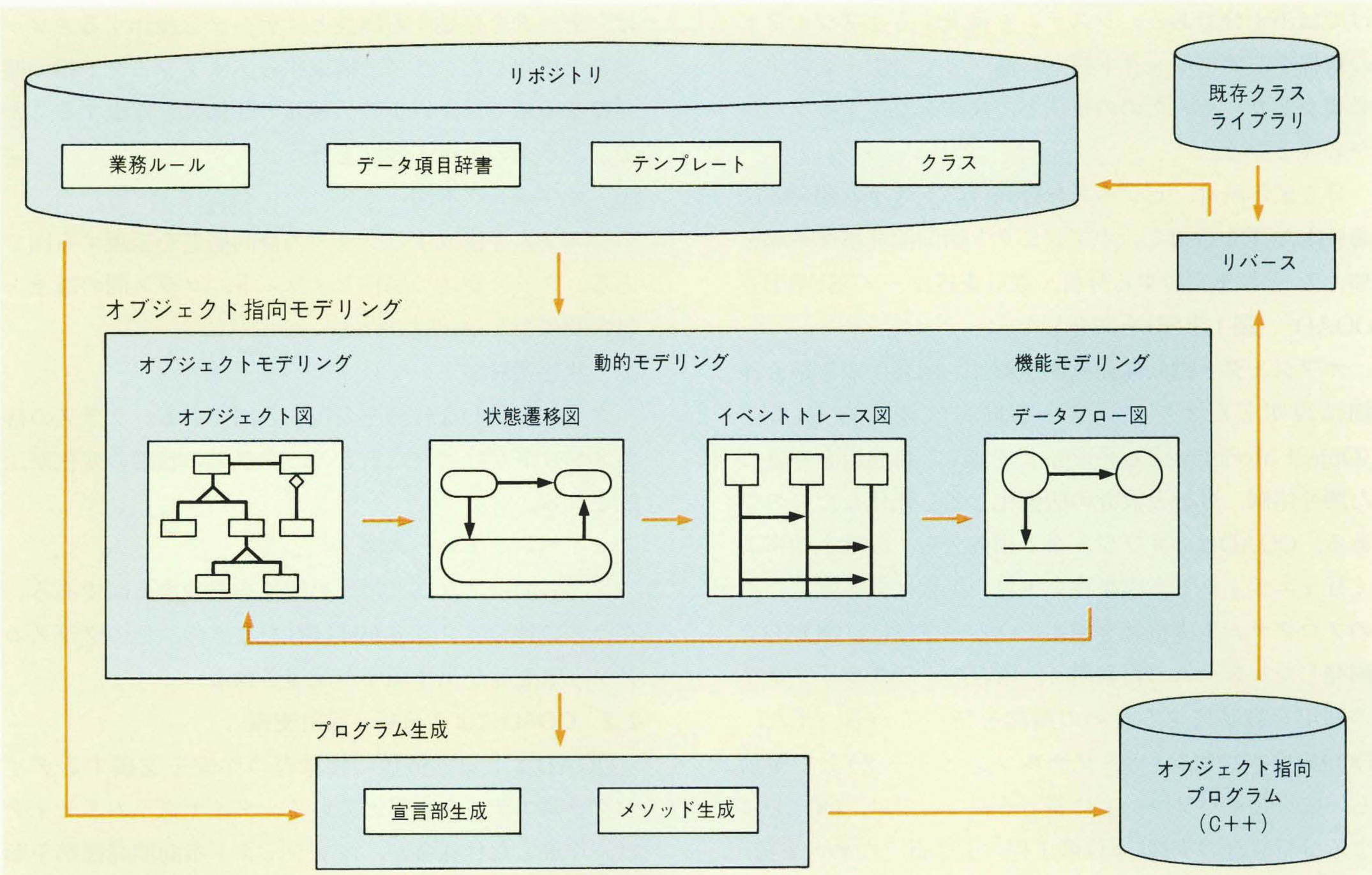
オブジェクト指向によるシステム分析・設計支援ツール“SEWB3/OOAD”

Object-Oriented Computer-Aided Software Engineering

高舘公人*　*Masato Takadachi*
友永佳津子**　*Kazuko Tomonaga*



SEWB3/OOADによるオブジェクト指向システム開発の流れ
HIPACE(Hitachi High-Pace)の中のオブジェクト指向開発標準手順による，上流から下流までの一貫したシステム開発を支援する。

日立オブジェクト指向開発標準手順に基づくオブジェクト指向分析・設計支援ツール“SEWB3/OOAD”(Software Engineering Workbench 3/Object-Oriented Analysis & Design)を製品化した。開発したツールは，オブジェクト指向分析・設計工程を支援する設計ドキュメントのエディタと，プログラミング工程を支援するプログラム自動生成ツールで構成する。

この支援ツールは，開発したプログラムの保守作業を容易にするために，変更個所を特定しやすくする情報や変更作業を軽減するための情報を，プログラムの開発中に蓄積する機能や，プログラム保守の際にそれらの情報を参照する機能を持つ。また，既存のプログラム部品(クラスライブラリと言う。)を利用することが開発の生産性を高めるので，その際に必要なクラスライブラリの設計情報を，プログラムコード内の定義情報(宣言部)から逆生成する機能を持つ。

以上の機能を持つこの支援ツールを活用することにより，再利用の容易なオブジェクト指向システムを短期間で開発することができる。

* 日立製作所 システム開発研究所　** 日立製作所 ソフトウェア開発本部

1 はじめに

オブジェクト指向技術が，再利用性の高いシステムを構築する技術として注目を集めている。しかし，再利用性の高いシステムを構築するためには，オブジェクト指向プログラミング言語を使用してシステムを実装するだけでは不十分である。システムを構成するオブジェクトの仕様を，オブジェクト指向に基づいて分析・設計する必要があり，そのための技法と，技法を支援するツールが必要となる。

日立製作所は，ビジネス分野向けのシステム開発方法論HIPACEを改訂し，オブジェクト指向開発標準手順を加え[1]，それを支援する分析・設計支援ツール“SEWB3/OOAD”（**図1**参照）を開発した。

オブジェクト指向開発標準手順は，開発手順を最も詳細に説明したオブジェクト指向方法論であるOMT (Object Modeling Technique)[2]を基に，再利用を考慮した開発体制，分析と設計の明確化などを強化したものである。OOADは，オブジェクト指向分析・設計を効率よく行うエディタと，成果物を下位の工程に引き継ぐためのプログラム生成機能を備えている。さらに，OOADで開発したシステムの再利用と，既存のクラスライブラリの利用を容易にするための機能を持っている。また，OOADはパソコン(パーソナルコンピュータ)上で稼動し，成果物をパソコン上に蓄える。ここでは，OOADによる分析設計の支援，下位の工程に引き継ぐための支援，および再利用の支援について述べる。

2 上流工程支援

この章では，オブジェクト指向開発標準手順で用いる仕様書と，OOADによる上流工程支援について述べる。

2.1 オブジェクト指向開発標準手順で用いる仕様書

オブジェクト指向開発標準手順で構築されたシステムは，データを格納する属性と，データを操作するメソッドを一体化したクラスで構成する。オブジェクト指向開発標準手順では，以下の三種類の仕様書を作成することで，クラスの分析・設計を行う。

(1) オブジェクト図

システムを構成するクラスの静的構造を表現する図である。クラスが持つ属性とメソッド，クラス間の継承・包含関係などが記述される。

(2) 状態遷移図

クラスの動的な仕様を表現する図である。クラスの持つメソッドが呼ばれたときの，クラスの状態の変化が記述される。

(3) イベントトレース図

クラス間のメソッドの呼出関係を表現する図である。クラスの持つメソッドが呼ばれたときの，他のクラスのメソッドを呼び出す順序が記述される。

2.2 OOADによる分析・設計支援

OOADは上記三種類の仕様書の作成を支援するダイヤグラムエディタを備えている。ダイヤグラムエディタは，作成した仕様書が，オブジェクト指向開発標準手順の規則に従っていることをチェックする機能を備えてい

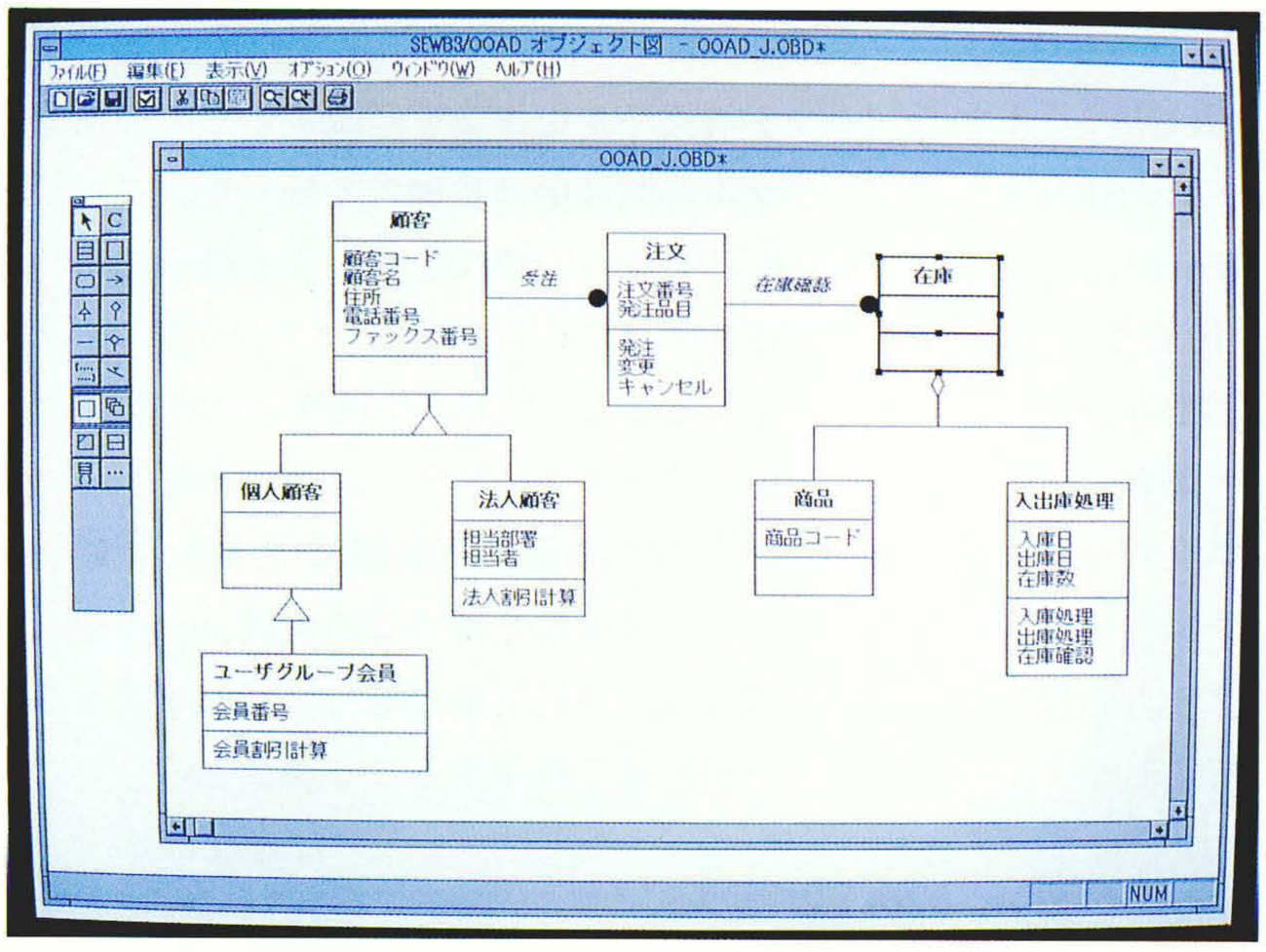


図1 SEWB3/OOADの画面例

「オブジェクト指向開発標準手順」で用いる仕様書のエディタ，プログラム生成ツールを備え，オブジェクト指向開発標準手順によるシステム開発を支援している。

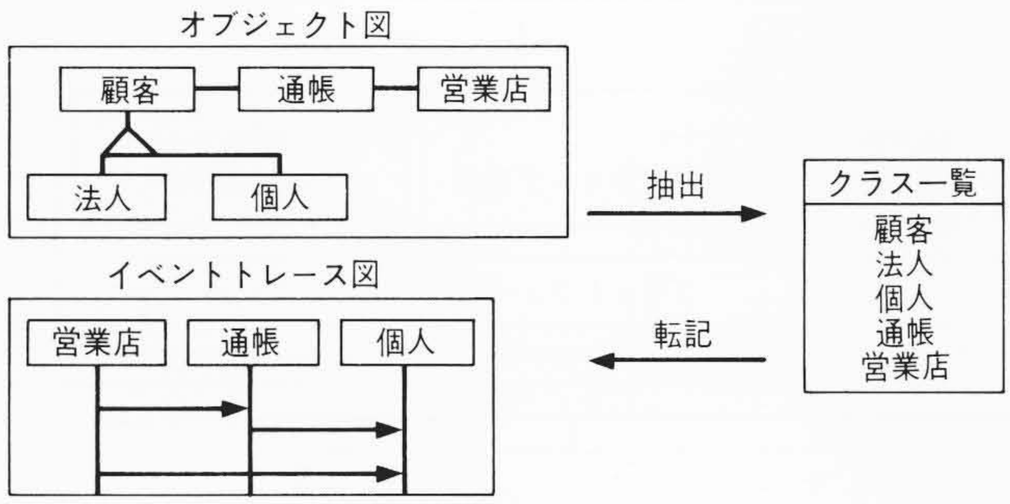


図2　仕様書間の設計情報の転記
すでに作成した仕様書の中の設計情報を転記することで，記述誤りを削減する。

る。これにより，クラス間の継承関係がループを形成するなどの誤りを早期に発見できる。

ある仕様書で記述されたクラス名や属性名などの設計情報が，別の仕様書にも記述されるので，仕様書は相互に密接に関連している。開発者は，何度も同じ設計情報を記述する必要があり，煩わしい。OOADは，設計情報の記述を容易にする機能を提供している。仕様書の作成時に開発者が記述項目を指定すると，すでに作成した仕様書の中の該当項目の一覧がメニュー表示されるので，メニュー選択によって簡単に記述できる(**図2**参照)。これにより，開発者は入力の手間が省けるとともに，ある仕様書に記述した設計情報を別の仕様書に誤りなく記述できる。

3　プログラミング工程への接続

効率よいシステム開発を実現するためには，分析・設計工程の成果物をプログラミング工程へスムーズに引き継ぐことが重要である。OOADは，分析・設計工程の成果物からC＋＋のプログラムを生成する機能を備え，コーディングの工数を削減している(**図3**参照)。

3.1　メソッド生成

オブジェクト指向のプログラムは，属性と属性にアクセスするメソッドをカプセル化することにより，クラスの持っているデータ構造を他のクラスから隠蔽(いんぺい)している。そのため，属性を個々に参照・更新するステップ数の少ない単純なメソッドが，属性の数だけ必要となる。

さらに，ビジネスシステムが扱う業務は，預金残高のチェックや給与の算出など，さまざまなチェックや計算に従って遂行される。このようなチェック・計算を行う式を，ここでは「業務ルール」と呼ぶ。このような業務ルールは，複数のシステムの間で共通利用することが多く，再利用性が高い。

OOADでは，プログラム生成の指示書であるテンプレートと業務ルールからプログラムを生成することを計画している。テンプレートには，生成するプログラムコードと，属性のタイプごとに生成するコードを変えるための条件分岐が記述してある。テンプレートから単純なメソッドを生成する。これにより，複数のクラスの各属性ごとに，同じ処理を繰り返しコーディングするような単純作業が自動化できる。

一方，業務ルールからデータチェックメソッドや属性値算出メソッドなどを生成する。これにより，さまざまなビジネスシステムの間で業務ルールが再利用できるようになる。

3.2　宣言部生成

オブジェクト図から，C＋＋のクラス宣言部とメソッド宣言部を生成する。オブジェクト図に記述した属性名やメソッド名が，生成するコードに反映される。

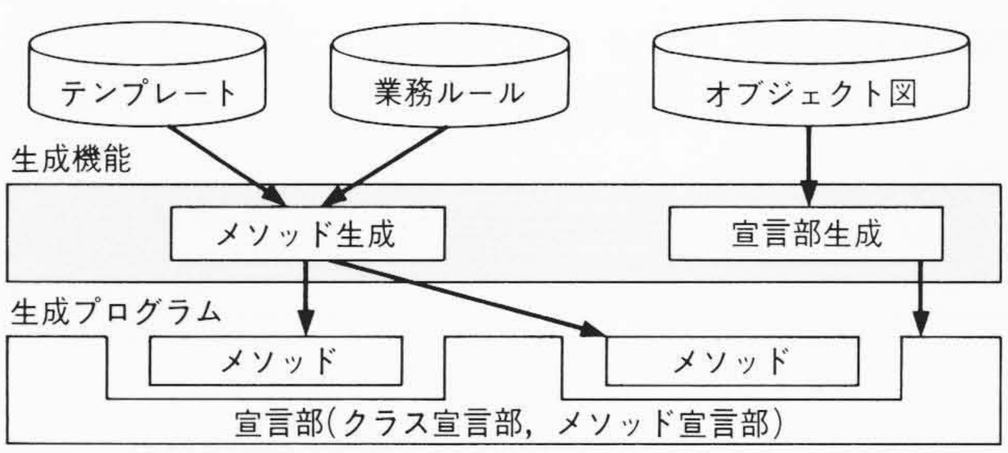


図3　OOADが提供する生成機能
設計情報から宣言部とメソッドを生成することにより，分析・設計工程とプログラミング工程の接続を実現している。

4　再利用の容易なシステムの構築支援

再利用を容易にするには，変更個所を特定することと，変更作業を軽減することが必要になる。OOADでは，これらに必要な情報をシステムの開発中に蓄積しておくため，再利用時にはそれらの情報が活用できる。

4.1　変更個所の特定を支援する機能

3章で述べた生成機能により，生成したプログラムと生成に使用した仕様書との間の依存関係を記憶する。依存関係を利用することにより，仕様を変更する際に，関連するプログラムが特定できる。

4.2　変更作業を軽減する機能

(1)　既存クラスを修正する機能

開発者がクラスの属性を変更する場合に，影響が及ぶクラス内の宣言部とメソッドを，OOADのプログラム生成機能を利用して再生成することが可能である。生成し

たプログラムに修正が施されている場合には，再生成したプログラムに，修正結果が自動的に反映される。これにより，仕様変更によって影響が及ぶ宣言部とメソッドが変更できる。

(2) 追加したクラスのメソッドを生成する機能

既存システムに新たなクラスを追加する際には，既存システムの処理方式に必要とされるメソッドを，追加するクラスにも備えなければならない場合がある。例えば，トランザクション処理や例外処理などを実現するシステムにクラスを追加する場合には，追加するクラスにこれらの処理を実現するメソッドを備える必要がある。しかし，処理方式を理解したうえで誤りなくメソッドをコーディングすることは，既存システムの開発者以外の者にとっては困難である。

OOADで開発したシステムは，3.1で述べたテンプレートを使用すれば，クラスを追加した際に，必要とされるメソッドを自動生成できる。これにより，処理方式の規則を詳細に知らない開発者でも，システムを誤りなく変更することができる。

5 クラスライブラリの活用

クラスライブラリを利用することにより，新規に開発すべきプログラムの量を削減することができる。クラスライブラリを基に開発システムで必要なクラスを新規に設計するためには，クラスライブラリが持っているクラス間の構造を知る必要がある。

OOADは，クラスライブラリのクラス宣言部からクラス間の構造を表現するオブジェクト図を生成できる。このオブジェクト図は，クラスライブラリ内のクラスを理解したり，検索したりする際に利用できる。また，オブジェクト図を生成することで，クラス名やクラスが持っているメソッド名を検索し，さまざまな仕様書へ転記できるため，クラスライブラリのマニュアルを検索する手間が省ける。

6 稼動環境とシステム構成

OOADはパソコン上で稼動し，作成した仕様書を，サーバに存在するリポジトリに格納する。オブジェクト指向環境のシステム構成を**図4**に示す。

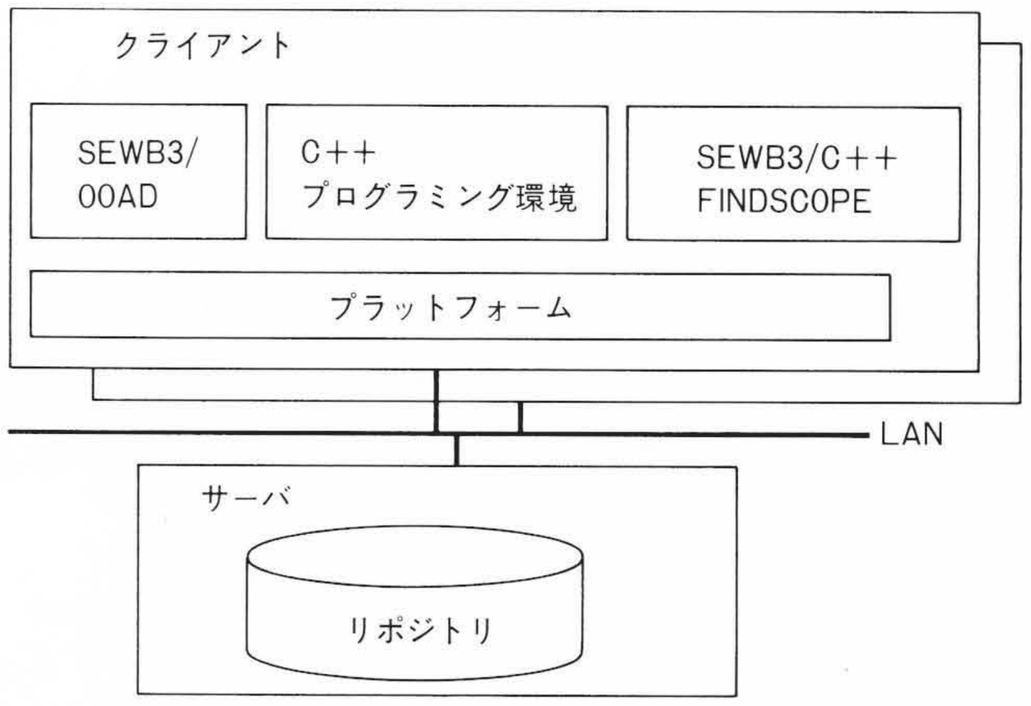


図4 オブジェクト指向システム開発環境の構成
パソコン上で作成した仕様書をサーバ上のリポジトリで管理する。

仕様書の作成，プログラムの生成などの開発作業は，開発者ごとに所有するパソコン上で行い，作成した仕様書・プログラムはサーバで共有できる。C++のプログラミング環境と合わせて使用することにより，生成したプログラムのコンパイル・テストが可能になる。さらに，SEWB3/C++FINDSCOPEを使用することにより，作成したC++のプログラムを解析して，クラス名やクラス階層を識別し蓄えることで，プログラムの検索，再利用が容易になる。

7 おわりに

ここでは，このたび開発したオブジェクト指向分析・設計支援ツール“SEWB3/OOAD”について述べた。SEWB3/OOADは，オブジェクト指向開発標準手順に基づく仕様書の作成を支援するエディタ，プログラム自動生成機能，再利用の容易なシステムを構築する機能，およびクラスライブラリの活用を支援する機能を備えている。

今後は，OOCOBOLのプログラム生成を可能にし，OOCOBOLのプログラミング環境[3]と接続するとともに，分散システムの開発支援機能を充実させていく考えである。

参考文献

1) 千葉，外：オブジェクト指向技術を用いたソフトウェアの開発技法，日立評論，**77**，12，839～842(平7-12)
2) J. Rumbaugh, et al.：Object-Oriented Modeling and Design, Prentice Hall(1991)
3) 西尾，外：プログラミング環境を充実させたオブジェクト指向COBOL，日立評論，**77**，12，855～858(平7-12)