ソフトウェア工学 122－6
（1999．3．18）

分散環境における再利用性の高いオブジェクト作成を支援する共同開発環境の研究

小池 誠†、並木 美太郎†、岩澤 京子‡
†-東京農工大学 工学部、‡-拓殖大学 工学部

本論文では、ソフトウェアを低コストかつ短期間に開発するために、ソフトウェア部品（オブジェクト）の再利用と共同開発による分散作業を行わせる方法を組み込んだ開発方法論を提示する。オブジェクトの再利用性を高めるためにオブジェクトの役割と再利用範囲に着目した分類法、共同開発のためにドキュメントの記述事項の提示を行う。そして、この開発方法論を利用できる共同開発環境Cafétériaを開発する。

Cafétériaの実装には、機種非依存性を持たせるために、オブジェクト指向言語であるJava言語を用いた。Cafétériaの開発を行った結果、再利用性の高いオブジェクトの蓄積、ドキュメントの形式化、オブジェクトの再利用による開発コストの削減という成果を得た。

Study on Collaborative Programming Environment to Develop High Reusable Objects on Distributed Environment

Makoto KOIKE †, Mitarou NAMIKI †, Kyoko IWASAWA ‡
†-Tokyo University of Agriculture and Technology, ‡-Takushoku University

This paper describes methodology of software development that can make the development in low cost and short term, by reusing software parts (objects) and by collaborating development. To increase reusability of objects, we use classification method that focuses on function and reusable scope of objects and also show documents for collaborative development. We develop collaborative programming environment called "Cafétéria" by using this methodology.

"Cafétéria" is implemented in Object Oriented Programming language "Java" for platform-neutral. We obtained results of accumulated high reusable objects, formed documents and reduced costs of development in "Cafétéria".

1. 緒言

近年、ソフトウェアはさまざまな機器に組み込まれており、その需要は増大する一方である。対してソフトウェアを開発するプログラマの数は不足しており、需要に対応するためには、プログラマの負担を増大させずに多くのソフトウェアを開発する方法が必要になる。また、技術の躍進が激しい分野のため、短期間に次々に新たなソフトウェアを開発する方法が必要である。

この方法として、ソフトウェア開発の労力を軽減するソフトウェア部品（オブジェクト）の再利用と分散作業による期間の短期化をはかれる共同開発がある。しかし、実際にこれらの方法を導入しても目標を達成された事例は少ない。原因としては、オブジェクトの再利用がなされていないこと、共同開発を行う時に必要な分担作業が分散することによる短期化よりも多くの作業を必要とされることである。

本研究では、この問題を改善した開発方法論の提示とこの方法論を組み込み、開発プロセスすべてを支援する共同開発環境 Cafétéria の開発を行い、評価を行う。

2. 従来の問題点

従来のオブジェクト指向での開発方法論にそった開発を行っても、オブジェクトの再利用性が活かされいてない原因は、開発プロセス中で再利用可能なオブジェクトを明確にしていないことに起因する。このため、オブジェクトの数が増加していくにつれ、再利用不可能なオブジェクトも検索の対象とすることによる検索の手間の増加、再利用可能なオブジェクトに特化した利用範囲などを示した詳細なドキュメントの作成が行えない問題が生ずる。

また、共同開発で分散による効果よりも、分担する手間の方が多い原因は、開発者間で利用するドキュメントに問題がある。このドキュメ

ントが開発に必要かつ十分なものだけでなければ、どのドキュメントを見れば次の作業を理解できるか分からず開発者は混乱をきたすことになる。

共同開発環境にも問題がある。従来の共同開発を支援する環境はユーザインタフェースの統合などを支援することを目的にし、開発プロセスすべてを支援している環境ではなかった。そのため、他のシステムを利用することにより整合性を保つことが困難になり、また、他のシステムの使用方法を学ぶ必要があり二重に手間がかかることになっていた。

3. 本研究の目標

本研究では、再利用性の高いオブジェクトの作成、蓄積と共同開発による分散作業の効率化を目的としたオブジェクト指向での開発方法論を提示する。そして、他のシステムを利用する必要がなく、上記方法論の開発プロセスすべての支援を提供する共同開発環境 Cafétéria の開発を行う。

4. 開発方法論

本研究で提示する開発方法論は従来の OMT 法や Booch 法などの構造化方法論をベースにしたオブジェクト指向方法論のアプローチの間違いをふまえ、オブジェクト指向言語専用の開発方法論を用いる（図 1）。本研究ではこの要求を満たすオブジェクト指向方法論 Drop[2]をベースにし、共同開発環境 Cafétéria に実装することを前提に改変した開発方法論を用いる。

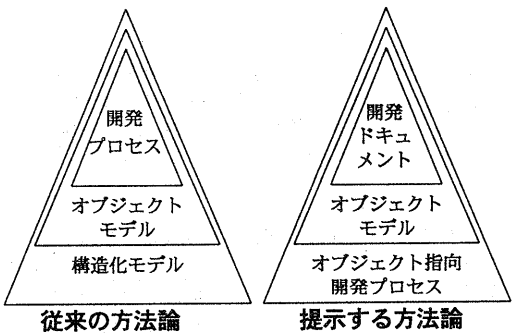

図 1. 方法論のアプローチの違い

本開発方法論では、オブジェクトの再利用性を高めるために、下記の二つの視点での分類を行う。

① 役割（データ処理、状態処理、表示）
② 再利用範囲（汎用、システム特化、アプリケーション特化）

この二つの視点、九つのカテゴリに分類することで、オブジェクトの構造的な欠陥の発見が行え、オブジェクトの構造を洗練したものにできるとともに、再利用可能なオブジェクトを分別できるようになる（図 2）。

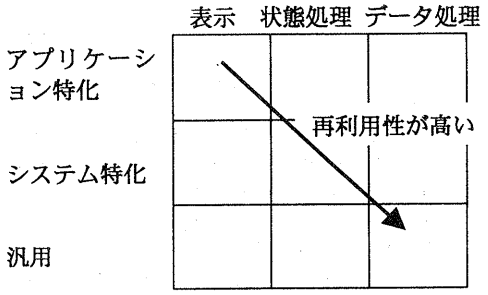

図 2. オブジェクトの分類

本方法論での開発者の構成は図 3 のように、一人の開発リーダと複数人の開発メンバから構成される。開発リーダが常時 Cafétéria をサーバとして起動した状態にしておき、開発メンバはクライアントとして接続する。

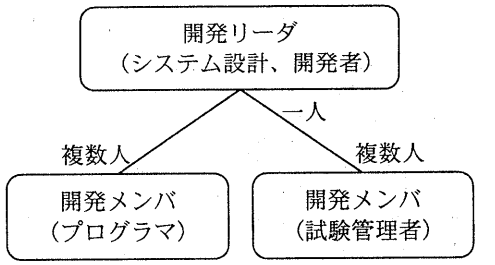

図 3. 開発者の構成

本方法論で提示する開発プロセスを表 1 のようにする。これらの開発プロセスをとおして、開発ドキュメントとオブジェクトを実装する。Cafétéria はクライアント・サーバ型のシステムであり、整合性を保つ方法として決定事項とされた開発ドキュメントはサーバ上で管理する。

共同開発に必要なドキュメントとしては、それぞれの開発プロセスに適合した開発ドキュメントを提示することにする。開発ドキュメントとは、開発中に作成されるすべてのドキュメントのことで、スケジュール、要求書、仕様書、オブジェクトのモデル図、オブジェクトのマニュアルなどが存在する。この中で、オブジェクトのモデル図の記述には業界標準のオブジェクト表記法 UML（Unified Modeling Language）[4]を採用する。

表 1. 開発プロセス

開発プロセス	作業内容（担当開発者）
開発プロジェクトの発足	プロジェクト名の決定（開発リーダ“L”） 開発者の登録（L）
スケジュールの決定	スケジュールの決定（L）
問題領域分析	問題領域分析（L） 開発ドキュメントの作成（Lまたは開発メンバ“M”）
システム共通設計	システム共通設計の作成（L） 開発ドキュメントの作成（LまたはM）
アプリケーション設計	開発メンバへの指示（L） アプリケーション設計案の作成（M） アプリケーション設計案の承認（L）
アプリケーション実装・試験	オブジェクトの分担（L） オブジェクトの実装（M） オブジェクトの試験（M）

5. Cafétéria の全体構成

Cafétéria では、再利用性の高いオブジェクトの明確化と共同開発用のドキュメントの提示、すべての開発プロセスを支援することを目的として、下記の機能を提供する。

(1) プロジェクトグループ管理機能

他のプロジェクトグループとの区別、開発者の認識、開発者以外からのアクセスを拒絶するための開発者を管理する機能。

(2) 開発プロセス管理機能

本研究で提示している開発方法論の開発プロセスを開発者に行わせるために、開発プロセスの表示とそれに対応した支援機能を開発者に提供する機能。

(3) スケジュール作成機能

開発の期限を守らせるためには、それぞれの開発プロセスにかかる期間を明確に示さなければならない。そのためのスケジュールを単純な操作で作成できる機能。

(4) 開発ドキュメント作成機能

開発中で作成しなければならない開発ドキュメントと記述事項を開発者に提示し、入力させる機能。

(5) オブジェクト抽出機能

オブジェクトの抽出を形式的に行わないと、各開発者が作成したオブジェクトの単位がまちまちになり再利用に向いていないオブジェクトが作成される。この問題を改善するためのオブジェクトの形式的な抽出を補助する機能。

(6) モデル図作成機能

抽出されたオブジェクトを構築する作業は、手直しや理解しやすい配置を行うことから、幾度もにわたる変更が必要であり、この作業を紙などで行うことを非効率的である。そこで、マウスオペレーションを基本にしたドローツールでオブジェクトモデルを作成できる機能。また、作成されたオブジェクトモデルを有効に利用するために、オブジェクトモデルから自動的にプログラムの基本形を生成する機能。

(7) オブジェクト分類機能

オブジェクトの再利用性を高めるためには、オブジェクトを役割と再利用範囲で分類することが必要である。この分類をマウスによるドラッグ＆ドロップにより簡単に行わせる機能。また、分類語のオブジェクトの開発ドキュメントを再利用範囲ごとに変更する機能。

(8) 分散オブジェクト管理機能

実装が完了したオブジェクトは各開発者のマシン上に格納する。そのために他の開発者の進行状況を把握し、作業の遅れなどを監視するための分散され格納されているオブジェクトの表示機能。

(9) 遠隔コンパイル機能

分散環境を指向した開発環境を想定しているため、低スペックの環境で実装を行うことも考えられる。その場合に、スペックの差異を軽減するために、開発中で一番処理能力を必要とするコンパイル作業を、高速なコンピュータ上で行わせる機能。

(10) 電子会議室

分散環境での共同開発では、遠隔地にいる開発者に指示を与えることや開発者から指示を受けることが必要である。そのための連絡手段としての受動的起動が可能な電子会議室。

Cafétéria の構成を図 4 に示す。すべての機能を統合的に管理するものとして、Cafétéria が存在する。その下に分散環境においてコンピュータを用いた共同開発専用の支援機能であるプロジェクトグループ管理機能、分散オブジェクト管理機能、遠隔コンパイル機能、電子会議室がある。プログラミングを支援する機能としてプログラミング支援機能がある。本研究で提示している開発方法論を提供するために開発方法論提供機能があり、その下に開発プロセスを管理する機能と支援機能を切り替える機能がある。

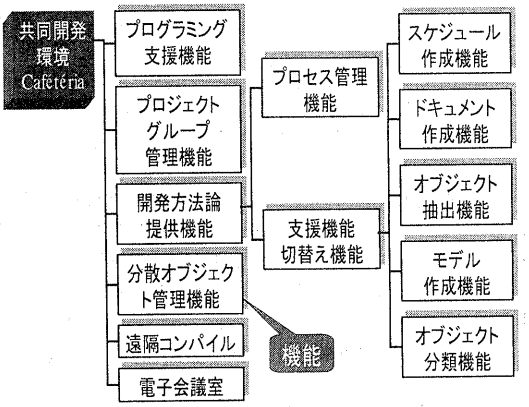

図 4. Cafétéria の全体構成

クライアントとサーバ間のデータの受け渡しには、分散オブジェクトを用いる。分散オブジェクトを用い、直接サーバ上のデータを処理しないことにより、クライアント側からの操作の誤りによるデータの削除や変更を不可能にし、開発ドキュメントの整合性を保つ（図 5）。

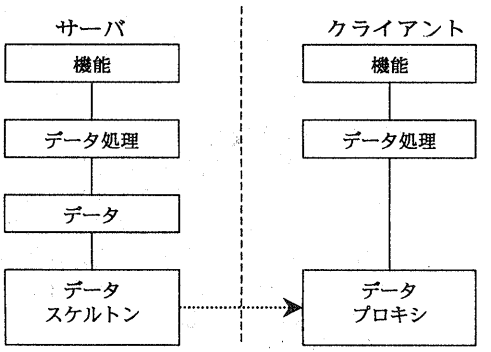

図 5. データの受け渡し

6. 各機能の設計

この章では Cafétéria の機能の中でも、特に特徴的な機能の設計について述べる。

6.1 プロジェクトグループ管理機能

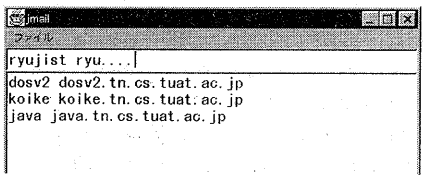

図 6. プロジェクトグループ管理機能

開発の最初に行うのがプロジェクト名の決定と参加する開発者の召集である。この前後で、顧客や上層部と交渉などを行う開発リーダを決定する。一つの開発プロジェクトでは一つのシステムまたはコンポーネントを開発する。プロジェクトグループへの開発メンバの追加や削除は開発リーダの権限で行い、図 16 の上部のフィールドで登録、下部のリストで削除が行える。ここで登録されたデータはすべての機能を使用する時に検査され、プロジェクトグループ以外からのアクセスを拒み、本システムのセキュリティを高める。

プロジェクトグループに登録される開発者はプロジェクトグループ管理用のオブジェクトに、開発者の名前（ユーザ ID など）と使用するマシン名（IP アドレスなど）の組を一つのオブジェクトとして追加される。

サーバは常に起動した状態にしておくことが前提であるが、万一サーバを立ち下げる時は、ローカルディスクにファイルとして保管できる（図 7）。

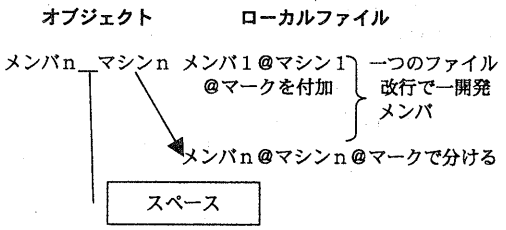

図 7. 開発者の格納方法

6.2 開発方法論提供機能の設計

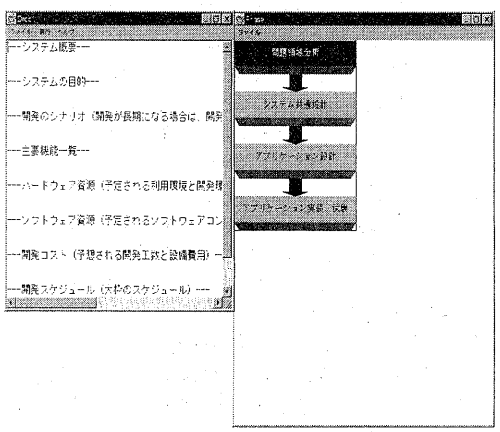

図 8. プロセス管理機能と支援機能

本機能は、開発者の登録が終了すると起動され、本研究で提示している開発プロセスにそった開発を開発者に行わせるためのものである。

現在の開発プロセスの内容をグラフィカルに表示し（図 8 右）、その開発プロセスでの作業を支援する機能を切り替えることで統一的な開発を行えるようにする。よって、開発プロセスに対応した機能の一覧情報の作成と他の機能を呼び出せる仕組みを開発する。支援機能切替えい機能では、開発ドキュメント作成機能、スケジュール作成機能、モデル図作成機能、オブジェクト分類機能の切り替えを行う。そのために図 9 のように、それぞれの支援機能の抽象オブジェクトとして支援機能アダプタオブジェクトを作成し、これを継承することで支援機能切替え機能からはどの支援機能を操作するのかを隠蔽する。開発プロセス管理機能から操作できる事柄は、開発ドキュメントのローカルファイルへの保存、開発ドキュメントを読み込み、そして、以後に記述する開発ドキュメントのための情報を生成するためのメソッドをすべて抽象メソッドとして定義し、支援機能を開発する時には、この抽象オブジェクトを継承し、必ずこのメソッドの実装部分を記述するように定義する。

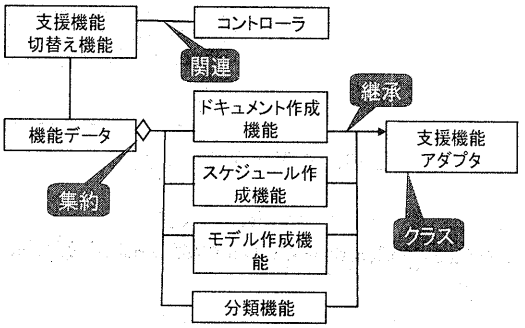

図 9. 支援機能の構造図

6.3 モデル図作成機能の設計

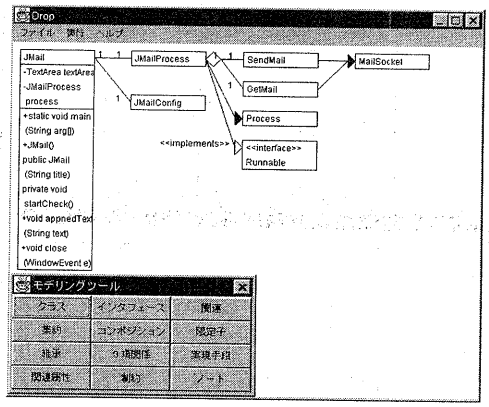

図 10. モデル図作成機能

モデル図の作成は開発ドキュメント作成の一部である。モデル図は文章だけで理解しにくい、オブジェクト間の関連や実行時の動的な振る舞いを表現するものであり、静的な構造を表すクラス図、動的な構造を表すオブジェクトメッセージ図、相互作用図があり、開発プロセスの項目ごとに使用できるツールが自動的に切り替わる用にする。

モデル図の表記に使用できるモデル要素を独立モデル要素と関連モデル要素とに分類する（表 1）。独立モデル要素は単体で表記でき、それ自体に意味のあるモデルである。関連モデル要素は独立モデル要素との関連によって表記されるモデルである。このように分類することによって、処理や表示を場合分けする。

表 2. モデル要素の分類

独立モデル要素	クラス、 インタフェース、 インスタンス、
関連モデル要素	関連、集約、継承、制約、 コンポジション、ノート、 限定子、三項関係、 インタフェース、 関連属性、リンク

操作は、マウスによるムーブ、クリック、ドラッグ、ドロップによる基本モデルの表記、移動、削除、キーボードによる基本モデルの編集とする。この繰り返しによってどのような複雑なモデル図も表現できる。

表示、移動などの処理は、モデル要素のオブジェクト単位で処理する。そのためにそれぞれのモデル要素を表現するオブジェクトを複数作成する。関連モデル要素には独立モデル要素そのものを格納することで、独立モデル要素の移動に対処することができる。モデルを格納するオブジェクトを図 11 に示す。すべてのモデル要素のスーパークラスとして抽象モデル要素オブジェクトを定義する。このオブジェクトに処理を行うオブジェクトから呼びだされるメソッドを定義することで処理側に実際に呼び出しているオブジェクトを意識させない構造にする。そして一番下のオブジェクトで個々にあった処理を記述（現在の座標、表示）することで特化する。

このような構造にすると生成するオブジェクトを一律に指定することができなくなる。そこで、オブジェクト名からそのオブジェクト名を持つオブジェクトをローカルディスクから検索し、発見できたら生成する。

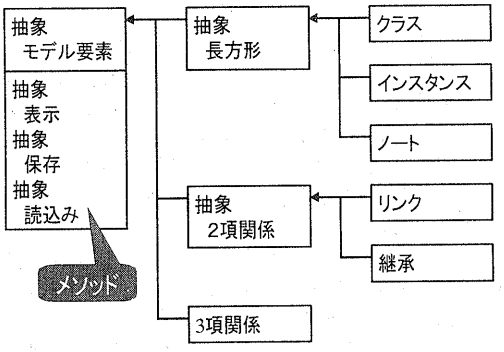

図 11. モデル要素の構造

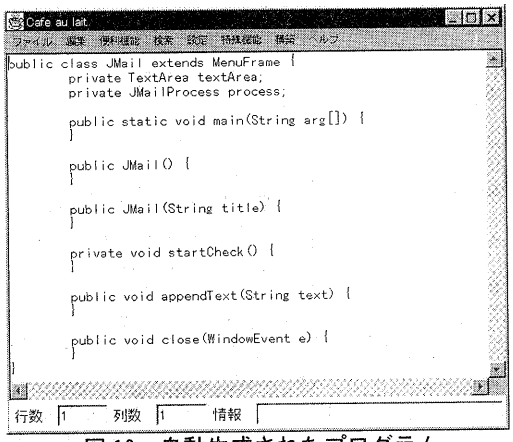

図 12. 自動生成されたプログラム

この機能では、作成されたモデル図から Java ソースプログラムの基本形を生成する。Java ソースプログラムの基本形を生成することにより、実装する際、メソッド名、引数、返値を間違うことがなくなり作業を効率的に行えるようになる。この機能は静的構造図だけで実行される。Java のソースプログラムの基本的な記述方法を独立モデル要素はオブジェクト自信が持っており、関連モデル要素では、独立モデル要素で作成された Java ソースプログラムに必要な記述を挿入することで行う。この動作では、独立モデル要素を初めにすべて実行し、その後に関連モデル要素の解析を行う。

図 13 のように左のようなクラスのモデルが記述されているとこの機能では右のような生成をする。この処理はまず独立モデル要素からプログラムの一番基本になるプログラムを生成する。これは、独立モデル要素に記述すべき内容（下線がない部分）がデータとしてある。その中に、クラス名、広域変数とメソッドをモデル図から受け取って挿入することによって基本形を生成する。その後に関連モデル要素がその独立モデル要素の中の特に継承、集約と実装手段が登録されている番号に指定されている場合、継承元のクラス、インタフェース名を受け取りクラス名の後に順次挿入し、完成されたプログラムの基本形がすべて出力される。

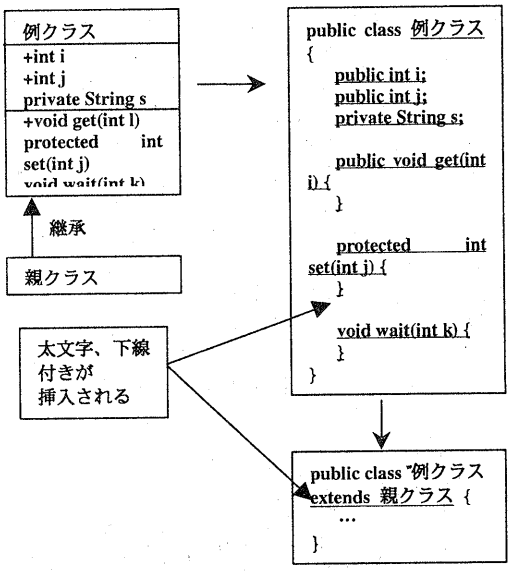

図 13. モデルからの変換例

6.4 オブジェクト分類機能の設計

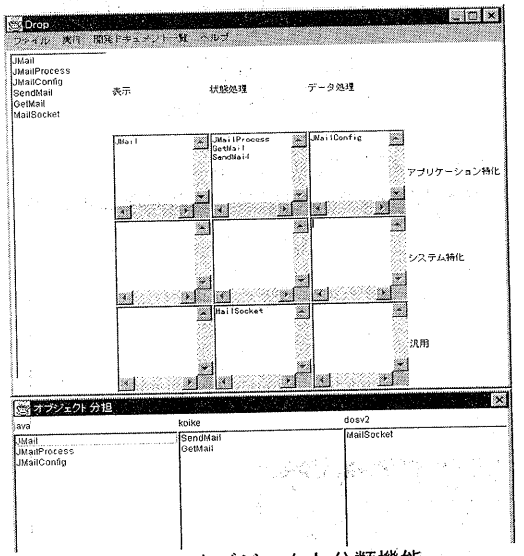

図 14. オブジェクト分類機能

図 14 はモデル図作成機能によって生成されたオブジェクトを九つのカテゴリに分類する機能

である。この分類によりオブジェクトがデータ処理、状態処理、表示の何に特化したものなのか、また、アプリケーション特化、システム特化、汎用なものなのかを再確認することができ、構造の練り直しや汎用化したオブジェクトの発見に役立てる。分類できないオブジェクトは構造的に間違ったオブジェクトとして構造を変える必要があることが発見できる。そして、分類結果から再利用可能なオブジェクトとして作成するオブジェクトを決定する。本機能では、オブジェクトモデルから生成されたオブジェクトの一覧の表示（図 14 上部ウィンドウ左）と九つのカテゴリ(図 14 上部ウィンドウ右)へのオブジェクトの移動を支援する。

分類されたオブジェクトをプロジェクトグループの開発者へ分担して、送信するための支援も行う（図 14 下部ウィンドウ)。本機能では開発者の表示とオブジェクトの分配の支援、オブジェクトのプログラムの基本形の自動配送を行う。

6.5 分散オブジェクト管理機能の設計

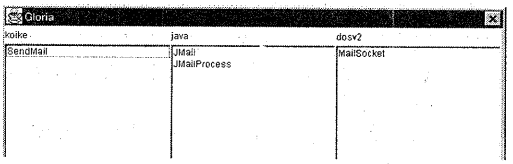

図 15. 分散オブジェクト管理機能

複数人で一つのシステムを開発する場合は、他の開発者の進行状況を知ることは重要である。特に開発リーダは開発者の進行状況を把握し、スケジュールにそった開発が行えるか考慮する必要がある。本機能はプロジェクトグループ中で作成されているオブジェクトの作成が完了した一覧を開発者単位で表示する。

本機能では、そのプロジェクトで作成が完了したオブジェクト（図 17）を検索し、一つのリストとして生成し、それを分散オブジェクトとすることにより他のマシン上から閲覧できるようにする。プロジェクトグループ管理機能を参照し、登録されているマシンから分散オブジェクトとしてオブジェクトのリストを入手する(図 16)。

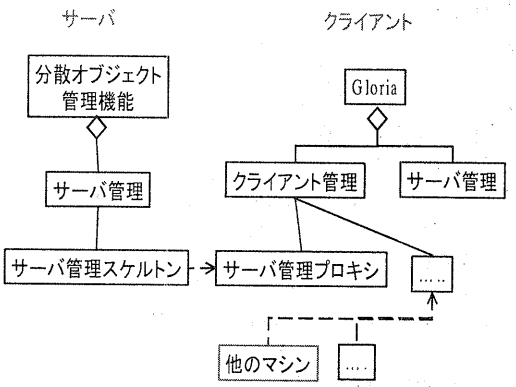

図 16. 分散オブジェクトの参照方法

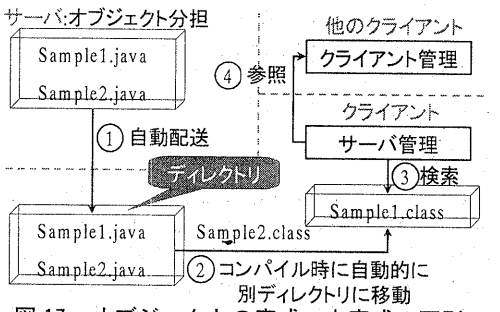

図 17. オブジェクトの完成、未完成の区別

本機能には、アプリケーション試験で他の開発者が作成したオブジェクトが必要な時に、オブジェクトの分散オブジェクト化、もしくは転送を行うことで利用できるようにする機能もある。任意のオブジェクトが選択されると、分散オブジェクトを自動生成する。この分散オブジェクトを利用しシステムを構築することで、分散システムの構築を容易にする。分散オブジェクト化の命令があると、指定されたオブジェクトのあるクライアントは、ローカルファイルからプログラムを読み込んで、分散環境のコンパイラをとおるように変換し、コンパイルを行う。ここで、コンパイラがエラーメッセージを返してきた場合は、命令を出したクライアントのマシンに変換前のプログラムを送信する。

7. 実装

Cafétéria の実装には、機種にとらわれず共同開発を行う環境を提供するために、Java を用いた。また、分散オブジェクトを実現する環境として分散環境 HORB を利用した。HORB は Java で記述されているため、機種非依存性であり、他の分散環境に比べ高速である。動作環境、実行を確認した環境を以下に示す。

動作環境

Java の動作が保証されている環境
JDK1.1 以上が必要
HORB1.0 以上が必要

実験に使用した環境

(1) サーバ
CPU：Sun UltraSPARK-ⅡiⅡ 270MHz
OS：Sun Solaris2.5.1
Java Version：JDK1.1.4

(2) クライアント 1
CPU：Intel PentiumⅡ 333MHz
OS：Microsoft Windows95
Java Version：Java2

(3) クライアント 2
CPU：Intel PentiumⅡ 300MHz
OS：Microsoft Windows95
Java Version：JDK1.1.7

サーバー台、クライアント二台を 10baseT の LAN で接続し、分散環境での実行を確認した。

8. 評価

Cafétéria の評価のために本研究で提示した開発方法論に乗っ取って Cafétéria の開発を行い、再利用可能なオブジェクトを蓄積した。完成した Cafétéria で評価のために、電子メールを自動受信し、設定文字数で分割、転送を行うシステム JMail を開発し、JMail のオブジェクトの構成と蓄積したオブジェクトの再利用率を評価した。

表 3. Cafétéria のオブジェクト数

作成したオブジェクトの種類	オブジェクト数
アプリケーション特化	193
再利用可能なオブジェクト	44
総オブジェクト数	237

表 4. JMail のオブジェクト数

作成したオブジェクトの種類	オブジェクト数
アプリケーション特化	6
再利用したオブジェクト	10
総オブジェクト数	16

上記表の結果が示すとおりに、Cafétéria の開発では、全体のオブジェクト中の 18%を再利用可能なオブジェクトとして発見し、蓄積した。その後、JMail の開発では、蓄積した再利用可能なオブジェクトの中から 23%を再利用でき、全体 62.5%のオブジェクトが再利用したオブジェクトが占め、開発コストを大幅に軽減できた。

9. 考察

本研究では蓄積した再利用可能なオブジェクトが存在しない状態での評価を行ったため、システムを開発するときに用いる基本的なオブジェクトが再利用可能なオブジェクトとして多数作成され、高い再利用率を得た。しかし、実際には今後の再利用可能なオブジェクトの発見と再利用率が問題になり、Cafétéria の有効性を完全に実証できたとは言い難い。さらに、再利用率を高めるためには、オブジェクト単位の再利用だけにとどまらずコンポーネント単位での再利用が必要になる。コンポーネント単位での再利用を行えるようにすることにより、同質システムの開発コストが大幅に削減できるようになる。

また、本研究では開発方法論と共同開発環境を分離したものとして考えずに、一つのものとして研究を進めた。そのことにより、従来の開発方法論や共同開発環境が生みだしていた矛盾を生成することなく実現できた。

10. 結言

本研究の成果は下記の通りである。

(1) オブジェクト指向開発での開発コストの軽減、開発期間の短期化を実現する開発方法論を提示した
(2) 分散環境上で動作する共同開発環境 Cafétéria を開発した

また、共同開発環境 Cafétéria の設計と実装、そして評価を行うことにより、下記の成果を得た。

(1) 再利用性の高いオブジェクトを蓄積した
(2) 開発ドキュメントを形式化した
(3) 再利用性の高いオブジェクトを再利用することにより開発コストを軽減した

参考文献

[1]萩本 順三、福村 真奈美、不破 康人 共著
“オブジェクト指向技術 応用実践 -Java によるビジネスアプリケーション開発モデルと実装技法-”
エーアイ出版株式会社 出版（1998）
[2]萩本 順三著
“オブジェクト指向方法論 Drop Version1.0”
http://www.njk.co.jp/otg/Drop/DropBook/（1996）
[3]萩本 順三著
“オブジェクト指向方法論 Drop Version2.0β”
http://www.njk.co.jp/otg/Drop/Drop_v20/（1998）
[4]日本ラショナル社
“UML ドキュメント ver1.1”
http://www.rational.co.jp/（1997）