ソフトウェア工学 87-8
(1992. 9. 29)

グループ開発のための オブジェクト指向開発支援システムの分析と設計

ペセモンテネグロ マヌエルヘスス　礒崎 賢一
九州工業大学 情報工学部

あらまし

オブジェクト指向プログラミングでは，ある一つのクラスに着目した場合に，その機能の定義が継承している複数の基底クラスに分散しており，機能の全体像を把握しにくいという問題がある．このため，分散している情報を統合して開発者に提示する支援機構が必要とされている．また，開発が大規模化するにしたがい，開発者相互の情報交換やコンパイル環境の構築などのグループ開発を支援する機能が必要になっている．本報告では，これらの要求分析に基づいて，ユーザインターフェースにブラウザを持ち，分散環境でのグループ開発支援機能を提供するプロジェクトサーバーを用いた，クライアントサーバーモデルによる開発支援システムの設計を示す．

和文キーワード：オブジェクト指向プログラミング，ソフトウェア分散開発，CASE，グループウェア，バージョン管理

Requirement Analysis and Design of Object-Oriented Development Support System for Group Development

Manuel Jesús PECE MONTENEGRO and Ken'ichi KAKIZAKI

Faculty of Computer Science and Systems Engineering
Kyushu Institute of Technology
680-4, Kawazu, Iizuka, Fukuoka 820, Japan

Abstract

In Object-Oriented programming, the definitions of the members in a certain class are distributed in various base classes, and it is difficult to grasp all of this information. We need to provide a facility that shows this information to the software developers. Moreover, for the large scale systems development, information exchange among development group members is necessary, as well as the construction of a compilation enviroment and other group development facilities. Based on this requirement analysis, we present the design of a Development Support System which uses the client-server model. It has a Class Browser as the user interface, and uses a project server to support the group development through the network.

英文 Key Words: Object-Oriented Programming, Software Distributed Development, CASE, Groupware, Version Management

1 はじめに

オブジェクト指向によるシステム開発では，対象のモデル化のしやすさ，モジュール性の高さ，再利用性の高さなどが大きな利点となり，ソフトウェア開発効率を向上させる有力な方式として期待されている．しかしながら，オブジェクト指向プログラミングには，継承や多態性などの従来のプログラミング言語にはないオブジェクト指向に特有の概念があるために，従来の開発手法に習熟した開発者が効率良くソフトウェアを開発する障害となっている．

また，システムの大規模化に伴い，ソフトウェアの開発作業は複数の技術者によるグループ開発が一般的になっている．グループ開発では，システムをいくつかの構成要素に分割して複数の開発者に割り当て，インターフェース部分の整合性を取りながら並行して開発を進めていく．このためには，メンバー間で緊密に情報交換を行ないながら開発を進めていく必要がある．しかしながら，ネットワーク機能の一般化により，場所的に分散した上で，異なるホストマシンで開発が進められるために，他のメンバーと相互に情報交換することが困難になっており，このような作業の支援システム [岡田 91] が必要となっている．

本報告では，オブジェクト指向プログラミング言語として多用されている C++ [Str91] を対象として，開発グループによるオブジェクト指向プログラミングの支援に必要な機能の要求分析と，実現方式の概要について示す．

2 要求分析

2.1 オブジェクト指向開発支援機能

オブジェクト指向プログラミングでは，使用可能なクラスの機能とインターフェースを把握し，それらのインスタンスを組み合わせて使用することと，適切なクラスを継承して新たなクラスを作成する作業が中心となる．したがって，システムを構成するすべてのクラスとそれらの間の関係を容易に把握するための支援機能が必要である．また，各クラスを利用するためには，それぞれのクラスが持つメソッドや変数を容易に調べることができる支援機能が必要である．これらの機能を備えた支援システムとして，Smalltalk-80 に実装されているブラウザ [Gol84] がある．Smalltalk-80 のブラウザは，オブジェクト指向プログラミングを支援するための環境として良く考慮されており，C++ を用いたオブジェクト指向プログラミングにおいても，C++ に特有な機能の支援機構を付加した上で，Smalltalk-80 と同様なブラウザをユーザインターフェースの中核に据える必要があると考えられる．

2.1.1 クラス階層の視覚化

クラスの全体像を直観的に把握できるようにするためには，クラス階層を視覚的に表現し，着目しているクラスの位置付が容易に理解できるようにする必要がある．C++ は多重継承を扱うことができるために，その階層関係を表現できるように，グラフ形式の表示を行なう必要がある．

C++ では，クラスの種類として，直接インスタンスを作成できる一般的なクラスと，派生クラスでしかインスタンスを作成できない抽象クラスがある．また，基底クラスの継承法としては，仮想基底クラスとしての継承やアクセスを制限した継承などの方式がある．このような，クラスの特性や継承の形式を容易に把握できるような情報の提示方式が必要である．

2.1.2 データメンバとメンバ関数の参照

オブジェクト指向言語では，継承機能を用いることにより各クラスに記述された機能を有機的に結合して利用でき，クラスを単位としたソフトウェアの部品化と再利用が可能になっている．しかしながら，各クラスで定義されている内容は基底クラスとの差分のみで，1つの機能の定義が複数の基底クラスの定義中に分散し断片化してしまっており，着目しているクラスの定義を参照しただけでは，機能とインターフェースの全容を把握することが困難になっている．また，そのクラスで定義されたものであっても，新たな定義が基底クラスですでに定義されているメンバ関数の上書きになる場合には，そのメンバ関数が仮想関数か否かがそのクラスの定義では判断することができず，基底クラスの定義に依存するという問題がある．

このような問題を解決するために，メンバ関数とデータメンバを参照する場合には，そのクラスで新たに定義された要素だけでなく，基底クラスから継承によって使用可能になるすべての要素を一度に参照できる必要がある．また，メンバ関数とデータメンバそれぞれで，機能とインターフェースの参照時に必要とされる属性が多数あるために，それらの属性が容易に把握できるようになっていなければならない．

データメンバとメンバ関数の情報を提示する際には，利用者の立場によって2種類の異なる要求がある．クラスの単なる利用者は，公開されたインターフェースのみが得られれば良い．一方，そのクラスを継承して新たなクラスを作成する利用者に対しては，クラスの内部で局所的に使用されている情報も含めて得られる必要がある．どちらの場合も，指定した情報が記述されているソースファイルを参照でき，必要性に応じて編集する機能が必要である．

C++ では，自分自身のメンバー関数の呼び出し形式と，通常の関数の呼び出し形式が同一であるために，メンバー関数の中での関数呼び出しが，どちらの呼び出しでどのファイルに定義が記述されているのか分かりにくい．したがって，メンバ関数ではない通常の関数も含めて，メンバ関数の名前からそのメンバ関数が定義されているクラスのリストを取得するための機能も必要とされる．C++ では，同一名の関数でも引数の型や個数が異なると別個の関数として認識されるために，引数の型や個数も含めてメンバ関数の定義を提示する機能が必要となる．また，オブジェクト内やオブジェクト間のメッセージの流れを把握するためには，メンバ関数の呼び出しのクロスリファレンスを提示する機能も必要であると考えられる．

2.1.3 クラス定義のための検査

クラス定義の支援機能として，クラス作成に問題が生じないように検査を行なう機能が必要であると考えられる．クラスの定義法は，以下の3種類に分類される．

1. 1つのクラスを継承する派生クラスを作成する
2. 複数のクラスを継承する派生クラスを作成する
3. 継承を用いずに全く新たなクラスを作成する

1，2の方法によって新たなクラスを作成する際には，意識的に上書きを行なう場合以外は，基底クラスと新しいクラスの間の定義の重複に注意する必要がある．特に2の場合には，複数の基底クラスで同一名のデータメンバやメンバ関数が定義されていると，新たなクラスを作成することはできるものの，重複しているメンバ関数やデータメンバの利用にさまざまな障害が生じるという問題がある．C++ の一般的なコンパイラでは，派生クラスでのメンバ関数やデータメンバの再定義は，意識的に行なっているものとして報告されず，基底クラス間で重複がある場合にも，派生クラスで参照の曖昧性が具体的に発生するまで何の警告も提示されないという問題がある．

多重継承によってクラスを作成した場合，クラスの継承関係が複雑になると，そのクラスを生成した場合にどのような順序でコンストラクタが呼ばれ，どのような順序で初期化の処理が行なわれるかが分かりにくくなる．このため，期待した初期化が行なわれないにも関わらず問題点に気がつくことができずに，障害の要因を除去できないという問題が生じる．このような問題が生じないようにするためには，オブジェクトの初期化，削除の順序がどのようになっているのかを提示する機能が必要である．

2.1.4 クラス定義の作成と変更の支援

新たなクラス定義を支援システムに加える方法としては，クラス定義を新規に作成する場合と，すでに作成されているクラス定義を追加する場合が考えられる．新規に作成する場合には，支援システムを利用して定義すればよいが，すでに作成されたクラスを追加する場合には，新しいファイルを支援システムに追加する機能が用意されていなくてはならない．また，新たなクラスを定義する作業では，ヘッダーファイル（クラスの宣言部）とメンバ関数の定義ファイル（クラスの実現部）を作成する．ファイルの最初の部分に記述する内容は定型であり，そのようなファイルの雛型を自動的に作成する機能があることが望ましい．

メンバ関数名や機能分割が不適切で，メンバ関数の名前や機能を変える必要が生じた場合には，システムの一貫性を維持するために，そのメンバ関数を用いているクラスの修正を行なう必要がある．このような修正を行なうために，メンバ関数の変更の影響を受けるクラスを検出し，変更者に提示する機能が必要になると考えられる．また，メンバ関数の変更により修正が必要になったクラスの開発者に対して，自動的に変更要請を伝達すると共に，変更内容を記録しておく機能が望まれる．

2.2 グループ開発支援機能

2.2.1 開発者間の情報交換

ワークステーションとネットワーク機能の普及により，ネットワークにより分散され，ホストマシンとファイルシステムが共有されない環境で並行して開発が行なわれるようになった．このため，技術者相互の間で，情報の交換や参照などを行なうことが困難になってきている．一方，オブジェクト指向によるプログラミングでは，従来のプログラミング言語によるモジュール分割を横の分割とみなすと，横の分割だけでなく，オブジェクト指向の継承機能を利用した具体化の段階が異なる縦に分割した構成要素を分担して開発することになる．このため，各分担部分を個別に開発する一方で，継承機能により他の開発者が開発したクラスを密接に使用するという状況が生じる．したがって，従来以上に他の技術者との情報交換 [古宮 91] [垂水 91] が必要になると考えられ，そのような情報交換を支援する機能が求められる．

明確に仕様を定め公開されているクラスは問題が少ないが，付加的に作成されるさまざまな支援用のクラスは，他の開発者に明示的に公開されることがないために，クラス名の重複などの問題が発生しやすいという問題がある．したがって，他の開発者が開発しているクラスの情報なども，自分が開発しているクラスの情報と同様に最新の定義による情報を取得できることが望まれる．

他の開発者の情報を参照するという受身の情報交換ができるだけでなく，自分がこれから行なうことの通知を行なう機能も必要である．あるクラス名や大域変数名を使用する場合，その名前を使用した定義を支援システムを通して確定するまでは，他の開発者が重複して定義に使用する可能性がある．このような問題を解決するためには，使用に先立ってクラス名や大域変数名を宣言しておき，他の開発者が重複して定義できなくする機能が必要である．

C++ に特有な内容だけではなく，C 言語でも問題となっていた大域的な定義に関してもクラス名などと同様に，プロジェクト内で一貫性を保つための機能が必要であると考えられる．このような要素としては，通常の関数，大域変数，データ型（構造体，typedef），マクロの定義などがある．

2.2.2 コンパイル環境の構築

ソフトウェアを開発する場合には，単にソースプログラムを記述するだけでなく，記述したプログラムのテストや，それらを統合したシステムの生成を行なえ [池田 91] なければならない．このためには，単に他の開発者のクラスの情報を参照できるだけでなく，コンパイルに必要なヘッダーファイル，ソースファイル，ライブラリ，オブジェクトコードなどを収集し，それらをコンパイル・リンクする環境の構築支援が行なえなければならない．

2.2.3 開発管理

グループ開発の支援機能としては，実際にプログラミングしている技術者間の情報交換の支援だけでなく，グループ開発を指揮している管理者が，開発の進捗状況を容易に把握できるための機能が必要であると考えられる．

2.3 操作性

ユーザインターフェースの中核として用いるブラウザは，マウスなどにより必要な情報を対話的かつ容易に取得できる操作性を備える必要があると考えられる．このためには，ブラウザ上に表示されている各種のリストやグラフの要素がマウスによる選択に適切な反応を示すようにしなければならない．

C++ では，同じ名前のメンバ関数でも，コンテクストにより呼び出される実体が異なるために，文法を理解したエディタなどによって対話的に支援する機能が望まれる．一方で，支援システムにエディタなどが組み込まれており，高い機能を実現していても，支援システムに組み込まれている機能だけではなく，一般的なツールの使用を望む技術者が多いと考えられる．特に，能力の高い技術者にはこのような傾向があり，また，それによる生産性の向上も大きいと考えられる．したがって，支援システムは，システムに埋め込まれたエディタなどのツールだけでなく，vi や emacs などの一般的なツールを使用してファイルを変更することも可能にする解放性を備えている必要がある．

3 支援方式と実現方式

3.1 ブラウザ

ユーザインターフェース部となるブラウザは，図 1 に示すように 5 個の要素で構成しており，ブラウザの最上部に指示を行なうためのボタンを配置している．その下でブラウザの中部にクラス名，メンバ関数，データメンバを提示する 3 組のリストを表示する．その下のブラウザの最下部に，グラフを用いてクラス階層を視覚的に表示する領域を作成する．

ブラウザのクラスリスト領域には，このプロジェクトで使用可能なクラスがアルファベット順に表示され，それらのクラスの階層情報はクラス階層グラフ領域に表示される．クラス階層グラフは，基底クラスを左側に表示する水平表示方式と，基底クラスを上側に表示する垂直表示方式を必要に応じて選択することができる．情報を得たいクラスは，クラス階層グラフやクラスリストの該当クラスをマウスでクリックすることによって選択することができる．クラスリストでクラスを選択した場合には，クラス階層グラフ上でそのクラスが強調表示され，クラス名からそのクラスの位置付が容易に把握できるように配慮している．

ブラウザ上でクラス，メンバ関数，データメンバなどを指定して，それぞれが定義されているファイルを参照，編集するウィンドウを開く機能を用意する．この機能によって，着目している対象の詳細を参照したり変更することが容易に行なえるようにしている．ウィンドウは複数開くことができ，複数のクラスやメンバ関数を相互に参照しながら効率良く作業を進めることができるように考慮している．Smalltalk-80 のブラウザとは異なり，基本的なブラウザとは別に編集用のウィンドウを開く方式を採ることにより，ブラウザを単純化すると共に，複数のファイルを同時に編集，参照対象にすることができるという利点がある．エディタとしては，X ウィンドウのウィジェットを用いて構築した，支援システムに組込みのエディットウィンドウだけでなく，vi や emacs などを選択可能にする．参照しているファイルが他の開発者によって更新された場合には，ブラウザ上にその通知が行なわれ，支援システムに組み込まれたエディタを使用している場合には，更新ボタンを押すだけで最新の定義内容を参照できる．他の編集ツールを使用している場合には，開発者自身によって最新の定義をエディタに再度読み込む処理を行なう．

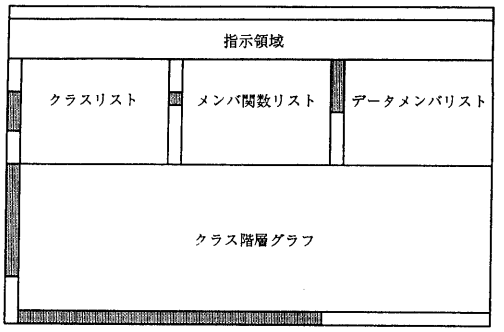

図 1: ブラウザの構成

3.2 プロジェクトの定義

グループによるソフトウェア開発を行なうためには，開発に参加するメンバーや開発に必要な資源の定義を行なう機能が必要である．プロジェクトを定義する際にプロジェクトサーバに登録する情報としては，以下のようなものがある．

1. プロジェクト名
2. 管理者
3. プロジェクトに参加しているホストマシン名
4. プロジェクトに参加している開発者
5. コンパイル用の環境の定義
6. 各開発者のホストマシン名
7. 各開発者の本プロジェクト用の作業用ディレクトリ

1～5はプロジェクトを管理する管理者が，6，7は各技術者が定義と変更を行なうことができる．また，管理者と開発者の登録情報としては，アカウントと実名，所属，連絡先などが記録される．これらの情報は，プロジェクト定義用のフロントエンドを利用して行なう．

プロジェクトの定義と管理を安全に行なうためには，管理者に特権を与え，管理者だけが重要な定義や操作を行なえるように制限しておく必要がある．プロジェクトの主要部分を定義変更する権限を持つ管理者は，特別なファイルに記述されている．また，各ホストのプロジェクトサーバーが，どのホストのプロジェクトサーバーのプロジェクト定義情報を受け入れるかも，制限を行なっておく必要がある．UNIX の場合には，これらの定義を /etc/project に記述する．

管理者の業務の支援機能としては，以下のような情報を提供する．

- 進捗管理

  作業の完了通知

  作業者毎の作業時間の積算

  各開発者のソースファイルの参照

- プログラムの統計情報の取得

  クラス数とメソッド数

  クラス定義とメソッド定義などの定義行数

  （プログラム行数，空行数，コメント行数など）

これらの情報は，開発効率を計る直接的な基準にはならないが，1つの目安として用いられることが多い．

3.3 オブジェクト指向開発支援

3.3.1 クラス階層グラフの作成

クラスが定義されているヘッダーファイルを解析することにより，各クラスの直接の基底クラスなどの基本情報が得られる．この情報を基にして，グラフ上での各クラスの表示位置を計算し，多重継承を含むクラス階層の表示を行なう．

C++ のクラスの種類をグラフ上に明示するために，各クラスの輪郭線として2種類の表現法を使用する．クラスが一般的なクラスの場合には通常の線でクラスの輪郭を描き，抽象クラスの場合には点線でクラスの輪郭を描く．これらの表現例を図2に示す．

図 2: クラスの特性の表現

継承の方法をグラフ上に明示するために，継承を示す直線に2種類の表現法を使用している．一般的な継承法の場合には通常の直線で表示し，仮想基底クラスとして継承する場合には点線で表示する表現法を取る．この表現例を図3に示す．この例では，Class1 と Class2 が ClassB を仮想基底クラスとして継承していることが示されている．

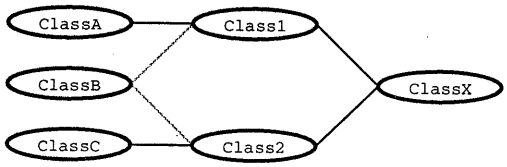

図 3: 継承表現

3.3.2 メンバ関数とデータメンバの情報

ブラウザでクラスが選択された場合には，基底クラスから継承されたものも含め，そのクラスのメンバ関数とデータメンバがブラウザ上に表示される．メンバ関数は，次のように関数名の前に4個のフィールドを持つ形式で表示される．

$\alpha\beta\gamma\delta$ 関数名 (引数の型 ...): 関数の型 [クラス]

α はアクセス指定を示し，以下の表示が行なわれる．無と示されている public では，α のフィールドには何も表示されない．

- 無 public
- + protected
- - private

β はスタティック指定の有無を示し，以下の様な表示が行なわれる．

- 無 通常のメンバ関数
- # スタティック指定されたメンバ関数

γ は仮想関数の指定を示し，以下の様な表示が行なわれる．

- 無 通常のメンバ関数
- ! 仮想メンバ関数
- ? 純粋仮想メンバ関数

δ はインライン展開の有無を示し，以下の様な表示が行なわれる．

- 無 通常のメンバ関数
- @ インラインメンバ関数

関数名の後には引数の型のリストが表示され，その後にコロンに引き続いて関数が返す値の型が表示される．この順番は，メンバ関数宣言の順序と逆であるが，メンバ関数を探す場合には，その名前で調べるのが一般的であるためにこのような表示順序にしている．また，最後の括弧内にそのメンバ関数が定義されているクラスが表示される．

メンバ関数とデータメンバの表示では，そのクラスの単純な利用者と，そのクラスを継承して新たなクラスを作成する開発者では，必要とする情報が異なる．このために，選択により public メンバだけというような指定した特性を持つメンバ関数やデータメンバだけを表示することができるようにしている．

クラスとメンバ関数を選択する方法として，クラスが主のモードとメンバ関数が主のモードの2種類を設ける．通常のメンバ関数の定義内容の参照では，対象となるクラスをまず選択し，そのクラスで使用可能なメンバ関数の定義を選択参照することが多い．しかしながら，メンバ関数の多態性に着目し，通常とは逆に，利用可能なクラスに定義されているすべてのメンバ関数のリストを表示し，そのリストから選択したメンバ関数を持つクラスのリストを表示する機能を用意する．この機能を用いることにより，あるメンバ関数がクラス毎にどのように定義されているかということを容易に調査することができる．

データメンバは，次のように変数名の前に2個のフィールドを持つ形式で表示される．

$\alpha\beta$ 変数名: 変数の型 [クラス]

α はアクセス指定，β はスタティック指定の有無を示し，メンバ関数と同様な表示が行なわれる．

3.4 クラス定義と更新の支援

新たなクラスは，ブラウザ上でマウスを使用することにより，対話的かつ容易に作成できるようにする．新たなクラスを定義する場合には，ブラウザ上で継承する1個以上の基底クラスを指定し，新たに作成するクラス名を入力する．入力したクラス名がすでに使用されている場合には，クラス名の変更要求が出力される．また，選択した基底クラスで重複するメンバ関数やデータメンバがある場合にはシステムは警告を出す．

選択された基底クラスで使用可能なすべてのメンバ関数は，ブラウザのメンバ関数リストに表示される．メンバ関数リストから新たなクラスで上書きするメンバ関数を選択する．また，新たに追加するメンバ関数とデータメンバの名前と仕様をキーボードから入力する．作成するクラスの定義と継承した基底クラスの定義との間に，データメンバの定義の重複があった場合にはシステムは警告を出す．また，新たに追加したメンバ関数が，すでにあるメンバ関数の上書きになる場合にはシステムは報告を表示する．

上記の操作によって，新たにクラスを定義するために必要な情報がそろうため，これらの情報を用いて新たなクラスの宣言部と実現部の雛型を作成する．クラス定義を行なうヘッダーファイルでは，基底クラスが指定されているため，それらのクラスを定義しているファイルをヘッダーファイルとしてインクルードする記述を加える．メンバ関数の定義を行なうクラスの実現部のファイルでは，クラスの定義を行なっているヘッダーファイルをインクルードするための記述と，各メンバ関数の雛型が作成される．さらに，それぞれのファイルの先頭部分に，ファイル名，プロジェクトの情報，著作権表示，バージョン管理情報などの定型記述が挿入される．

新たなクラス定義が作成できたり，従来のクラス定義を修正した場合には，それらのファイルが仕様を満たし，正常にコンパイルできることを開発者が確認した後に支援システムに通知する．

支援システムにクラス定義が更新されたことが通知されると，支援システムは，まずバージョン管理を行なうための情報がファイルに記述されているか否かを検査する．バージョン管理機構としては，RCS などを用いる．次に，プロジェクトに定義されているコンパイル用のコマンドライン（およびオプション）を用いて，コンパイルが正常に終了することを確認する．これらの検査が正常に行なわれたことが確認された後に，ファイルがバージョン管理機構にチェックインされる．また，チェックインを行なうとその情報が進捗管理情報として管理者に通知される．

ファイルの更新通知により，対象となるソースファイルを自動的にバージョン管理機構にチェックインすることにより，新たなバージョンに何らかの問題が生じた場合には，前のバージョンを取り出し再配布することができる．この方式では，グループによる開発が安全かつ容易に行なわれるという利点がある．

ある開発者が作成したクラスが改良され多くの開発者やプロジェクトで利用され，基本クラスとして認知された場合には，基本クラスに分類を変更する．

3.5 グループ開発の支援

3.5.1 クラス情報の参照

グループ開発における開発者別の3種類のクラスを一目で理解できるように，図4に示すように3種類を異なった形で表示する．基本クラスは，他のソフトウェアの開発などで作成され，汎用性があり有用なクラスとして基本的に利用されるクラスライブラリを示している．基本クラスのプロジェクトへの登録は，基本クラスを示す特別な開発者名で，ソースファイルが存在するディレクトリやファイルを指定する．この表示形式では，そのプロジェクトで使用，開発されているすべてのクラスの全体像を容易に把握できるだけでなく，自分が作成したクラスの派生クラスを他の開発者が作成している状況なども容易に確認することができ，変更を行なう時に注意を促すことができる．

各開発者は，自分のクラスのみ [Ell91] を変更すること

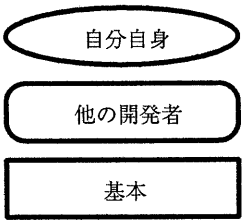

図 4: クラスの開発者別の表現

ができる．他の開発者の作成したクラス定義を参照することや，継承して新たなクラスで機能を上書きすることにより変更することはできるが，ソースファイルを直接変更することはできない．

自分が作成しているクラスの定義を変更し，その変更が確定した場合には，ブラウザのインターフェースを利用して支援システムにその通知を行なえるようにする．変更通知が行なわれると，追加変更されたファイルのチェックインがバージョン管理機能に行なわれると同時に，他のホストも含め他の開発者に提供する情報が更新される．支援システムにクラス変更が通知されるまでは，変更を受けた内容は他の開発者からは見ることができない．

3.5.2 コンパイル環境の構築

開発のために参照するクラスの情報などはその解析情報の交換でよいが，コンパイルとリンクを行なうためには，ヘッダファイル，ソースファイル，オブジェクトファイルなどのファイルの実体そのものを複写すると共に，各ホスト上の適切なディレクトリに設定しなければならない．

本支援システムでは，各ファイルの原本の存在場所がプロジェクトの定義情報に含まれているために，ファイルが更新された場合には，そのファイルをプロジェクトに参加しているホストに自動的に配布することができる．したがって，開発者は，どのファイルが原本であり，最新のバージョンがどのようになっているかということに注力せずにすむようになっている．

システムが完成しリリースする場合には，ソフトウェアの構成要素となるクラスなどのソースファイルのバージョンを記録しておき，そのソフトウェアを作成するために必要な環境を再構築する機能を持たせる．各ファイルのバージョンは RCS によって付加されたものを用いる．

4 システム構成の概要

4.1 サーバークライアントモデル

本支援システムは，ネットワーク上に分散して行なわれるグループ開発を支援するために，サーバークライアントモデルに基づいて実現され，システムは次の 2 つの構成要素からなる．

- ブラウザクライアント
- プロジェクトサーバー

ブラウザは，ユーザインターフェースを受け持ち，開発者の問い合わせをプロジェクトサーバーに伝達し，プロジェクトサーバーから取得した情報を視覚的に表示するための処理を行なう．

プロジェクトサーバーは，次のような機能を持つ．

- プログラムの解析
- ネットワークインターフェース機能
- 他のホストマシンへのサービス機能

図 5 に示すように，ブラウザは開発者 1 人ごとに起動され，プロジェクトサーバーは，グループ開発に関係しているホストマシンでそれぞれ 1 個づつ起動される．ブラウザの情報取得は，すべて各自のホストマシンのプロジェクトサーバーを経由して行なう．これにより，開発者が必要とする情報が他のホストマシン上に存在する場合には，サーバーがネットワークを経由して自動的にその情報の収集をすることで，開発者がネットワークの存在を強く意識せずに済むようにすることができる．

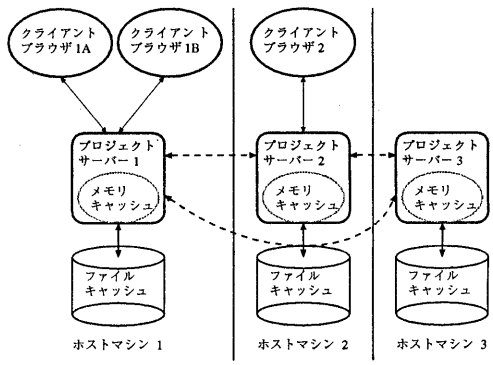

図 5: サーバーとクライアント

4.2 サーバーとクライアントの基本処理

ブラウザは開発者の要求にしたがい，自分のホストマシンのプロジェクトサーバーに対して，必要な情報を得るための問い合わせを行なう．プロジェクトサーバーは必要に応じて起動されるために，プロジェクトサーバーが立ち上がっていない場合には，その起動を行なう．

プロジェクトサーバーは，自分自身のホストマシンに該当する情報源があり，自分自身で分析して応えられる情報に関しては，必要な処理を行なった後にブラウザに対してただちに答を返す．他のホストマシンに問い合わせなければならない情報に関しては，該当するホストマシンのプ

ロジェクトサーバーへの問い合わせを行なう．プロジェクトサーバーは，同じホスト上のブラウザからの問い合わせに応答するだけではなく，他のホストのプロジェクトサーバーからの問い合わせにも応答し，そのホストで得られる解析情報を提供する．

そのホストのすべてのブラウザが終了し，他のホストからの問い合わせが一定時間なかった場合には処理を終了する．

4.3 情報収集とキャッシング

要求があるたびにソースファイルの解析やホスト間での転送を行なうと，処理時間が増大して応答時間が長くなると共に他の利用者に迷惑をかけたり，ネットワークのトラヒックが増大するなどの問題が生じる．また，支援システムが起動されるたびに，プロジェクトに関連するすべてのファイルが解析されると，システムの起動に多大な時間がかかってしまう．このために，繰り返し利用される情報は，キャッシュによって効率良く提供できるようにしておく必要がある．

プロジェクトサーバーは，ソースファイルやオブジェクトファイルが更新された場合には，その最新の内容をプロジェクトに参加しているすべてのホストに転送し，各ホスト毎に参照やコンパイルに利用できるように最新の環境を維持する．また，解析した情報をキャッシュとしてメモリ中に蓄えておき，他の開発者を含め2回目以降の問い合わせに対して効率良く応答できるようにする．また，終了する際にメモリ中の解析情報を処理をファイルに書き出しておき，プロジェクトサーバーが再起動された場合に，そのファイルの内容を読み込んで，多量のソースファイルを解析することなく，短時間で起動できるようにする．キャッシュの内容は，開発者単位で保存する．

情報をキャッシングし各ホストごとにファイルとして永続的に記録しておくことにより，性能上の利点があるだけでなく，他のホストがダウンしたり回線が切断されているなどの理由で，他のホストのプロジェクトサーバーと接続できない場合でも，キャッシュの内容によりサービスを継続させることができるという利点がある．

ファイルが変更された場合には，プロジェクトサーバーによって再度解析が行なわれる．クラス定義だけの変更であればその必要はないが，マクロ定義などの変更が含まれている可能性が高いために，変更が行なわれたファイルに依存しているファイルも再度解析を行なう．このために，ファイルを解析した時にその依存関係を記録しておく．ファイルの解析の結果，以前の解析内容に対して相違点が検出された場合には，その定義を他のプロジェクトサーバーにも伝達して，最新の定義を維持する．

5 まとめ

本報告では，グループによるオブジェクト指向開発の支援システムに要求される機能の要求分析と実現方式の概要を示した．

本報告の分析と設計に基づき，ワークステーション上でC++ を用いて支援システムの開発を行なっている．システムのほぼ全ての要素をオブジェクトとして定義しており，支援システムの機能が向上していくにしたがって自分自身の開発を支援し，本支援システムの開発途上で遭遇した問題点に対して，有効な支援を行なえるように拡張していきたいと考えている．システムの実現手法に関しては，反応時間とメモリ使用量などの評価に基づき，サーバーとクライアントの機能分担の設計やサーバーやクライアント間のプロトコル定義などを現在行なっている．

C++ の支援機能としては，我々が使用していないこともあり，フレンド関数，オペレータ，テンプレートなどに関する分析を行なっていない．また，開発時の対話的な支援だけでなく，文書化機能の提供などが，支援機能の向上として上げられる．

オブジェクト指向プログラミングによる開発効率の向上は，小さなソフトウェア部品のオブジェクト化ではなく，システムの骨格をオブジェクトの集合として提供する，アプリケーションフレームワークの構築によって実現されると考えている．したがって，問題の分析とオブジェクトの機能定義やクラスの階層関係を設計するツールを実現することと，アプリケーションフレームワークの構築と，その利用支援システムの実現を将来の目標としている．

参考文献

[Ell91] Ellis, C. A., Gibbs, S. J. and Rein, G. L.: "Groupware, some issues and experiences", *Communications of the ACM*, Vol. 34, No. 1, pp. 38–58 (1991).

[Gol84] Goldberg, A.: "Smalltalk-80, The Interactive Programming Environment", Addison Wesley (1984).

[Str91] Stroustrup, B.: "The C++ Programming Language, 2nd Ed.", Addison Wesley (1991).

[岡田 91] 岡田世志彦, 飯田元, 井上克郎, 鳥居 宏次永岡 渡, 梅本肇, 酒井睦：“ソフトウェアの分散開発のモデル化の試み”, ソフトウェア工学研究会資料 SE-82-1, 情報処理学会 (1991).

[古宮 91] 古宮誠一：“ソフトウェア協調型設計過程のモデル化と知的支援方法について”, ソフトウェア工学研究会資料 SE-83-15, 情報処理学会 (1991).

[垂水 91] 垂水浩幸：“ソフトウェア設計用グループウェアに関する一考察”, ソフトウェア工学研究会資料 SE-80-21, 情報処理学会 (1991).

[池田 91] 池田健次郎, 坪谷英昭, 岸知二：“共同開発時におけるインテグレーション支援機能”, ソフトウェア工学研究会資料 SE-79-3, 情報処理学会 (1991).