プログラミング 18－9
(1998. 3. 23)

Javaにおける分散開発環境の研究

小池 誠, 岩澤 京子

東京農工大学

〒184-0011 東京都小金井市東町1-8-21パークサイドヒル102

0423-86-3567

mkoike@cc.tuat.ac.jp

あらまし

Javaの統合開発環境において，分散開発環境を提供しているものは数少ない．また，オブジェクト指向におけるオブジェクトの再利用性，拡張性を取り入れている環境は希少である．そこで，この研究では，“Drop”オブジェクト指向方法論の考えを取り入れたJava用の分散開発環境を開発する．この開発環境を開発者が使用することによって，再利用オブジェクトの蓄積，アプリケーションソフトの効率的な開発が行えるようになる．

キーワード Java，開発環境，分散オブジェクト，オブジェクト指向，CASEツール

Study of Distributed Programming Environment for Java Language

Makoto KOIKE, Kyoko IWASAWA

Tokyo University of Agriculture and Technology

1-8-21-102 Higashi-cho, Koganei-shi, Tokyo 184-0011 Japan

0423-86-3567

mkoike@cc.tuat.ac.jp

Abstract

It is a small number that Integrated Development Environment (IDE) offers Distributed Development Environment to user. It is rare that IDE have notion of reusability and extensibility in object-oriented. So , this study presents Java based Distributed Programming Environment that accept idea of object-oriented methodology “Drop”. This users can do accumulation of reuse object and efficiency development of application.

key words Java, Development Environment, Distributed Object, Object-Oriented, CASE tool

1. はじめに

現在，成熟期を迎えているJava言語の統合開発環境でも，大型プロジェクト用に多人数の開発者が共同して開発できるものは少ない．これは，統一された認識を全開発者に与えられないからである．

さらに，開発コストを軽減するためのオブジェクトの再利用性／拡張性という概念は，誰もがその有効性を認識し，利用しようとするが実際には生かされていない．

そこで，以上の概念を練り込んだ開発環境が必要となる．これを，実現するために，以下に述べる方法を用いる．

現在，オブジェクト指向での表記法では，“UML（Unified Modeling Language）”[97-rational]が一般的である．そこで，“UML”を使用し，共同開発者全てに，同一オブジェクトの認識を持たせることで，多人数での共同使用に対処する．

また，オブジェクトの再利用性／拡張性について，もっとも体系的に捕らえてあるあるオブジェクト指向方法論“Drop”[97-hagimoto]を開発手法に取り入れる．

そして，共通オブジェクトを認識させ，変更案を全メンバーから操作できるようにするために，分散オブジェクト環境を利用する．中でも，操作性，機能性ともに優れている“HORB”[96-hirano]を利用する．

これらの，技術を利用し，機能，操作性ともに優れた分散開発環境Cafeteriaを設計し，開発したので報告する．

2. “Drop”について

萩本氏らが開発したオブジェクト指向方法論であり，現在までに使われていた方法論とは，設計思想の段階から異なる画期的な方法論である．この方法論では，オブジェクトの再利用基盤とアプリケーションを作成するプロジェクトの管理を明確化し，再利用オブジェクトの作成と短期間での製品開発を可能にした．

3. “HORB”について

平野氏が開発したJava専用の分散オブジェクト製品である。分散オブジェクト環境を提供する製品は、数多くあるが中でも、この“HORB”は初心者でも簡単に利用できる操作性の良さと、多彩な機能を持っている。

4. 設計方針

この開発環境Cafeteriaは“Drop”方法論を用いたJavaプログラム開発支援ツールである．ただし，“Drop”方法論で区別されているシステム開発チームと，再利用クラス開発チームの割りつけは，このツールでは使用する開発者の責任である．

このツールを用いての，Javaの再開発クラスおよびシステムの製作は，以下の手順によって行われるものとした．

① オブジェクト抽出のための，現実モデルの記述
② 要求モデルへの変換
③ アーキテクチャーモデルへの変換
④ プログラミング

この工程をシステムの作成のたびに繰り返していくことで，システムの作成と再利用クラスの蓄積が実現できるようにする．

5. システムの構成

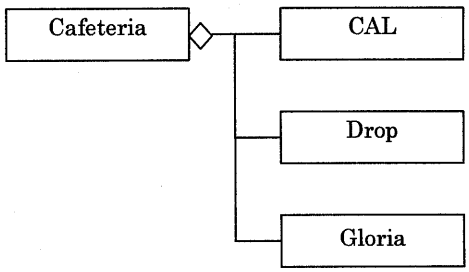


図1　Cafeteriaの構成

CafeteriaはCAL，Drop，Gloriaから構成されている分散開発環境である．CALはJavaプログラミング支援ツール（エディタ，クラスビューワ，クラス検索などが行える）であり，DropはUMLに乗っ取ったオブジェクトの表記ツールである．Gloriaは分散したオブジェクトを管理する機能がある．

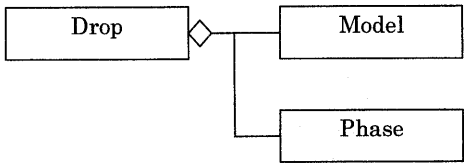


図2　Dropの構成

DropはModelとPhaseから構成される．ModelはUMLに乗っ取った構造モデルによって，オブジェクトの表現を記述するツールである．Phase

は“Drop”方法論に準じた開発フェーズを管理するツールである．この2つを使用することで，オブジェクトプログラミングの表記までがおこなえる．

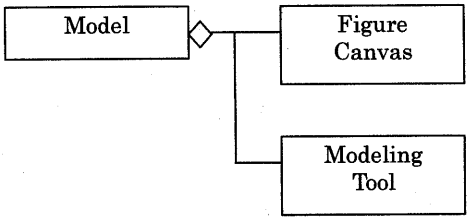
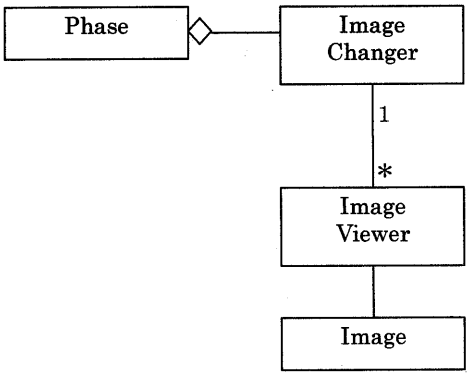


図３　Modelの構成

Modelはメインオブジェクトである．このオブジェクトは，FigureCanvasとModelingToolから構成される．FigureCanvasは現実モデルの記述や表記用のアイテムを配置するためのキャンバスであり，ModelingToolは表記用アイテムを選択するためのツールバックスである．これらは，Phaseとリンクしあい，その時のフェーズに適した構成になる．



１：１つを生成
＊：0個以上を生成

図４ Phaseの構成

Phaseはメインオブジェクトである．このオブジェクトは，ImageChangerからなる．ImageChangerはその時のフェーズに対応した画像にImageViewerを切り替えるためのオブジェクトで，ImageViewerは画像を読み込み表示するオブジェクトである．Imageは読み込まれる実際のフェーズの画像が格納されている．

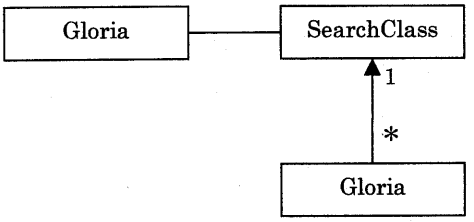
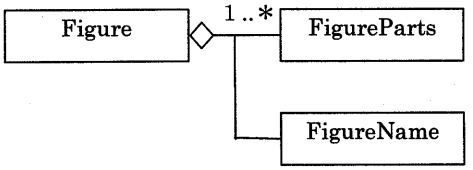


図5　Gloriaの構成

Gloriaは，SearchClassを生成する．SearchClassは，他のマシンのGloriaを分散オブジェクトとして，参照することによって，他のマシン上のオブジェクトを取得する．

6. データ構造



1..＊：1つ以上を生成

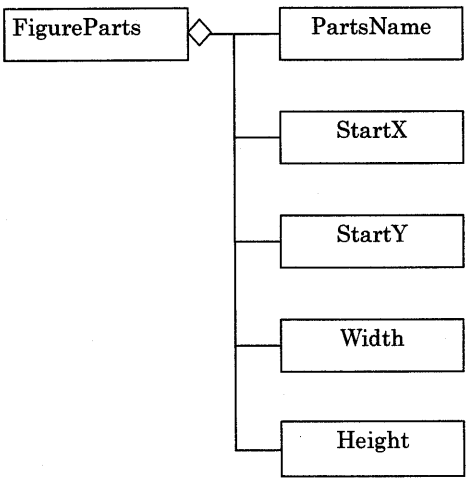


図６ Figureのデータ構造

Figureはそれぞれ表記アイテムとそれに付随する文字列からなっており，Figureのデータ構造は，図６のようになっている．1つのFigureに対して，そのFigureの名前と登録されている個数だけのFigureの基本図形（正方形，楕円，線，菱形など）からなるPartsから構成されている．FigurePartsは図形名，開始座標（ｘ，ｙ），幅，高さからなりどのような図形をどこに描画するかが格納されている．文字列を描画する状

況では，図形名に文字列の情報を格納することで拡張性を持たせる．

また，このオブジェクトは，CALからも参照でき，実際にプログラミングを行うときにすでに，エディタ中に記述されており，プログラミングの効率化とモデルに対するオブジェクトの正確さを保証している．

7. アルゴリズム

モデリングの操作は以下の手順のようになる．

① 表記用アイテムの選択
② 描画位置の決定
③ アイテムへの内容の記述

この繰り返しを行うことで，モデルを生成する．この操作は，以下の４つからなる．

- マウスの移動

表記用アイテムの位置決め，内容を記述するアイテムの選択に使用．

- マウスのクリック

表記用アイテムの位置の決定，内容を記述するアイテムの決定に使用

- マウスのドラッグ

描画したアイテムの移動

- キー入力

内容の記述

ここで，マウスのクリック（ドラッグ）を，マウスのダウンとリリース（と移動）という操作に分ける．そして，マウスがダウンされたとき，そこに既に表記用アイテムがあるかどうかを判定し，ない場合，表記用アイテムの描画，ある場合，内容の記述へと移るようにする．同様にドラッグによるアイテムの移動を行う．（図７）

また，ここで問題になるのは，③の内容の記述である．初期に与えられている表記用アイテムは，設定されている幅，高さになっている．この場合，記述する内容が表記用アイテムのサイズを超えてしまうと正確なモデルが生成できなくなる．そこで，ここでは，サイズを超える場合は，表記用のアイテムの高さを増加させて，段落を増やすことによって対処した（図８）．

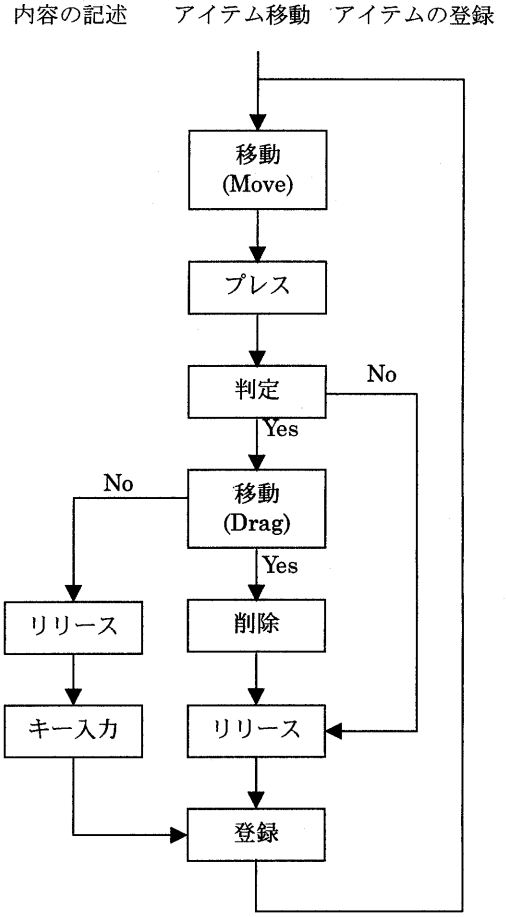


図7 モデリングのアルゴリズム

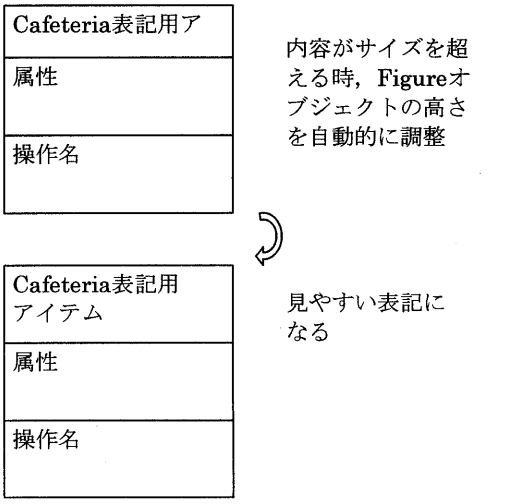


図８　表記用アイテムの調整

この時，Figureオブジェクトの内容を記述中のFigurePartsのHeight変数，下にあるオブジェクトのStartX変数を段落分増加させる．

8. 実装

8.1 モデル図の共有

この開発環境では，一人の使用から複数人の使用までを想定している．複数人で使用をする場合，“UML”で記述しているモデルを全ユーザに同一のものを提供する必要がある．そこで，使用人数に対応した構造を持っている．これを“HORB”を利用し実現している．

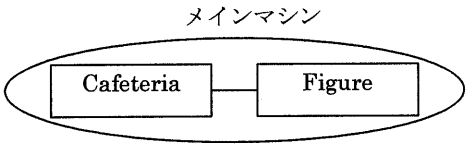


図9　シングルユーザの構造

シングルユーザの使用の場合は，そのマシンのディレクトリ中にあるオブジェクトから，単純にインスタンスを生成し，参照・編集ができるようになっている．

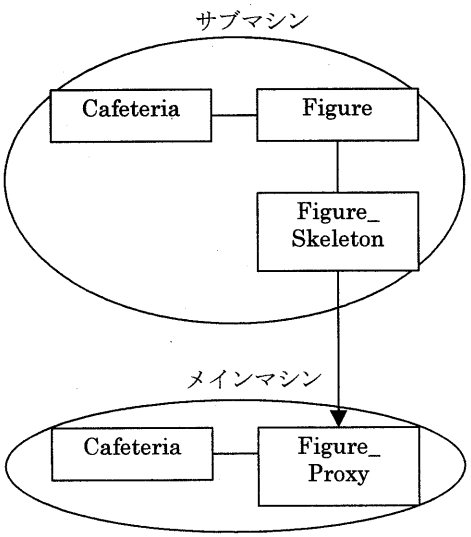


図10　マルチユーザの構造

それに対して，マルチユーザで使用する場合は，システム開発のリーダーを決定し，リーダーのマシン上にあるモデルのみを現在，採用されているモデルとする．

リーダーは自分のマシン上にあるモデルのオブジェクトをシングルユーザの時と同様に生成し使用する．

他の開発者は，リーダーのマシンのオブジェクトを分散オブジェクトとして使用する．この場合，参照のみが可能であり，編集を加えたモデルは，自分のマシン上のみに保存され，新規に採用されるまで，他のマシンからは，参照する事が出来ないようになる．

8.2 オブジェクトの相互利用

また，別マシン上にあるオブジェクトを相互に利用するために，それぞれのオブジェクトに対して，プロキシーとスケルトンの両方を生成し，図11のような構成をとる．

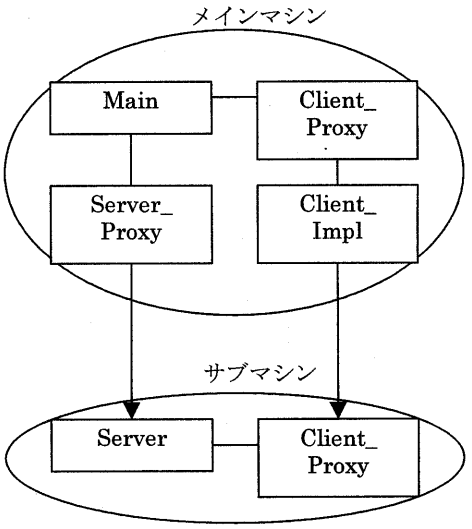


図11　相方向通信の構造

このようにすることによって，通常のクライアント／サーバシステムにみられる単方向の通信だけでなく，ユーザの操作に対する反応を返すことが可能になる．

9. 実行結果

この開発環境の実行結果を以下に示す.

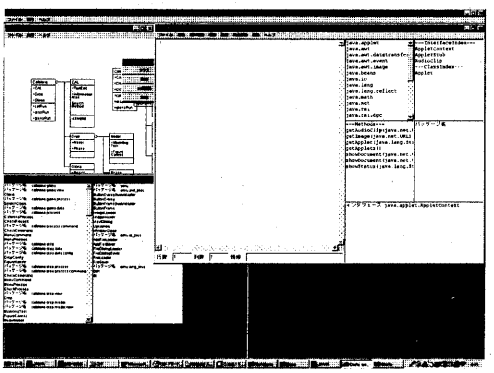


図12　全体画面

図12はCafeteriaを実行した実行画面である. 一番上にメインの操作メニュー. 右上にプログラミング支援環境, 左上にモデリングツール, 左下にオブジェクトの管理ツールが表示されている.

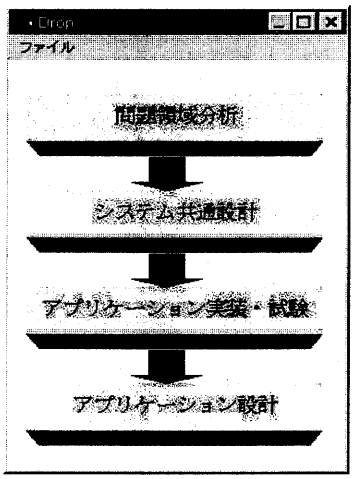


図13　Phaseの実行画面

図13は, 現在進行しているフェーズを示す画面である. この画面の進行に従っていくつかのモデリングをする手順が示される.

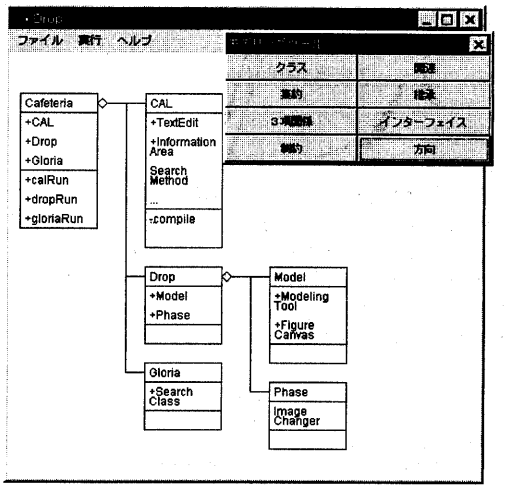


図14　Modelの実行画面

図14はモデルの作成画面である. 右上の小ウィンドウがモデリング用の表記用アイテムを選択するアイコンである. これは, 現在のフェーズに対応した表記用アイコンを表示するようになっており, ここで表記用アイテムを選択し, メインのウィンドウで操作を行うことでモデリングが行える.

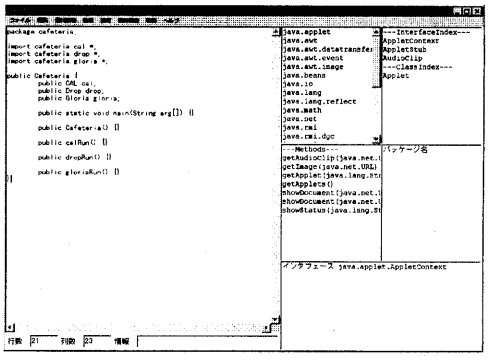


図15　CALの実行画面

図15は描画されたモデルをもとにプログラミング支援環境に自動的に出力されるプログラムの枠組みの一部である.

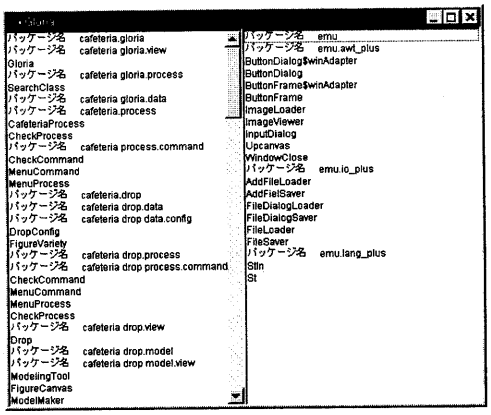


図16　Gloriaの実行画面

図16はオブジェクトの管理画面である．左側に，自分のマシン上で保持しているオブジェクトの一覧，右側に，他のマシン上にあるオブジェクトの一覧が表示されている．これは起動されているマシン数に対応して，表示画面が増加する仕組みになっている．

10. 考察

“HORB”を使用することによって，自分が起動したマシン以外にあるオブジェクトを使用することができる仕組みを行えるようになった．しかし，これには，制限があり，インスタンスを生成して使用することのみであり，継承のように，実際にインスタンスを生成しない利用の仕方では，他のマシン上のオブジェクトは，使用することができない．現時点では，この場合は，ネットワークを通して，ファイルをコピーすることで対応しているが，将来的には，ファイルをコピーせずに自由に行えるようにする必要がある．また，同様に，JavaのAPIやインタプリタの実装されていないマシン上でのプログラムの実行を行わせる拡張も考えている．

“Drop”とJavaプログラミング支援環境を統合することによって，オブジェクトの再利用性／拡張性を高めるだけでなく，プログラミングの効率化をも行えるようになった．“Drop”に基づいて，生成されたモデルを支援環境から参照し，プログラミングの編集を手助けすることで，それぞれ単体で用いる状況に比べ開発時間をかなり短縮する事ができる．

11.今後の課題

現時点では，分散開発環境の基盤が完成した．今後の課題として，企業での使用に耐えうるようなセキュリーティーの強化がある．企業単位での使用を想定して製作された環境なので，開発グループ以外からのアクセスは，完全に排除しなければならない．

また，バージョン管理を行うことも必要である．開発が進んでいくうちにバージョンは，あがっていくが，これを管理し，有効に利用できるようにしておくことが，将来的な利用を見越したアプリケーションには必要不可欠である．

今後の研究としては，コンピュータごとの負荷分散をはかり，コンパイル・実行速度の高速化を行う．

また，現在のプログラミングにおけるキーワードともいえるデータベース，分散オブジェクトなどを簡略化して利用できる機能の研究も必要である．

12. おわりに

“Drop”と“HORB”の洗練された技巧のおかげで，オブジェクト指向の再利用性／拡張性を高めることができるJava用の分散支援環境を制作することができた．現時点では，機能的にまだ不十分であるが，先に述べたような機能やまた新しい概念などを追加していくことで，常に新しく，しかし技術の動向に左右されることのない開発環境としていくことが大切である．

参考文献

[98-hagimoto]萩本順三・福村真奈美・不破康人共著，“最新オブジェクト指向技術応用実践”，エーアイ出版株式会社
[96-hagimoto] 萩本順三著，“Drop”，http://www.njk.co.jp/otg/Drop/DropBook/
[96-hirano] 平野聡，“HORB”，http://www.njk.co.jp/openlab/horb/
[97-rational]日本ラショナル社“UMLドキュメント”，http://www.rational.co.jp/