ソフトウェア工学 97－7
（1994．3．1）

分散開発を支援するオブジェクト指向アーキテクチャの一提案

堀 雅和　　落水 浩一郎
(株) インテック・システム研究所　北陸先端科学技術大学院大学

本稿では、分散開発環境を形式化するための一つの基本的要件として、現実世界の仕事遂行における2つの主要な協調活動を明らかにし、分散開発環境を支援するためのオブジェクト指向アーキテクチャの提案を行う。

今回提案するオブジェクト指向アーキテクチャでは、作業を行う作業空間に加え、グループオブジェクトと作業情報オブジェクトの概念を導入した。そしてこのアーキテクチャを自己反映計算の機能を持つ並列オブジェクト指向言語 ABCL/R2 で実装し、その有効性を調べた。

A Proposal of an Object-Oriented Architecture to Support a Distributed Development

Masakazu Hori
INTEC Systems Laboratory, Inc.

Koichiro Ochimizu
Japan Advanced Institute of Science and Technology, Hokuriku.

In this paper, we make clear one of basic points to formalize an environment for a distributed development. For this purpose, we analyse two main cooperative activities to carry out a job, which we come across in a real world, and we propose an object-oriented architecture to support such activities under an environment for a distributed development.

In the object-oriented architecture which we propose, we introduce the concepts of a workspace, a group object, and a work information object. We implement this architecture by ABCL/R2, which is an object-oriented concurrent reflective programming language, and we observe the effectiveness of the architecture.

1 はじめに

1.1 本研究の目的

本研究では、分散開発環境を形式化するための一つの基本的要件を明らかにし、分散開発環境を支援するためのオブジェクト指向アーキテクチャの提案を行う。

この研究の目標は、分散開発支援のツール層とOSの間の中間層をグループオブジェクトで形式化することにより、種々の応用プログラムに対するインフラストラクチャを単一の枠組で形式化することである。この中間層は、図1に記されている Vela 第2版のアーキテクチャ[2]のうち、

- 分散開発支援グループオブジェクト群
- CSCSD インフラストラクチャ
- 分散開発環境

の部分にあたる。この中間層の機能は、下層に対しては、分散透過性[9]を保証し上層に対しては、アプリケーションに依存しない仮想性を与える。

本稿では、上記のモデルの各層が持つべき機能について、とくに、次節で述べる分散開発における基本的要件を例にとり、それを満足させる立場から考察する。

1.2 分散開発支援の一基本的要件

近年プログラム開発を行うに際し、単一グループだけでなく複数サイトに分散した別組織によって仕事を分担し、共同開発を行う事例(図2参照)が増加してきている。そのような形態でプログラム開発を行う場合、一般的に開発の主管となる組織が他の組織へ要求仕様書やテスト仕様書といった書面により、仕事を外注に出す。外注先では、渡された仕様書をもとに開発やテストを行い、その成果物を納品する。その間のプログラムの分散開発を支援するツールといえば、マルチメディアリアルタイム会議システムをはじめとして、電話や電子メールなどのコミュニケーションツールがほとんどであり、分散開発を支援するための有効なモデル、システムはあまりみられない。

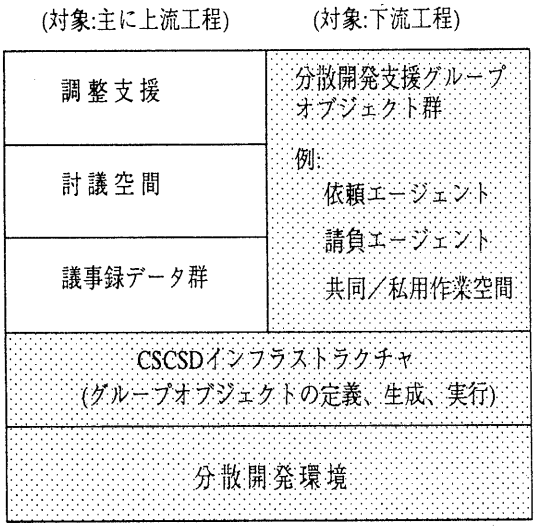

図 1: Vela 第2版のアーキテクチャの構想

以上のような現実問題を、分散開発環境を形式化する立場から一般化してみると以下のような単純な協調関係に帰着することができる。

「依頼元 A が、達成すべき仕事 J を (J_1, J_2) に分割し、J_2 の部分を請負先 B に依頼し、結果を受け取る。このあと、自身でおこなった仕事の J_1 と統合することにより、依頼元 A の仕事が達成される。」

このように単純な協調関係であるにもかかわらず、

1. 請負先 B に仕事を依頼する。
2. 請負先 B から結果を受け取る。

という2つの主要な協調活動は、分散開発を支援する環境の実現にあたって、以下に述べるように形式化すべき重要なセマンティクスを持つ(図 3参照)。

図 3において(1)に関しては、次のような現実問題がある。開発の主管組織は、その組織が有するプロジェクト・データベースから各外注先に

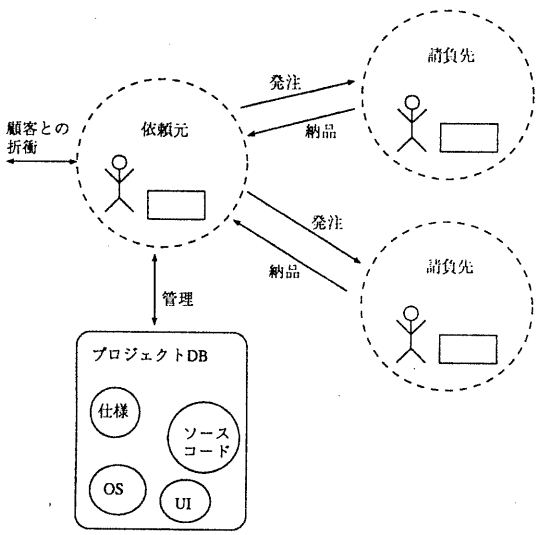

図 2: 分散開発の形態

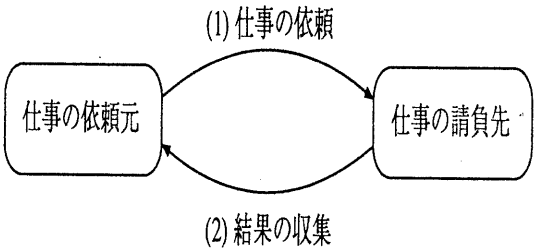

図 3: 分散開発における 2 つの主要な協調活動

向け、発注する仕様のグループを切り出し、外注先に送る。外注先では、その仕様に基づき開発を行うが、その際にその外注先の人達は、自分達が開発に必要とする情報を、そのプロジェクト・データベースから得る必要がある。この時、

- 外注先に渡される情報は、依頼された仕事を達成するために必要十分であるとともに、
- 依頼された仕事を達成するために必要な作業環境も同時に設定される必要がある。

また (2) に関しては、以下のような問題がある。依頼された仕事は、成果物をプロジェクト・データベースに登録することで終了する。この時、

- インタフェースの整合性などのプロジェクト・データベースで定義されている全体制約を納品された成果物が満しているかどうか自動的にチェックする。

もし全体制約を満していない場合、データベースで解決できる場合は自動的に解決し、できない場合は不整合が生じた箇所を指摘できる必要がある。

本論文の構成は、次の通りである。まず第 2 章において、「作業環境の設定」に関して、作業空間の概念を導入、定義する。ここで作業空間とは、本アーキテクチャが利用者に与える最上位の仮想性である。第 3 章では、作業空間を実現、管理する役割を果すグループオブジェクトの概念を導入する。第 4 章では、ABCL/R2 を用いて今回提案するモデルを実装し、分散開発における各個人の作業環境がどの程度実現できるのか検証した結果について述べる。第 5 章で、成果と今後の課題をまとめる。

2 作業空間

2.1 作業空間の概念の定義

複数人による分散開発において、各人に必要十分な仕事の環境をネットワーク透過に与えることは重要である。この節では、作業空間の概念の定義を行う。ここで、図 2における依頼元と請負先を特に区別する必要のない場合は、構成員と呼ぶことにする。

構成員は、名前と自己の作業空間を持つ。構成員が直接アクセスできるのは、自分の持つ作業空間内の情報だけであり、この情報をもとに仕事を行う。つまり構成員は、次のような 2 項組により構成されている。

$$AgentA_i = (Name_i, Workspace_i)$$

$Name_i$: 構成員 A_iの名前
$Workspace_i$: 構成員 A_iの作業空間

1. 作業空間

   作業空間は、仕事に関する情報の集合で構成されている。この作業空間の構成要素

は、依頼する仕事の内容あるいは各フェーズにおいて変化する。また各作業空間の関係も変化する。

2. 仕事

仕事はその内容が仕様で規定されており、仕事を行うとはその仕様に基づきさまざまな成果物 (例:ソースコード、ドキュメント、実行ファイル等) を生成することを指す。

2.2 分散開発における協調活動モデル

上記で定義した作業空間の概念を用いて、図 3 の協調活動をモデル化する (図 4)。ここで、構成員の機能がシステム内部の機能として実現されている場合には、それを代行者と呼ぶことにする。図 4においては、依頼代行者と請負代行者の 2 つの代行者が導入されている。

図 4において実線の楕円で囲まれた領域は、各代行者の作業空間を表す。またこの図で各代行者の作業空間がオーバラップした領域は、お互いの共有作業空間となる事を示している。依頼代行者が請負代行者に仕事を依頼する時に、依頼した仕事以外に仕事を行う上で必要な情報を共有作業空間に入れることで、請負代行者の作業空間 (仕事の環境) を設定する。この時、仕事の内容に応じたデータの見せ方ができることが必要となる。

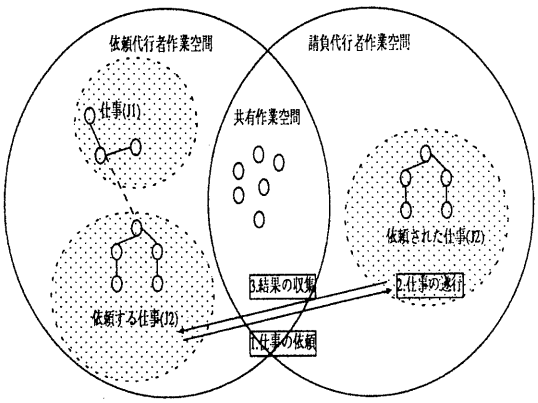

図 4: 分散開発における協調活動モデル

3 グループオブジェクトによる協調活動の実現と作業空間の管理

3.1 オブジェクトモデルの基本的要件

2 章で示した分散開発における協調活動モデルを、オブジェクトにより実現する際、これをグループオブジェクトの概念を導入して形式化する。

3.1.1 グループオブジェクトによる作業空間の管理

グループオブジェクトは代行者に対応し、作業空間の管理を行う。作業空間は、作業情報オブジェクトの集合により構成されている。グループオブジェクトの役割は具体的には、

- 作業空間に属する作業情報オブジェクトの管理
- 作業空間に共通なメソッド (環境、協調作業支援) の管理
- 作業情報オブジェクトの作業空間への参加/退出の管理

などを行う。

3.1.2 作業情報オブジェクト

作業情報オブジェクトは、各グループオブジェクトが持つ作業空間を構成する情報である。作業情報オブジェクトは、

- エントリオブジェクト
- 任意個の成果物オブジェクト

により構成された複合オブジェクトである (図 5)。

1. エントリオブジェクト

エントリオブジェクトは、作業情報オブジェクトの入口に位置し、作業情報オブジェクトへのアクセスを最初に受ける。コンパイルのように複数の成果物オブジェクトにアクセスして実行するような複雑な処理の場合は、エントリオブジェクトがその

処理を代行する。しかし、成果物オブジェクトへの参照や編集などのような処理要求の場合は、受け取ったメッセージを、透過的に成果物オブジェクトに送る。

2. 成果物オブジェクト

成果物オブジェクトは、ひとつの仕事を実現していく上で発生する各種中間成果物(例:仕様書、ソースコード、実行ファイル)ひとつひとつに対応するオブジェクトである。各成果物オブジェクトは、必ずしも同じメソッドを持つ必要がなく、各オブジェクトに独自のものを持つことができる。

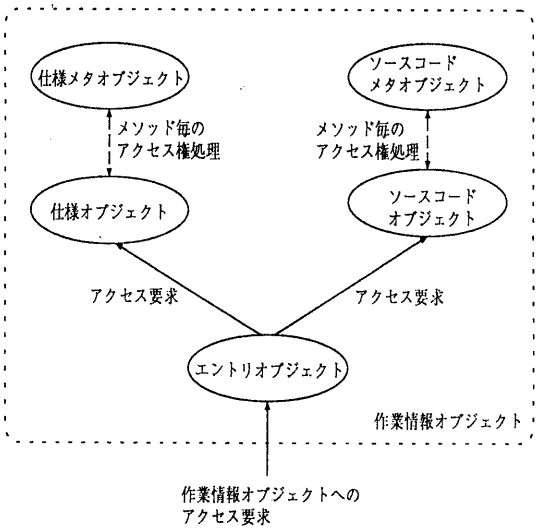

図 5: 作業情報オブジェクトの構成

依頼代行者は、依頼する仕事の情報として作業情報オブジェクトを請負代行者の作業空間に入れることで、仕事の依頼を行う。また請負代行者は、依頼された仕事が終了したら、その結果である作業情報オブジェクトを依頼代行者に返すことで、結果を返す。

3.1.3 作業情報オブジェクトのメタオブジェクトによるモデル化

作業空間内の作業情報オブジェクトに関して、必要なメソッドやアクセス権を構成員の役割に応じて、動的に設定、変更する必要がある。これはメタオブジェクトによるリフレクション機能を利用することで解決できる。

リフレクションは、計算システムが自分自身の計算過程に対し、自分自身を感知したり、自分自身を変更したりする機構である [3]。このリフレクション機構は例えば、システムの仕様変更やシステムの環境の変化に対し、システム全体を停止せずに柔軟に対応する目的で用いられている [8]。このリフレクションはメタレベルの表現単位により、次の 3 つのアーキテクチャに分類されている [6]。

1. 個体ベースのアーキテクチャ(Individual-based Architecture)

   システムの各オブジェクトが自分専用に、振舞いを表現するメタオブジェクトを持つ。

2. グループベースのアーキテクチャ(Group-wide Reflective Architecture)

   オブジェクトのグループの集団的な振舞いを、メタレベルで表現する。

3. ハイブリッドなアーキテクチャ(Hybrid Group Reflective Architecture)

   各オブジェクトの振舞いを表現するメタオブジェクトと、グループの振舞いを表現するメタオブジェクトとがメタレベルに存在する。

3.1.4 オブジェクトによる協調活動モデル

図 4に記した代行者による分散開発のモデルを我々が提案するオブジェクトモデルにより表現したのが、図 6である。図 6により、作業情報オブジェクトの集合によりグループオブジェクトの作業空間が実現されている様子がわかる。ここで各作業情報オブジェクトは、どのグループ作業空間に属しているか、また各グループオブジェクトは、自分の作業空間にどの作業情報オブジェクトが存在するかを知っている。依頼グループオブジェクトは、仕事の依頼時に各メソッドのアクセス権を設定して依頼することで、請負グループオブジェクトは、常に最適な作業空間を構成することができる。

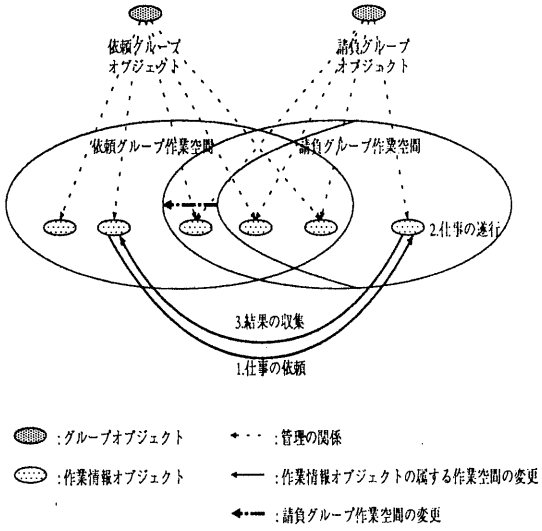

図 6: オブジェクトによる協調活動モデル

4 ABCL/R2 による実装例

この章では、自己反映計算の機能を持つ並列オブジェクト指向言語 ABCL/R2[4][5][6] により、今まで述べてきた協調活動モデルを実装し、シミュレーションを行ったので、その結果について述べる。

4.1 シミュレーションの概要

このシミュレーションの目的は、まず第一に我々が提案する協調活動モデルがコンピュータ上で実現可能か確認する点にある。そして第二点目としては、作業空間が、仕事を行う上でどの程度有効な環境を提供するか確認する点にある。このような目的で、以下のような依頼元 A が2つのソースコードを完成させる仕事をシミュレーションした。内容は、次の通りである。

依頼元 A が、仕事 J_1 と J_2 の仕様書を作成し、仕様書をもとにソースコードを作成することを仕事として、一方の仕事 J_2 を請負先 B に発注する。請負先 B は、示された仕様書をもとに、ソースコードを完成したら、そのソースコードを依頼元 A に、仕事の結果として返す。

4.2 各オブジェクトの実装の概要

4.2.1 依頼/請負グループオブジェクト

```
;;; グループオブジェクトジェネレータ
[object group-obj-gen
 (script
  (=> :new
     ![object group-obj ; グループオブジェクト
     (state
      ;; 属性の定義
      ...
     (script
      (=> [:order e-name prod w-name] ; 仕事の依頼
         ;; e-name は、作業情報の名前
         ;; w-name は、依頼先グループオブジェクト名
         (temporary ret e-obj w-obj)
         ;; e-obj は、作業情報オブジェクト ID
         ;; w-obj は、請負オブジェクト ID
         ...
         [w-obj <= [:join e-obj prod Me]])
      (=> [:finish e-name prod] ; 仕事の終了通知
         ...
         [w-obj <= [:notify e-obj]])
      ;作業空間への参加
      (=> [:join e-obj prod sw-obj]
         ;; sw-obj は依頼元オブジェクト ID
         ...)
      (=> [:notify e-obj] ; 仕事が終了したことの通知
         ...)
      (=> :create-job ; 作業情報オブジェクトの作成
         ![ent-obj-gen <== :new])
      ;成果物オブジェクトの作成
      (=> [:create-prod e-name prod]
         ;; prod は、成果物オブジェクトの種別
         ...)
      ;成果物オブジェクトにデータを設定
      (=> [:set-prod e-name prod data]
         ...
         [e-obj <= [:set prod data]])
      ;成果物オブジェクトにアクセス権の設定
      (=> [:set-acc e-name prod m-name u-name acc]
         ;; m-name は、メソッド名
         ;; u-name は、ユーザ名
         ;; acc は、アクセス権 (Yes,No,Hide)
         ...
         [e-obj <=
              [:set-acc prod s-name t-type acc]])
         ))]))]
```

図 7: 依頼/請負グループオブジェクト

依頼および請負グループオブジェクトは、図 7 に示されているようなグループオブジェクトジェネレータにより作成される。これらのオブジェクトの役割は、作業情報オブジェクトの管理である。依頼/請負グループオブジェクトは、

- 仕事の依頼 ([:order ...])
- 仕事の終了通知 ([:finish ...])

- 作業空間への参加 ([:join ...])
- 作業情報オブジェクトの作成 (:create-job)
- 成果物オブジェクトの登録 ([:set-prod ...])
- アクセス権の変更 ([:set-acc ...])

などの機能を持つ。

4.2.2 作業情報オブジェクト

作業情報オブジェクトは、1個のエントリオブジェクト (図 8) と任意個の成果物オブジェクト (図 9) から構成される。このシミュレーションでは、2つのエントリオブジェクトとそれぞれのエントリオブジェクトに対し、仕様書とソースコードという2つの成果物オブジェクトが付随している。グループオブジェクトは、

- 成果物オブジェクトの登録 ([:register ...])
- 成果物データの取得 ([:get ...])
- 成果物データの設定 ([:set ...])
- 成果物オブジェクトのアクセス権の設定 ([:set-acc ...])

などといった要求をエントリオブジェクトに対し行う。エントリオブジェクトは、受け付けたメッセージを適当な成果物オブジェクトに対し、要求を出す。成果物オブジェクトのメタオブジェクトは、成果物オブジェクトの各メソッドのアクセス権を管理しており、要求されたメソッドのアクセス可能性は、メタオブジェクトが検査し、アクセス権がない場合は、そのアクセスを拒絶する。

4.3 評価

このシミュレーションにより依頼グループオブジェクトが、必要な作業情報オブジェクトの各メソッドに、適当なアクセス権を設定して請負グループオブジェクトの作業空間に設定することで、請負グループオブジェクトが仕事を行うのに必要十分な作業環境が構築されることがわかった。しかし今回のシミュレーションでは、

```
;;; エントリオブジェクトジェネレータ
[object ent-obj-gen
 (script
  (=> :new from c-worker
     ![object ent-obj ; エントリオブジェクト
     (state
      ...
     (script
      ; 成果物オブジェクトの登録
      (=> [:register name p-obj] from cr-obj
         [p-obj <= [:set-creator cr-obj creator]])
      ; 成果物データの取得
      (=> [:get name] from cr-obj
         (temporary p-obj)
         ;; p-obj 成果物オブジェクト ID
         ![p-obj <== [:get cr-obj]])
      ; 成果物データの設定
      (=> [:set name data] from cr-obj
         ...
         [p-obj <= [:set cr-obj data]])
      ; アクセス権の設定
      (=> [:set-acc name src t-type acc] from cr-obj
         ;; p-obj 成果物オブジェクト ID
         ...
         [p-obj <=
           [:set-acc cr-obj s-name acc w-type]])
         )]))]
```

図 8: エントリオブジェクト

言語的制約により、単一マシン上で行った。今後は、分散処理環境で、このモデルの有効性について検証する必要がある。

5 まとめ

本研究では、仕事の依頼/結果の収集という非常に単純ではあるが、現実世界でよくみられる協調活動をモデル化した。この協調活動において、各個人の作業空間をどのように設定するかということが非常に重要であることがわかった。仕事を行う上で、必要十分な情報を、適当なアクセス権を設定して仕事の請負代行者に提供することが必要である。しかも仕事の内容や仕事に関連した仕事が内容により変化する。また、仕事開始当初には、すべてを洗い出しておくことができないというのが一般的である。このような状況に、今回提案したオブジェクト指向アーキテクチャは、十分対応できると評価している。

現在、協調活動「結果の収集」の部分に関するモデル化を検討中である。

```
;;; 成果物オブジェクトジェネレータ
[object prod-obj-gen
 (script
  (=> :new
     [object prod-obj ; 成果物オブジェクト
     (group prod-group)
     (state
      ...
     (script
      ;作成者の設定
      (=> [:set-creator cr-obj cw-obj]
         [[meta Me] <= [:set-creator cw-obj]])
      (=> [:get cr-obj] ; 成果物データの取得
         !prod-data)
      (=> [:set cr-obj data] ; 成果物データの設定
         [prod-data := data])
      ; アクセス権の設定
      (=> [:set-acc cr-obj m-name acc w-type]
         [[meta Me] <=
                [:set-acc m-name acc w-type]])
       )]))]
```

図 9: 成果物オブジェクト

参考文献

[1] Gehani,N., and Jagadish,H.V.: Ode as an Active Database: Constraints and Triggers. Proceedings of the 17th International Conference on Very Large Databases, (1991), pp.327-336.

[2] 門脇 千恵、落水 浩一郎: 非同期分散方会議の事象駆動型討議プロセスによるモデル化と調整支援への応用. 情処研報 ソフトウェア工学, Vol.96, No.25 (1994), pp.193-200.

[3] Maes,P.: Concepts and experiments in computational reflection. In Proceedings of OOPSLA'87, Vol.22, pp.147-155. SIGPLAN Notices, ACM Press, Oct. 1987.

[4] 増原 英彦、松岡 聡、渡部 卓雄、米澤 明憲: 自己反映計算の機能を持つ並列オブジェクト指向言語の高能率な実現方式. レクチャーノート/ソフトウェア学 4 オブジェクト指向コンピューティング I,(1993), pp.209-224, 近代科学社.

[5] Matsuoka,S., Watanabe,T., and Yonezawa,A.: Hybrid Group Reflective Architecture for Object-Oriented Concurrent Reflective Programming. Proceedings of ECOOP'91, No.512 in Lecture Notes in Computer Science (1991), pp.231-250, Springer-Verlag.

[6] Matsuoka,S., Watanabe,T., Ichisugi,Y., and Yonezawa,A.: Object-Oriented Concurrent Reflective Architectures. In Proceedings of the Workshop on Object-Based Concurrent Programming. (Proceedings to be published as Tokoro,M. et.al.eds. a volume in Lecture Not4es in Computer Science,Springer-Verlag, 1992), 1991..

[7] 落水 浩一郎: ソフトウェアプロセスに関する研究の現状. 情処研報 ソフトウェア工学, Vol.93, No.59 (1993), pp.59-67.

[8] 岡村 英明、石川 裕、所 真理雄:リフレクション機能を持つ分散プログラミング言語システム AL-1/D. レクチャーノート/ソフトウェア学 4 オブジェクト指向コンピューティング I, (1993), pp.1-16.

[9] Rodden,T., Mariani,J.A. and Blair,G.: Supporting Cooperative Applications. Computer Supported Cooperative Work, (1992), No.1, pp.41-67.