グループウェア 18－5
（1996. 6. 14）

CSCWアプリ間の連携を重視したグループウェア・フレームワークCCF

深澤寿彦　黒澤貴弘　井上 淳　吉本雅彦　佐藤宏明　山川 正
キヤノン(株) CMプロジェクト

近年のグループウェア・アプリケーション(CSCWアプリ)の多様化は、複数のCSCWアプリを組み合わせて使用したいというニーズを生み出している。

本稿では、複数のCSCWアプリを連携させて動作させるための機構を提供するグループウェア・フレームワークCCF(Collaboration Control Facility)について述べる。

CSCWアプリの連携は、各CSCWアプリを構成するプロセス群の間の相互作用と、CSCWアプリがユーザに提供する協調作業モデル間の相互作用として扱うことができる。CCFでは、プロセス群の間の相互作用を記述するNodeモデルを提供する。また、協調作業モデル間の相互作用のサポートのために、モデルの構成要素間の関係を記述し、制御するためのプログラマ・インタフェースを提供する。

A Groupware Framework - CCF for Cooperation of CSCW Applications

Toshihiko Fukasawa, Takahiro Kurosawa, Sunao Inoue,
Masahiko Yoshimoto, Hiroaki Sato, Tadashi Yamakawa
CM Project, Canon, Inc.

The recent increase of the variety of groupware applications(CSCW applications) fosters people's needs for making CSCW applications cooperate with each other.

In this paper, we describe a groupware framework CCF: Collaboration Control Facility which provides mechanisms for making CSCW applications cooperate with each other.

The cooperation of CSCW applications is treated as interaction with processes of other applications or interaction with other cooperation models. CCF provides "Node model" in order to program and control the interaction with processes of other applications. CCF also provides program interface to program relations of cooperation models for the interaction with other cooperation models.

1 はじめに

近年、高速ネットワークとマルチメディア・コンピューティングの急速な発展に伴い，CSCW(Computer Supported Cooperative Work) 分野における研究開発が活発になってきている。

空間的に分散した環境間で同期的に行なわれる協調作業を支援することを目的とした同期型 CSCW アプリケーション (以下 CSCW アプリと略す) の分野においてもその応用形態は多様化の兆しを見せており、遠隔会議システムや遠隔プレゼンテーション・システム等の協調作業を直接的に支援するものだけでなく、遠隔地に設置されたビデオ・カメラの画像を参照する遠隔状況把握アプリなど、協調作業の準備段階なども含めた総合的な協調作業の支援が指向されてきている。

従来、グループウェア・フレームワークは個々の CSCW アプリをいかに簡便に実現するかという問題に取り組んできた。これらのグループウェア・フレームワークでは、複数のホストに分散したプロセスが協調してユーザに協調作業環境を提供するソフトウェアとして CSCW アプリを実現することをサポートする。そして、CSCW アプリ内の個々のプロセス／プログラムをいかに協調して動作させるかという問題に注力してきた。

その結果、データ共有, メッセージなどによるデータ交換の機構、Session,Conference などの協調作業の場のメタファーに基づくプロセス制御の機構、協調作業向けユーザインタフェースなどさまざまなメカニズム／プログラム・インタフェースが提案されている [1][2][3]。

これに対し、本稿では複数の CSCW アプリ間の相互作用や関係をあつかう枠組を対象とし、このような枠組を提供するグループウェア・フレームワークのプロトタイプである CCF (Collaboration Control Facility) について述べる。

CCF では、CSCW アプリの協調作業モデル／プロセス構成の両面についての CSCW アプリ間の差異を記述する方法を提供することにより、CSCW アプリ内だけでなく、CSCW アプリ間での協調動作を可能にする。

以下ではまず、CSCW アプリ間の協調動作の 1 つである連携について考察し、この連携を実現するうえで解決しなければならない問題点と解決法を示す。次に、CCF における連携のサポートについて説明していく。

2 CSCW アプリの連携

まず、例を用いて CSCW アプリ間の連携について説明する。ここでは、遠隔地に設置されたビデオカメラの画像を参照するための「遠隔状況把握アプリ」と「ビデオ会議アプリ」の組み合わせにおいて、「状況把握アプリ」でユーザの在室を確認してから、そのユーザと「ビデオ会議アプリ」で作業を開始する、という状況を考える (図 1)。

この組み合わせにおいては、「状況把握アプリ」でのユーザの検索／確認作業から「ビデオ会議アプリ」での共同作業へのスムーズな移行をサポートすることが望まれる。この要望を実現するためには、2 つの CSCW アプリ間でのユーザ等に関する情報の受渡しや、状況把握アプリが使用していたカメラ等の資源を効率良くビデオ会議アプリに受渡すなどの相互作用を行なって、ユーザインタフェース上の不整合や各 CSCW アプリ間の資源の競合などをうまく解消しなければならない。同様な要求は、「状況把握アプリ」と「ビデオ会議アプリ」が同時に

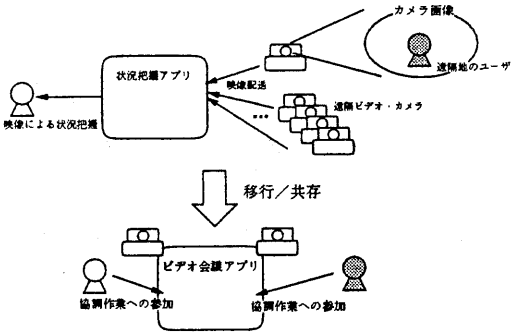

図 1: 遠隔状況把握からビデオ会議への移行

動作 (共存) する場合においても発生する。

そこで「CSCW アプリの連携」を、上記の例の様な

- ある CSCW アプリから別の CSCW アプリへの移行
- 2 つ以上の CSCW アプリの共存

において CSCW アプリ間に発生するさまざまな相互作用のことであると定義する。

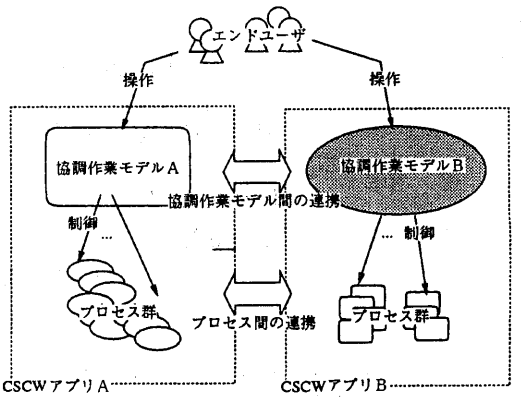

図 2: CSCW アプリの連携

CSCW アプリの連携は、連携の対象によって大きく以下の 2 種類に分けることが可能である (図 2)。

1. **協調プロセス群間の連携**
2. **協調作業モデル間の連携**

一般に、CSCW アプリは複数のホストに分散して動作するプロセス群として実現される。このため、CSCW アプリの移行／共存は、各 CSCW アプリを構成するプロセス群 (協調プロセス群と呼ぶ) の間の移行／共存としての側面を持っている。

「**1. 協調プロセス群間の連携**」とは、この協調プロセス群の間の移行／共存として実現する際に必要となる連携のことである。この連携は、CSCW アプリをいかに効

率よく協調させるかという問題に関わっており、具体的には以下の2種類の連携が考えられる。

- **共有可能なハードウェア資源およびソフトウェア・モジュールの重複の解消**
  移行の対象となるCSCWアプリ間で、同一のビデオカメラ, マイク等の資源を利用したり、同一のプログラム, プロセスを構成要素としている場合がある。この場合は、資源やプロセスの再利用や共有を可能にすることにより、移行／共存を効率良く実現することが可能となる。
  このサポートがなければ、同一プログラムを要求のたびに起動してしまう様な状況が発生してしまう等の問題が発生する。
- **共有不能なハードウェア資源のアクセス制御**
  協調作業形態の共存においては、資源そのものが排他的なアクセスしか許さない様な状況も発生する。この場合、CSCWアプリ間での資源へのアクセス制御の機構が不可欠となる。

また、もう一つのタイプの連携である「**2. 協調作業モデル間の連携**」とは、CSCWアプリがユーザに対して提供する「会議」,「講演会」,「共同執筆」などの協調作業モデル間の連携である。

この連携は、ユーザに対して複数のCSCWアプリを如何に統一的に提示するかというユーザインタフェースの問題に関わっている。その意味で協調作業モデル間の連携は、対象となる協調作業モデルによって様々な相互作用が発生する。代表的な連携として以下のものが挙げておく。

- **協調作業モデルの構成要素間の関連の制御**
  協調作業形態が異なれば、その構成要素も異なってくる。しかし、移行／共存時には、その構成要素間に何らかの関連が発生することがある。
  例えば、「遠隔状況把握アプリ」から「会議アプリ」への移行を考える。ここで、「遠隔状況把握アプリ」では、遠隔地に設置されたビデオ・カメラを協調作業形態の基本的な構成要素となる。これに対し、「会議アプリ」では会議の参加者であるユーザを基本的な構成要素とする。
  2つのCSCWアプリ間の移行において、移行前のビデオ・カメラに写っていた人物が移行後のユーザになるという関連が存在し得る。この関連を制御可能にすることにより、カメラに写っていたはずのユーザを、「会議アプリ」起動後に改めて会議の参加者として指定しなければならない、といった状況を避けることが可能となる。

以上のような連携をサポートする上で障害となるのは、移行／共存の対象となる各CSCWアプリにおける

- **プロセス構成と資源へのアクセス形態の違い**
- **協調作業モデルの構成要素、およびその振舞いの違い**

が必ずしも明らかではないことである。これらの差異が明確でなければ、各構成要素の対応関係等を決定することができず、連携の記述(プログラミング)／制御の実現が困難になる。この問題はCSCWアプリの多様化に伴い、ますます「協調プロセス群間の連携」,「協調作業モデル間の連携」を実現する上での大きな障害となると考えられる。

我々が検討を行なっているグループウェア・フレームワークCCF (Collaboration Control Facility) では上記の問題に対し、協調プロセス群／協調作業モデルの両面について、

- その構成／形態／振舞いの記述を可能にし、
- その記述に基づいた、連携のプログラミングおよび制御手段を提供する、

ことによりCSCWアプリの連携の実現をサポートする。

次章以降では、CCFの概要について紹介してから、協調プロセス群間の連携／協調作業モデル間の連携の各々についてのCCFにおける実現について説明していく。

3 グループウェア・フレームワークCCFの概要

グループウェア・フレームワークとしてのCCFは、C++ツールキット **CCF Toolkit** を提供する。CCF Toolkitは、協調プロセス群と協調作業モデルの構成／振舞い等を記述するためのプログラマ・インタフェースを提供する。また、CSCWアプリ間で行なわれる連携を記述するためのプログラマ・インタフェースもCCF Toolkitによって提供される。

CCF Toolkitを組み込んで作成されたCSCWアプリは **CCF Core System** によって制御／管理される。

CCF Core Systemは、CCF Core ServerとCCF Core Interfaceという2種類のプログラムで構成されている。CCF Core ServerはCSCWアプリに関する情報を管理する集中サーバである。また、CCF Core Interfaceは利用者に起動等CSCWアプリの制御インタフェースを提供するプログラムである。

図3にCCF Core Systemの概念図を示す。

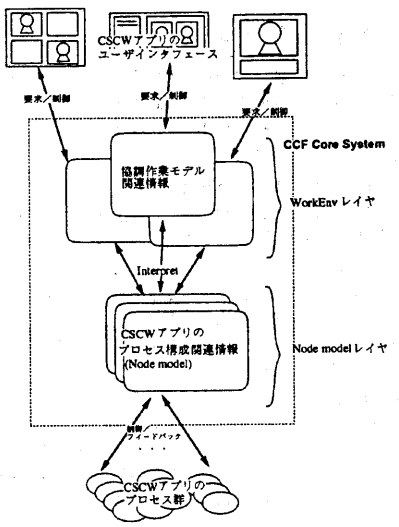

図 3: CCF Core Systemの概念図

CCF Core Systemの内部は、2章で述べた2種類の連携のタイプに対応した以下の2つのレイヤから構成されている。

- Node model レイヤ
  CSCW アプリがどの様なプロセスから構成され、各プロセスが構成要素としてどの様に関連しているかを管理するレイヤである。後述する Node モデルという CSCW アプリのモデルに基づき、CSCW アプリのプロセスの起動, 終了等の制御を行なう。連携の制御においては、協調プロセス群の間の連携機構を提供する。
- WorkEnv レイヤ
  CSCW アプリの協調作業モデルを扱うレイヤである。各 CSCW アプリが提供する協調作業モデル上の操作と、Node model レイヤでのプロセス単位の処理の対応を管理／制御する。連携の制御においては、協調作業モデル間の連携機構を提供する。

では、次章から協調プロセス群および協調作業モデルにおける連携サポートについて説明する。

4 協調プロセス群における連携のサポート

ここでは、CCF による協調プロセス群の連携サポートについて説明する。協調プロセス群の連携は Node model レイヤの管理下に置かれている。

Node model レイヤは、

- CSCW アプリを構成する協調プロセス群を記述するための Node モデル
- Node モデルに基づく、連携の制御機構

を提供することにより連携をサポートする。

4.1 Node モデル

Node モデルは、我々が検討しているプログラム／プロセス単位での CSCW アプリ制御モデルである。この Node モデルは以下の構成要素から成り立っている。

- **Node**
  CSCW アプリのプログラムが動作するユーザ環境やホストを抽象化したものである。後述する Resource や Module の管理単位として機能する。
- **Module**
  Module は CSCW アプリを構成するプロセスを抽象化したものである。Node 内で管理されている Module は、Node に対応するホスト上のプロセスとみなすことができる。
- **Resource**
  Resource はカメラ、マイク等の外部入出力機器を代表とする資源を表現する。Node 内で管理されている Resource は、Node に対応するホストに設置されている外部入出力機器あるいはその管理サーバとみなすことができる。
- **Channel**
  Channel は Module および Resource 間の通信路を表現する。1:1 の通信だけでなく、broadcast の様な 1:N の通信を 1 つの Channel で表現することができる。また、Channel は同一 Node 内の Module,Resource 間だけでなく、異なる Node の Module,Resource 間でも形成することができる。

Node モデルで表現された CSCW アプリの例を図 4 に示す。Node モデルは、CSCW アプリの協調プロセス群が動作する各ホストにおける、

- 動作するプロセス
- 使用される資源
- プロセス間の通信路および資源へのアクセス・パス

を記述する。

Node モデルにより、CSCW アプリは、関連するホストと対応づけられた Node の集まりとして表現される。上記の各構成要素には、それぞれ一意な名前が付加され、CCF Core Server によって集中管理される。また、CCF Core Server は Node モデルを使用して、実際の CSCW アプリの振舞いをモニタし、必要に応じて制御を行なう。

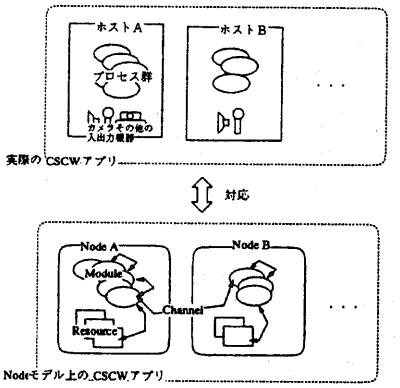

図 4: Node モデルの例

上記の様な Node モデルを形成するためには、個々のグループウェア・アプリがどの様な Node を形成するかをあらかじめ記述しておく必要がある。

この Node モデルの設計図となる情報を Node フレーム情報と呼び、Node フレーム情報作成のためのインタフェースは CCF Toolkit の C++クラス群によって提供される[1]。

Node フレーム情報も CCF Core Server によって管理され、必要に応じて Node モデルの作成に使用される。

4.2 Node モデルに基づく連携の制御

2 章で述べたように、協調プロセス群の連携は CSCW アプリの移行／共存時に必要となるプロセスおよび資源の共有／アクセス制御として扱うことができる。

このプロセスおよび資源の共有／アクセス制御は、Node,Module,Resource,Channel の置換処理として記述／制御することが可能である。

図 5に、この Node モデルでの連携処理のアルゴリズムを示す。図中で、2 つの CSCW アプリ A から B への移行が行なわれるものとする。さらに、これらのアプリが両方とも Node を形成するホストに注目すると、CSCW アプリ A の Node1 から CSCW アプリ B の Node2 への

[1] Node フレーム情報の記述には、C++クラスではなく専用の登録ツールを用意する方法も考えられる

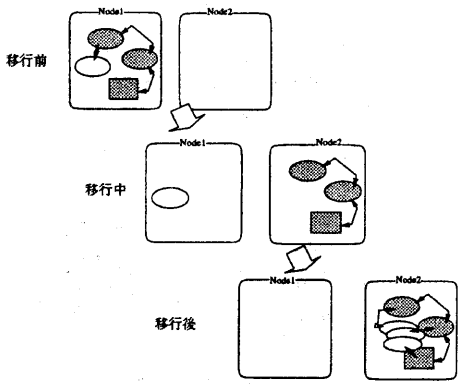

図 5: 連携処理のアルゴリズム

移行が行なわれるものとする (図 5-移行前)。この Node 間の移行は、Node1 の Module,Resource,Channel を用いて Node2 を再構成することによって実現される。

この再構成処理は、移行前の Node1 と移行後の Node2 を比較し、同一名の Module,Resource が使用されている場合には、これらを再利用する (図 5-移行中)。ただし、同一名であっても Module,Resource に対して再利用が不可である旨が指定されている場合を除く。また、Channel については同一名であっても、Channel の両端の Module,Resource が再利用されていなければ、Channel は再利用されない。この様して再利用された Module,Resource,Channel では、対応するプロセス等が CSCW アプリ B の一部として使用される。

再利用される構成要素が決定されれば、Node2 の生成に不足している Module,Resource を生成し、最後に全ての Channel を生成する (図 5-移行後)。この処理によって新たなプロセスが起動され、さらに再利用されたプロセスと新しいプロセス間での通信路が形成されることになる。

以上の処理をすべての Node に対して実行することにより、CSCW アプリ A から CSCW アプリ B への移行が完了する。

また、CSCW アプリ間の共存の処理も上記の様にして行なわれるが、Resource に対しては、どちらか一方のアプリしか資源にアクセスできない様な状況が発生する。この場合、Node モデル上では、両方の Node に同じ Resource が存在していることになる。ただし、その Resource と他の Module 間に Channel を形成できるのは一方の Node だけである。必要に応じて、Channel を張り替えることにより、資源を共有する CSCW アプリ間でのアクセス制御を Node モデル上で扱うことが可能となる。

以上の様に、CSCW アプリの開発者は、提供する CSCW アプリの Node モデルによる記述と、その中で再利用可能な Module,Resource,Channel を記述することにより、他の CSCW アプリとの協調プロセス群間の連携に必要な情報を提供することができる。

4.3 Node モデルによる連携制御の実現

Node モデルで記述された CSCW アプリに対して、CCF Core System の Node model レイヤは Node モデル上の各構成要素と実際のホスト, プロセス, 資源, 通信路の対応関係を管理する。

そして、Node モデル上の操作を実際のホスト, プロセス, 資源, 通信路に反映させる。また逆に、実際のホスト, プロセス, 資源, 通信路の振舞いをモニタすることにより、結果を Node モデル側に反映させるなどの処理を行なう。

Node モデルから実際への反映は、ユーザが CCF Core Interface などを介して CSCW アプリの操作を要求したときに行なわれる。

実際のホスト, プロセス, 資源, 通信路から Node モデルへの反映は、CSCW アプリ側が独自にアクションを起こしてから、その旨を CCF 側に通知する場合に行なわれる。また、エラー等によるプロセスを異常終了を CCF Core Server が検知した場合も、関連する Module,Resource,Channel の消去が行なわれる。

以下に、Node モデル上で可能な操作と、それに対応する CCF 側の処理を列挙する。

- Node の生成および消去
  Node 内部の Module,Resource,Chanell の生成削除
- Module の生成および消去
  対応するプログラムの起動／終了
- Resource の生成および消去
  対応する資源管理サーバの起動／終了
- Channel の生成および消去
  対応する通信路の生成／消去処理

協調プロセス群の連携においては、アクセス制御等の実現のために Channel の制御の実現が重要である。Channel に対応する通信路の制御は、CCF 側からグループウェア・アプリのプログラムに対して送られる以下のメッセージのハンドラ手続きによって実行される。このメッセージ・ハンドラは、CSCW アプリの開発者によって提供され、socket,rpc 等による実際の通信路の形成処理を行なう。

- CREATE-CHANNEL # Channel の生成
- DESTROY-CHANNEL # Channel の消去

通信路の形成処理は、Channel の両端の Module ないし Resource のどちらか 1 方がオーナとなり、上記のメッセージを受け取って通信路の生成／消去を行なう (図 6)。また、オーナではない Module,Resource に対応するプロセスには、通信路生成／消去の通知メッセージが、オーナ側の処理の終了後に送られる。通知メッセージには、socket,RPC 等で通信路を確立するために必要なポート番号等の情報が付加されるので、通知メッセージを受け取ったプロセスが通信路を確立することが可能となる。

5 協調作業モデルにおける連携のサポート

CCF の WorkEnv レイヤでは、協調作業モデルの連携における協調作業モデル間の情報の対応をあつかう。

Node model レイヤにおける協調プロセス群の連携サポートの機構は、CCF Core Server によって提供されたが、協調作業モデル間の連携制御機構は CSCW アプリ側によって提供される。

そのかわりに、CCF は

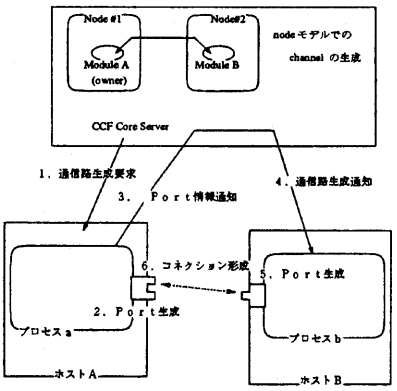

図 6: Channel による通信路の形成

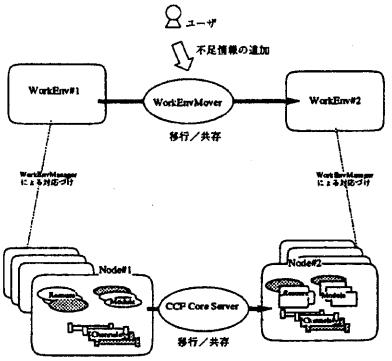

図 7: WorkEnv レイヤでの連携サポート

- 各 CSCW アプリが提供する「会議」,「講演会」といった協調作業モデルを記述するための C++インタフェースと、
- 2 つの協調作業モデル間におけるユーザ等の構成要素の対応関係を記述／制御するための C++インタフェース

を提供する。

この協調作業モデルの記述は、

- 協調作業モデルの構成要素と，Node モデルの構成要素の対応関係
- 協調作業モデルの構成要素に対する操作と，Node モデル上の操作の対応関係

をプログラミングし、CCF Core System にそのプログラムを追加することによって行なわれる。このプログラムを WorkEnvManager と呼んでいる。

また、協調作業モデル間の対応関係も同様にプログラムとして記述される。このプログラムは WorkEnvMover と呼ばれる。WorkEnvMover は、CSCW アプリ間の連携時に CCF Core Server の WorkEnv レイヤによって起動され、協調作業モデル間の情報の共有／再利用処理を実行する。(図 7)

WorkEnvManager および WorkEnvMover は CSCW アプリの一部としてアプリ側の開発者によって提供されることになるが、CCF はこれらのプログラムを利用して協調作業モデル間の連携を実現する。

6 おわりに

本稿では、同期型 CSCW アプリケーション間の連携について述べ、そのグループウェア・フレームワークによるサポートの実現における問題点とその解決方法について述べた。また、グループウェアフレームワーク CCF における実現について説明した。

CSCW アプリの連携は、各 CSCW アプリを構成するプロセス群の間の相互作用と、CSCW アプリがユーザに提供する協調作業モデル間の相互作用としてあらわされる。

連携のサポートは、これらの相互作用に対して

- 協調プロセス群／協調作業モデルの構成／形態／振舞いの記述を可能にし、
- その記述に基づいた、連携のプログラミングおよび制御手段を提供する

ことにより実現することができる。

CCF では、協調プロセス群の間の相互作用を Node モデルによって記述／制御する。また、協調作業モデル間のサポートとしては、モデルの構成要素間の関係を記述し、制御するためのプログラマ・インタフェースを提供している。

ただし、本来は協調作業モデル間のサポートにおいて、構成要素間の関係以外の連携も考慮する必要がある。このことから今後の課題としては、構成要素の対応関係以外の連携も記述することができる協調作業記述モデルの具体化がもっとも重要であると考えている。

参考文献

[1] Earl Craighill et al., "SCOOT: An Object-Oriented Toolkit for Multimedia Collaboration", Proc. ACM Multimedia 94, pp.41-49 (1994).

[2] Pavel Curtis et al., "The Jupiter Audio/Video Architecture:Secure Multimedia in Network Places", Proc. ACM Multimedia 95, pp.79-90 (1995).

[3] Ralph D. Hill et al., "The Rendezvous Architecture and Language for Constructing Multiuser Applications", ACM trans. CHI, vol.1, No.2, pp.81-125 (1994).

[4] Mark Roseman et al., "Groupkit: A groupware toolkit for building realtime conferencing applications", Proc. ACM CSCW 92, pp.43-50 (1992).

[5] 深澤, 他, "グループウェア・アプリケーションフレームワーク CCF(Collaboration Control Facility) の構想", 第 52 回情処全大論文集, vol.6, pp.271-272(1996).