協調ハイパーメディアシステムについて

牛田修司, 神谷慎吾,谷口秀夫

NTTデータ通信(株)　開発本部

システム開発作業は、提案書や仕様書などの様々な生産物を複数のユーザが作成する複雑な協調活動である。我々はシステム開発支援を目的に一連の作業を支援する環境である協調ハイパーメディアシステム(CHMS:Cooperative Hyper Media Syetm)の開発を進めている。

CHMSは、更新伝播や排他制御の機能を持つ機能部品という共有データを提供する。また、新しい機能部品の定義や機能部品間の関係を記述する機能を提供する。

CHMSでは各作業の生産物は機能部品とその組み合わせによって表現され、これによってシステム開発の作業支援を行なうものである。

CHMS:An Infrastructure for building groupware application

Shuji Ushida, Shingo Kamiya, Hideo Taniguchi

NTT DATA COMMUNICATIONS SYSTEMS Corp.

System development process consists of complex cooperative work, in which many people create various documents.

This paper describes the conceptual model of CHMS(Cooperative Hyper Media System) as an environment that supports cooperative work for creating application softwares in system development.

CHMS provides shared objects with change propagation and exclusive control. Users can delive new shared objects from existing objects and can define scripts among objects.

Each documents are represented by the combinations of the shared objects and constraints among them in CHMS.

1.はじめに

計算機を用いて協調活動を支援する研究が、活発に進められている[1][2]。最近は、支援効果の評価方法に関する研究結果[3]や協調活動自体の深い分析[4]も報告されている。具体的な支援方法を提案している研究例として、電子会議などのリアルタイム・コミュニケーションを支援するもの[5]や、オフィスフロー等の定型的な業務の流れを支援するもの[6][7]の報告がある。これらの多くは、作業内容が比較的均質な特定の共同作業局面を支援する場合を扱っている。

これに対し、システム開発作業は提案書や仕様書等の様々な生産物を多くの人間が作成／利用するという複雑な協調活動である。そのため、生産物は互いに密接な関連を持っている(図1)。

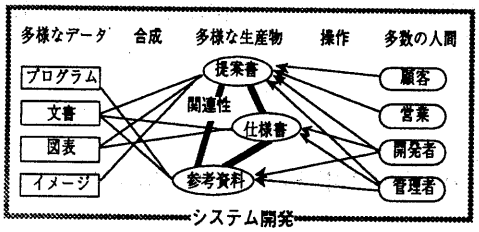

図1. システム開発の全体像と協調活動

我々は、システム開発支援を目的として協調ハイパーメディア・システム(Cooperative HyperMedia System:以下CHMS)の構築を進めている。CHMSは、システム開発の個々の作業を支援するだけではなく、一連の作業の支援を行なう環境である。また、複数のユーザが様々なメディアを共有し、それらのメディア間に様々な関係付けを行なうシステムである。

複雑な構造を持ったデータを複数のユーザが共同操作する形態の支援方法については、マルチユーザ・ハイパーテキスト系の研究事例[8][9][10]が知られている。我々のシステムは、これらの報告にあるハイパーテキストの共有機能をベースとしているが、データ間の関連性の扱いと作業局面にあわせたカスタマイズ機能を重視している点に特徴がある。

本稿ではシステム開発における問題点を整理し、さらにCHMSにおける対処及び実現方法について報告する。

2. 問題点と対処

2.1 問題点

システム開発個々の作業はマシン利用が進んでいるが以下のような問題点がある。

(1) 支援システムが個別に実現されている

システム開発には様々な作業が存在する。現在それらを個々に支援するシステムはいくつか存在するが、一連の作業を一貫して支援するシステムは存在しない。そのため、ユーザインタフェースも不統一となる。支援される作業が不連続となり、ユーザの利用促進が阻害される。

また個々の作業を支援するシステムを個別に作成する場合、それぞれに通信機能やGUI機能等を組み込む必要があり、開発コストが高くつく。

(2) 各作業の生産物について相互の整合性が保証されない

例えば設計を行なう場合、仕様書の内容を基に作業が行なわれるが、その時点で仕様の変更等が生じた時仕様書に遡って内容を更新し、無矛盾性を保つ必要がある。このようにシステム開発では各作業は密接な関連を持っているが、現在は各作業の支援は独立で行なわれており、これら生産物間の関連については支援されていない。

(3) 作業局面の特徴や利用者の嗜好にあわせたカスタマイズが難しい

各生産物に関して、共通様式が決まっていても、業務分野毎に様式をチューニングする必要がある。また、同じ作業を行なう場合も様々なレベルのユーザがおり、各ユーザに適切な操作法を提供できる必要がある。

2.1 対処

作業工程別にシステム開発支援を実現していく場合、機能中心としたアプローチとなるため上のような問題が生じてしまう。我々は作業進展に従って受け渡されるデータに注目し、データ中心のアプローチでシステム開発支援を行なうことにより課題に対処していく。

CHMSでは機能部品と呼ばれるデータにGUIや通信などの機能を持たせ、様々な生産物は機能部品とその組み合わせによって実現する(問題点1への対処)。またCHMSでは関係オブジェクトと呼ばれるものによって機能部品間に様々な関係式を記述する機能を提供する(問題点2への対処)。機能部品の設定の変更や機能部品の定義をユーザに開放することにより、より柔軟なカスタマイズ環境を提供する(問題点3への対処)。

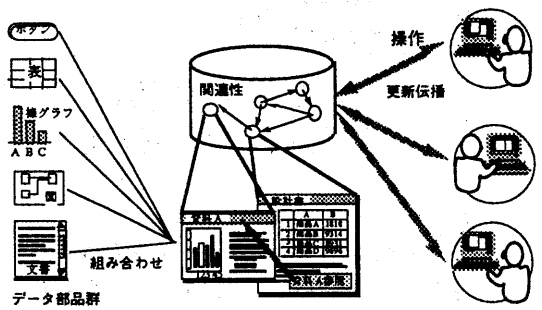

図2. CHMSのイメージ

3. 実現イメージ

3.1 機能部品の概要

3.1.1 機能部品の特徴

CHMSの処理の単位である機能部品は以下のような性質を持っている。

(1) テキスト、数値、図形などユーザにとって何らかの意味を持つ
(2) OODBによって管理される永続オブジェクトである
(3) 画面への視覚化およびユーザからの操作に関する処理を規定するGUIフレームワークを持つ
(4) 更新伝播、排他制御管理などの機能を持ち、複数のユーザから共有される

機能部品には、テキスト、数字、図形などのデータ部品以外に、他の機能部品のレイアウトを管理する構造化部品が存在する。構造化部品は機能を合成してより複雑な機能を実現する。CHMSでは基本的な機能部品を組み込み機能として提供し、それ以外は機能部品の合成と後に述べるユーザ定義の機能部品により実現する。

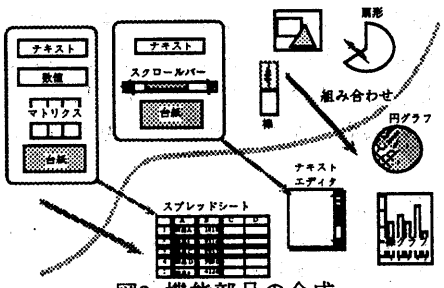

図3. 機能部品の合成

3.1.2 機能部品の構成

機能部品はOODB内の永続オブジェクトとして

実現されている。そのため、オブジェクトを実際のウィンドウ上に視覚化したり、ユーザからのイベントをオブジェクトに反映させるGUIフレームワークを機能部品に持たせている。CHMSでは機能部品の意味を表すデータ部とオブジェクトの表示やユーザからの要求を処理する部分を分けて実現することとし、SmalltalkのMVCモデルをもとに、Model/Representaion/Preference/Widgetの4つのオブジェクトから構成される形とした。それぞれのオブジェクトの役割を以下に示す。

(1) Model

MVCモデルのMに相当し、セマンティックなデータを格納するオブジェクト。永続オブジェクトとして実現され、複数のユーザから共有される。

(2) Representation

Representation(以下Rep.)はMVCモデルのVCに相当するものであり、Modelの視覚表現を決めると同時に、ユーザからの操作に対する処理が登録される永続オブジェクト。Model同様複数のユーザから共有される。

(3) Preference

Preference(以下Pref.)は各ユーザの個人的な情報を記述する永続オブジェクトであり、Rep.の管理する表示や操作に関する情報を上書き、あるいは捕捉する永続オブジェクト。Pref. はユーザ間では共有されずユーザ毎に用意される。

(4) Widget

Widgetは実際のウィンドウシステム上のウィンドウを管理するオブジェクト。ウィンドウに関する情報はウィンドウシステムに依存する非永続的な情報であり、そのためWidgetはユーザプロセスが起動された時、Model/Rep./Pref.の情報をもとに生成される非永続オブジェクトである。

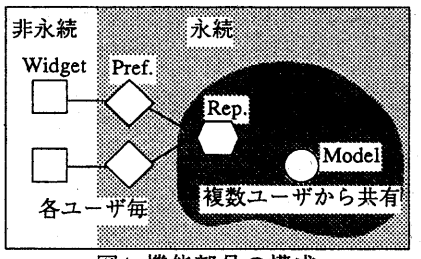

図4. 機能部品の構成

Modelには複数のRep.を登録することができ、各ユーザの画面にはそのModelに登録されているRep.の数だけ機能部品が表示される。機能部品の視覚表現はRep.によって決まるため、1つのModelに異なる種類のRep.を登録すると、Modelはそれぞれの Rep.で規定される視覚表現で表示される。図5は、テキストを保持するTextModelに対してテキストをそのまま表示するTextRep.とModelの内容には関係なくアイコンを表示するIconRep.を登録した例である。その時、画面上には文字列とアイコンが表示される。同様に1つのModelに同じ種類のRep.を複数登録するとModelが複数の場所に同じ視覚表現で表示される。

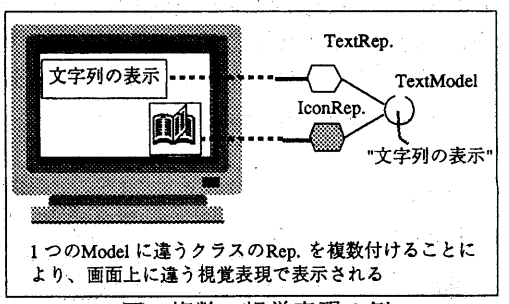

図5. 複数の視覚表現の例

ユーザからの操作に対する処理は、イベントの種類とそれに対応するメソッドを登録したトランスレーションテーブルによって行なわれる。トランスレーションテーブルはRep.とPref.に保持される。Rep.のトランスレーションテーブルはその視覚表現に適したModelに対する処理が登録され、Pref.のトランスレーションテーブルには

各ユーザが自分向けにカスタマイズした処理が登録される。ユーザからイベントはWidgetを介してPref.に送られ、Pref.は自身のトランスレーションテーブルを検索し、対応するイベントがあった場合そのメソッドが実行するが、見つからない場合はRep.にそのイベントを送り、Rep.は同様の処理を行なう。

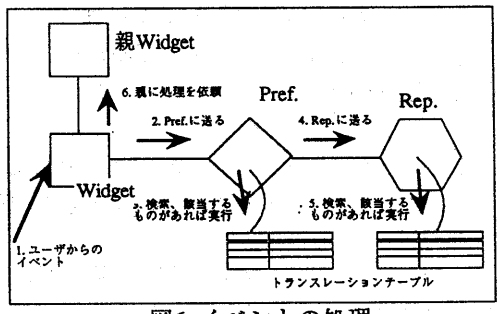

図6. イベントの処理

3.2 機能部品の共有

機能部品は複数のユーザから共有され、並行操作が行なわれるため、各クライアントプロセスとOODBとの無矛盾性を保証する機能を持つ必要がある。CHMSではクライアントプロセスとOODBの無矛盾性を保つため(機能部品を構成する)オブジェクトに更新伝播と呼ばれる機能や、排他制御の機能を持たせている。これらの機能は自動的に行なわれ、ユーザは意識しないで機能部品を利用することができる。また、マルチカーソルによって他ユーザの操作を伝える機能も持つ。

3.2.1 更新伝播

同期型協調作業でオブジェクトを共有するためには、各クライアントプロセス中の同一オブジェクトは内容が同一で、かつ画面上の表示もその内容が反映されている必要がある。そのためCHMSではあるオブジェクトの内容の変更を伝える更新伝播機構を用い、これらの無矛盾性を保っている。

更新伝播には1つのクライアントプロセスの中で無矛盾性を保証するためのプロセス内の更新伝播と、複数のクライアントプロセス間で無矛盾性を保証するためのプロセス間の更新伝播がある。

(1) プロセス内の更新伝播

プロセス内の更新伝播は基本的に画面表示の無矛盾性を保つために用いられる。CHMSの機能部品はそのGUIフレームワークにより、(1つのModelが)クライアントプロセス内で複数の場所に表示されうる。そのため機能部品の内容が更新された場合、その画面上への表示は更新された内容と矛盾が生じないように変更される必要がある。

CHMSのオブジェクトは更新伝播を行なうため、更新依存オブジェクトの集合を持っている。更新元オブジェクトの内容が更新された時、そのオブジェクトは自身の持つ更新依存集合内の各要素に更新伝播する。更新伝播を受けたオブジェクトにはそれぞれ自身の内容を更新するメソッドUpdateが規定されており、それを実行する。CHMSでは更新伝播は基本的にModel→Rep.→Pref.→Widgetという方向で行なわれ、最終的にWidgetがUpdate(すなわち画面表示の更新処理)を行なうことによって、画面表示の無矛盾性が保たれる。

なお更新依存オブジェクトは複数の更新元オブジェクトに更新依存することもできる。

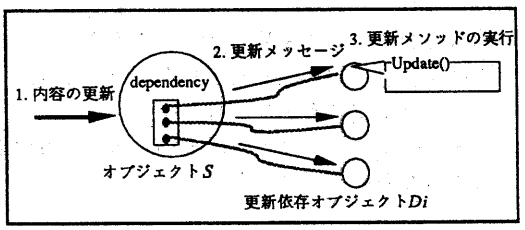

図7. プロセス内の更新伝播

(2) プロセス間の更新伝播

機能部品はOODBの永続オブジェクトとして実現されており、各ユーザのクライアントプロセスは利用するオブジェクトをメモリ内にロードして使用する。

実際のオブジェクトの更新はクライアントプロセス内で行なわれる処理以外に、他ユーザのプロセスにも伝える必要がある。プロセス内の更新伝播の処理により、あるクライアントプロセス内での無矛盾性は保たれるが、データベースは最新の情報を通知する機能を持たないため、あるクライアントプロセスがデータベースの内容を更新しても他のクライアントプロセスのオブジェクトの内容は更新されず、OODBとの間に矛盾が生じてしまうからである。

CHMSではリアルタイムのオブジェクト共有を実現するため、プロセス間の更新伝播の機能を設けており、それによりクライアントプロセス間でのオブジェクトの内容の無矛盾性を保証する。

プロセス間の更新伝播は、次のように行なわれる。オブジェクトの内容を更新したクライアントプロセスが他のクライアントプロセスに更新を通知し、更新情報を受け取った各クライアントプロセスはその情報をもとに自身がメモリ内に展開しているオブジェクトの内容を更新する。

CHMSでは各クライアントプロセス間で直接更新情報を送受信するのではなく、協調活動サーバと呼ばれるプロセスを介して行なわれる。協調活動サーバは更新を行なったクライアントプロセスから更新情報を受取り、それを他のクライアントプロセスに分配する。

実際のオブジェクトの内容の更新はOODBを介して行なわれる。

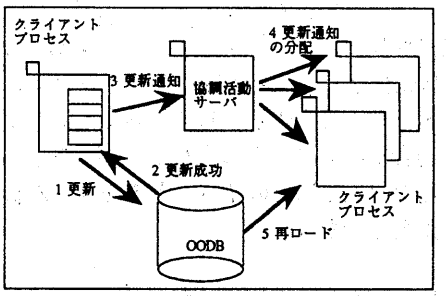

図8. プロセス間の更新伝播

3.2.2 排他制御

オブジェクトの並行操作に不可欠な排他制御は、通常のデータベースにおける排他制御の扱いと同様、ロックによって行われる。

オブジェクトの参照系メソッドが呼ばれるときは、処理の前に実行オブジェクトに対しリードロック(共有ロック)をかけ、更新系のメソッドが呼ばれる時はライトロック(独占ロック)をかける。

クライアントプロセスはメソッドを呼び出すとき、ロック履歴の記録を開始する。そして、メソッドが終了した時点で、それまでにメソッド呼び出しの先頭でかけたロックを(かけた時とは逆の順で)アンロックする。この方法は2相ロックの一種であるから、直列可能性は満足される。

CHMSではメソッド呼び出しの入れ子の順序に従ってオブジェクトのリンクをたどりながらロックをかけていくが、既に他者がオブジェクトにロックをかけている場合、デッドロックを生じる可能性があるため、他者がロックを解除するまでプロセスをウェイトさせるのではなく、メソッドを異常終了させ、メソッド呼び出し前のチェックポイントにおける状態までOODBの状態をロールバックし、全てのロックを解除する。この方法はデッドロックしないのにアボートしてしまう場合があるが、CHMSではユーザにメソッド記述の開放をするため、直列可能性とデッドロックフリーを保障することが重要であると

考え、このような方式を採用している。

3.2.3 マルチカーソル

更新伝播と同様の処理として、CHMSにはマルチカーソルモードがある。マルチカーソルモードでは、あるユーザのカーソルが他のユーザの画面上に表示されカーソルの移動等が伝えられる。CHMSでは通常はシングルカーソルモードでカーソルの移動は通知されないが、その場合あるユーザが行なった変更は更新伝播により他のユーザのクライアントプロセスの画面にも反映されるが、その変化はそのユーザから見た場合突然起こる印象を与える。ホワイトボードのような機能部品を複数ユーザが同時に参照し、皆で図形を書きながら議論するような同期型の作業を行なう場合は、他ユーザの操作過程がある程度見えた方が便利である。

マルチカーソルモードでは各ユーザのマウスの移動以外に領域選択のためのラバーバンドや機能部品移動のためにドラッグ中に表示されるスケルトンが通知の対象になる。

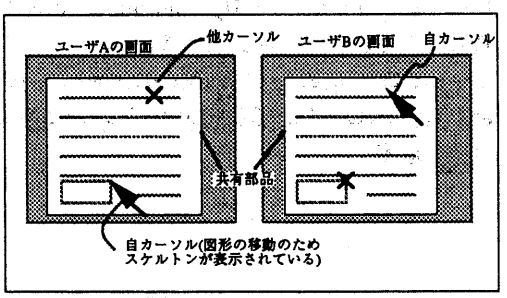

図9. マルチカーソル

カーソルの動きを他クライアントプロセスに伝える方法は更新伝播と同様に協調活動サーバによって行なわれるが、カーソル等の伝達対象は非永続的なものであるためOODBへのアクセスは一切行なわれない。

3.3 利用者インタフェース

3.3.1 機能部品の追加

CHMSでは構造化部品により、新しい部品を作成することができるが、それ以外にエンドユーザがプログラミングできる環境も提供する。ユーザが行えるのは、Pref.のトランスレーションテーブルの変更や構造化部品による機能部品の組み合わせ以外に、部品の定義や機能部品間の関係の記述などがある。

新しい機能部品の定義は、部品を構成するオブジェクトの定義という形で行なわれる。それぞれのオブジェクトを新たに定義する場合、既にあるオブジェクトの子クラスとしてその差分だけを記述する。

CHMSでは更新伝播を自動的に行なうため、更新系のメソッドの処理は実際には

1. オブジェクトの内容を更新する処理
2. OODBへの更新
3. 協調活動サーバへのメッセージ送信
4. 更新伝播処理

という形で行なわれる。ユーザがメソッドを定義する場合、1の通常の処理のみを記述する。2～4の処理についてはCHMS側が自動的に実行する。

3.3.2 関係の記述

ソフトウェア開発支援において、データ間の関係付けも重要である。データを関連付ける機能として各種リンク機能を提供するが、リンクはデータ間の関係を示す機構としては制約が多く、ソフトウェア開発支援などには不十分である。そのため、CHMSではオブジェクト間の関係を直接表現する関係オブジェクトの導入により、関係付け機能を強化する。

オブジェクト間の関係は関係オブジェクトで記述される。関係オブジェクトは対象となるオ

ブジェクトのリストとオブジェクト間の関係を持っている。オブジェクト間の関係としては属性値間の等式、不等式および意味的な関係を考えている。ただし意味的関係については現在未整理の段階であり、現在システム開発(特に上流工程)を調査・モデル化しており、その結果を反映させる予定である。

関係オブジェクトに登録されているオブジェクトSの内容が更新されると制約ソルバと呼ばれる処理系が呼ばれ、Sが属する関係を全て満たしていく。制約ソルバはSが属する関係だけでなく、それを充足することによって新たに生じる関係全てを満たしていく。

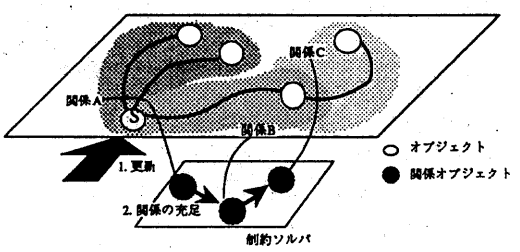

図10. 関係の充足

4. 今後の課題

システム開発業務の支援を目的として構築を進めているCHMSについて、その狙い、特徴および構成を述べた。

今後は、本システムを実際のシステム開発に試行的に適用し、実用可能性の評価を行なっていく予定である。

[参考文献]

[1] 松下、「グループウェアの社会・文化的考察」、情処グループウェア研究会、*93-GW-1*, 1993, pp.1-10.

[2] 石井、「グループウェア技術の研究動向」、情報処理, Vol.30, No12, 1989, pp.1502-1508.

[3] 山上、「長期協調支援におけるシステム特性とグループ特性」、情処グループウェア研究会、*93-GW-1*, 1993, pp.19-26.

[4] 桑名、坂本、「コラボレーションルームの設計とその利用」、情処グループウェア研究会、*93-GW-2*, 1993, pp.33-40.

[5] K. Watanabe, et. al., "Distributed Multiparty Desktop Conferencing System: MERMAID," *Proc. of ACM CSCW'90*, 1990, pp.27-38.

[6] T. Winograd, "A Language Perspective on the Design of Cooperative Work," *Proc. of ACM CSCW'86*, 1986, pp.203-220.

[7] KY. Lai and T. W. Malone, "Object Lens: A "Spreadsheet" for Cooperative Work," *Proc. of ACM CSCW'88*, 1988, pp.115-124.

[8] Y.Tanaka, et al., "A Synthetic Media Architecture for an Object-Oriented Open Platform," *Proc. of the IFIP 12th World Computer Congress*, Madrid, Sep.1992, pp.104-110.

[9] 村永、「ハイパーメディアを用いたグループウェアMuHymeの設計と実現」、情処グループウェア研究会、*92-GW-1*, 1992, pp.59-66.

[10] J. M. Haake and B. Wilson, "Supporting Collaboration Writing of Hyperdocuments in SEPIA," *Proc. of CSCW'92*, 1992, pp.138-146.

[11] R. D. Hill, et. al., "The Rendezvous Language and Architecture," *CACM*, Vol.36, No.1, Jan. 1993, pp.62-pp.67.