M-057

異種端末間でのリアルタイム協調作業支援を可能にするオブジェクト同期手法の提案

A Method of Object Synchronization for Supporting Real-time Collaborative Activities between Heterogeneous Terminals

植田 亘† Wataru Ueda
野口 尚吾‡ Shogo Noguchi
高田 秀志† Hideyuki Takada

1. はじめに

コンピュータによる共同作業を支援するシステムについては，CSCW(Computer Supported Cooperative Work)という分野で多くの研究が重ねられている．また，この分野の一つとして，リアルタイムな協調作業支援に関する研究も行われている．コンピュータ上でリアルタイムな協調作業を行うことは，場所を越え，作業の効率化や創造性の向上が期待できる．しかし，このような協調作業はPCを用いて行われることが多く，携帯端末で利用されることは少ない．

携帯端末には，誰でも手軽に，また場所や時間を気にせずに操作できるという利点がある．本来，PCで利用されていたEメールやWebブラウザなどは，携帯端末で利用できるようになり，より利便性が向上し，有用性が増した．このようにリアルタイムな協調作業を携帯端末上で行うことが可能となれば，利便性が高まり，有用性が増すことが期待できる．しかし，PCと携帯端末では性能が異なるため，協調作業を行うさいに実装の異なるシステムを用いなければならない．端末間の性能差を吸収し，実装の異なるシステム間でも協調作業を行うことを可能にするために，本稿では，オブジェクトの状態に着目した異種端末間でのリアルタイム同期処理手法を提案する．

2. 協調作業支援システム開発基盤 "CUBE"

我々は，協調作業支援システムの開発に必要な機能を提供するJavaを用いた開発基盤 "CUBE(Collaborative Universal Basic Environment)"の構築を行っている．この開発基盤は，複数のノード間でのリアルタイムなシステムの処理の同期を可能にする．また，通信にP2P方式を用いた柔軟性のあるネットワークの構築や，ノードグループの管理などの機能を提供する．

CUBEでは，TeaTime[1]で提案されている複製計算モデルに基づいて，オブジェクトの状態を同期することにより，ノード間で処理を同期する．オブジェクト指向で構築されたシステムにおいて，オブジェクトの状態の変更は，アクセサメソッドなど，何らかのメソッド呼び出しを通じて行われる．そこで，CUBEでは，オブジェクトのメソッド呼び出しを同期することによって，複数のノード間でオブジェクトの状態を同期する．まず，オブジェクトにIDを付与し，これを用いてオブジェクトを管理する．そして，何らかのメソッド呼び出しが行われたさい，そのメソッドを持つオブジェクトのIDと，呼び出されたメソッドを同期対象のシステムに通知する．通知を受け取ったシステムは通知されたIDに適合するオブジェクトから指定されたメソッドの呼び出しを行う．

CUBEによるオブジェクト同期の例として，異種端末間におけるパズルアプリケーションを想定する．図1のように，あるPCアプリケーション上でパズルのピースが移動されると，システムの内部では，ピースオブジェクトが持つピースの位置を変更するメソッドが呼び出される．次に，そのメソッド呼び出しが携帯アプリケーションに通知され，同じメソッド呼び出しが行われることにより，携帯アプリケーション上でもピースの移動が同様に行われる．

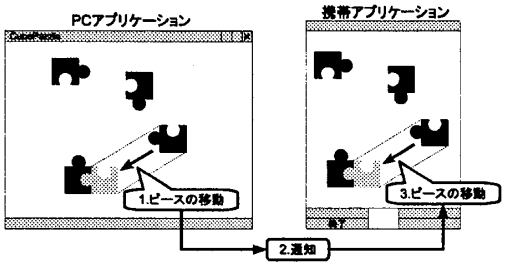

図1: CUBEによるオブジェクト同期の例

3. 異種端末間でのオブジェクト同期手法

PCと携帯端末などの異種端末では端末の性能が異なるため，同じ実装のシステムを動作させることが難しい．しかし，実装が異なっても同じ要求を満たすシステムであれば，これらのシステム上の処理を同期することによって，リアルタイムな協調作業が可能になると考えられる．オブジェクト指向で構築されたシステムであれば，同じ機能，および役割を果たすオブジェクト同士を同期すればよい．

3.1 インタフェースの利用

CUBEでは2章で述べたように，メソッドの呼び出しを同期することによって，オブジェクトの状態を同一に保つ．これを実現するために，JavaのProxyクラスを利用している．このProxyクラスを利用するには，インタフェースを定義しておく必要がある．Proxyクラスは，同期させるオブジェクトに対して，そのインタフェースを実装する代理オブジェクトを生成する．つまり，同期させるオブジェクトとその代理オブジェクトは同じインタフェースを実装している．

代理オブジェクトの機能を図2を用いて説明する．PC上でPieceオブジェクトのsetPosition()が呼び出される場合，実際には代理オブジェクトのsetPosition()が呼び

†立命館大学情報理工学部
‡立命館大学大学院理工学研究科

出される．代理オブジェクトは setPositon() の呼び出しを同期対象のシステムに通知した後，Piece オブジェクトの setPosition() を呼び出す．このように，同期させるメソッドの呼び出しは代理オブジェクトを介して行われる．この代理オブジェクトは実装しているインタフェース型のインスタンスとして扱われる．そのため，同期させるオブジェクトの実装が異なっても，代理オブジェクトは共通のインタフェースを持つ．これを利用すれば，異種端末間においても，実装の違いを意識せずにオブジェクトのメソッド呼び出しを同期することができる．

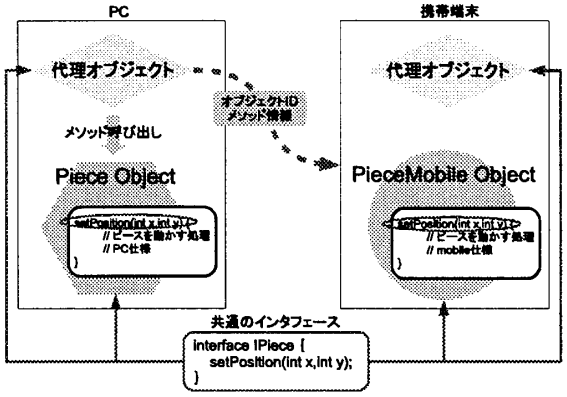

図 2: メソッドのインタフェースの統一

3.2 メソッド呼び出しの通知

メソッド呼び出しの同期は図 2 のように，オブジェクト ID とメソッド情報を同期対象のシステムに通知することで行われる．メソッド情報とは，そのメソッドに対応する Method オブジェクトや，引数および引数の型である．これらの情報を持つオブジェクトを，シリアライズして送信する．このとき，前節でも述べたように，メソッドのインタフェースが同一であるため，同期対象のシステムでは実装の違いに関係なく通知されたメソッドを呼び出すことが可能である．

3.3 オブジェクトの生成

CUBE では，新たなオブジェクトが生成されたさい，生成されたオブジェクトのクラスと ID を同期対象のシステムに通知する．そして，同期対象のシステム上で，通知されたクラスからオブジェクトを生成し，同じ ID を付与する．実装の同じシステム間ではこのようにして，オブジェクトの生成と ID の付与を行うことができる．しかし，システムの実装が異なる異種端末間では，実装が異なるオブジェクトを同期しなければならない場合がある．

実装の違うオブジェクトはクラス定義が異なるため，クラスと ID を通知するだけではオブジェクトの対応が取れず，オブジェクトの生成と ID の付与を行うことができない．そこで，同様の機能を持つクラスを表形式で対応づけた，クラス対応表を各ノードに持たせる．新たなオブジェクトが生成されたさい，図 3 のように，まず生成されたオブジェクトの情報を同期対象のシステムに送信する．オブジェクトの情報を受け取ったシステムは，受け取った情報をクラス対応表と照合し，対応するクラスのオブジェクトを新たに生成する．そして生成されたオブジェクトに通知された ID を付与する．これによって，実装の違うオブジェクトも同様に扱うことが可能になる．

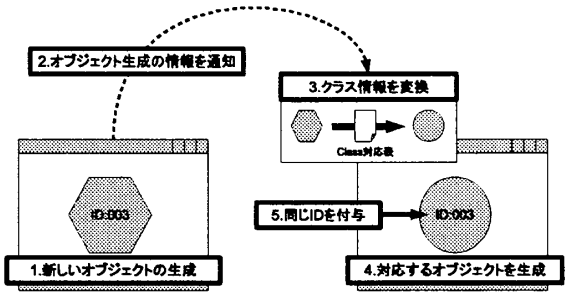

図 3: オブジェクト生成の通知

4. 適応例

本手法の適応例として，教育現場で使用できる作業支援システムが考えられる．複数の学習者が一つの高解像度大スクリーンの表示物を携帯端末をリモコンのように使用し，共同で編集する．これによって，従来の PC を介しての協調作業よりも学習者間の物理的距離が接近し，コミュニケーションが取りやすくなり，作業効率が向上するといったような効能が期待できる．

このように，本手法は異種端末間の同期を可能とすることにより，PC と携帯端末，各々の特徴をより生かした協調作業支援システムの構築を可能にする．

5. おわりに

本稿では，携帯端末や PC を含む異種端末間でのシステムの処理を同期する手法を提案した．オブジェクトを ID で管理し，オブジェクトのインタフェースを共通にすることで，端末間の性能差を吸収し，システムの処理を同期することができる．これにより，異種端末が混在するネットワークにおいてもシステムの処理の同期を行うことが可能となる．本手法を用いれば，より場所や端末の性能に縛られない自由な協調作業支援を行うことが可能となり，有用性が高まることが期待できる．

今後は，携帯端末の利用に起因する問題点であるネットワーク接続の管理や通信速度による遅延について考察していく．その後，携帯端末と PC 間でのサンプルアプリケーションの開発，および，それらの検証実験や既存の研究との対比を進めていく．

謝辞

本研究を進めるにあたり，有益なご助言を頂きました立命館大学情報理工学部島川教授および研究室の方々に感謝いたします．

参考文献

[1] David P. Reed, “Designing Croquet’s TeaTime - A Real-time, Temporal Environment for Active Object Cooperation”, OOPSLA 2005, 2005.