Vol.2009-GN-72 No.1
2009/5/21

複製計算に基づく協調システム基盤 CUBE の構築と異種端末環境への適用

植田 亘[†1] 野口尚吾[†1] 高田秀志[†2]

複数のユーザ間でのデータ共有や，コミュニケーションの円滑化を支援するアプリケーションは，グループワークなどの協調作業を支援するシステムとして有用である．このようなシステムを開発するさい，システムが取りうるアーキテクチャは数種類考えられる．著者らが開発している Java による協調システム基盤は，複製分散型のアーキテクチャを採っている．この開発基盤の提供する特徴的な機能として，オブジェクトミラーリングが挙げられる．これは，各アプリケーション間で各々のアプリケーションが保持するオブジェクトのふるまいを同期するものである．開発者は，この機能を利用することによって，例えば，あるユーザがアプリケーション上で行った操作に対する処理を他端末上で複製することができる．

本稿では，まず，この開発基盤とオブジェクトミラーリングの仕組みを述べ，さらに，オブジェクトミラーリングを携帯端末と PC 端末などの異種端末間においても可能とする手法について述べる．

A Collaborative System Framework CUBE Based on Replicated Computing and Its Application to Support Heterogeneous Environment

Wataru Ueda,[†1] Shogo Noguchi[†1]
and Hideyuki Takada[†2]

Collaborative applications are helpful to support group work. We are working on the Java development framework for systems supporting collaborative activities. This framework is based on a distributed replication architecture which is a kind of architectures for such systems. One of the most characteristic functions is Object Mirroring. Object Mirroring synchronizes object behavior among different nodes. For example, the developer can build a collaborative application which enables users to share their operations among terminals.

This paper explains this development framework focusing on Object Mirroring and the method which realizes it among heterogeneous terminals.

1. はじめに

ユーザ間での情報共有や，コミュニケーションを支援するシステムは，コンピュータを用いた協調作業やグループワークなどに有用であり，さまざまな目的の CSCW アプリケーションが存在する．このようなシステムを開発するさいには，開発者はネットワーク通信，ノード管理，ノードのグループ管理などの多様な機能を開発しなければならない．著者らの開発している，協調システム基盤 “CUBE(Collaborative Universal Basic Environment)” は，このような協調支援システムの基本的な機能を提供することを目的としている．

CUBE の提供する特徴的な機能として，オブジェクトミラーリングが挙げられる．オブジェクトミラーリングは，TeaTime[1)] で提案されている複製計算に基づき，各ノード上にオブジェクトの複製を生成し，それらのオブジェクトのふるまいを同期する．CUBE では，Java オブジェクトのメソッド呼び出しを複製することで，オブジェクトミラーリングを実現する．オブジェクトミラーリングを用いると，各ノード上のシステムのデータ変化やユーザ操作に対する処理を，他ノードにおいて複製することが可能となる．さらに，CUBE はオブジェクトミラーリングを PC と携帯端末などの異種端末間においても実現する．一般に，PC 端末と携帯端末にはさまざまな差異が存在する．CPU の処理能力やメモリ容量などの端末の性能の差異，および入力デバイスなどの違いによるユーザインタフェースの差異などがこれにあたる．また，Java のように，アプリケーションが動作する端末ごとに最適な機能セットが用意されている場合がある．そのため，異種端末間では，各端末上のシステムの実装は，その端末に適した実装であり，他端末のものとは異なっている．つまり，異種端末間でのオブジェクトミラーリングを可能とするためには，各端末の実装の差異，つまり各端末上のシステムを構成するオブジェクトの実装の差異を吸収する仕組みを構築することが必要である．

本稿では，CUBE において，複製計算を実現する仕組みと，それを異種端末間で実現するための手法について述べる．

†1 立命館大学大学院 理工学研究科
Graduate School of Science and Engineering, Ritsumeikan University
†2 立命館大学 情報理工学部
College of Information Science and Engineering, Ritsumeikan University

2. 協調システム基盤 CUBE

協調システム基盤 CUBE は Java を用いて構築され，さまざまな OS 上で動作可能な汎用性の高いフレームワークである．本節では，まず，CUBE の全体像について述べる．その後，CUBE の最も特徴的な機能であるオブジェクトミラーリングについて説明し，最後に，CUBE を用いて構築した協調学習支援アプリケーション “Snow Boy[2)]”を紹介する．

2.1 CUBE の構成

CUBE は図 1 のように，

- オブジェクトミラーリング
- グループサービス
- ネットワークレイヤ

の 3 つの要素から構成される．オブジェクトミラーリングは各ノードが保持するオブジェクトのふるまいを同期する機能である．グループサービスは，オブジェクトミラーリングが反映される範囲を階層構造を持ったグループで管理することを可能とする．そして，ネットワークレイヤは，さまざまな通信プロトコルを隠蔽し，ノード間での P2P 通信を可能とする．

CUBE のアーキテクチャは複製分散型である．そのため，ネットワークトラフィックが小さく，それぞれのノード上のアプリケーションから実行結果を受け取ることが可能になり，アプリケーションの応答時間が短くて済む．

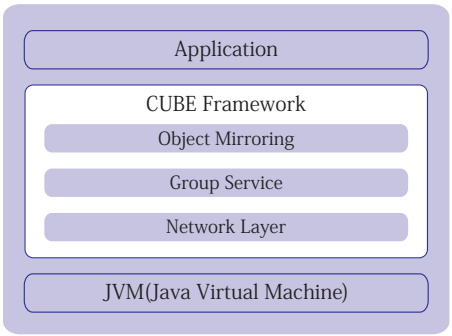

図 1 CUBE の構成

2.2 オブジェクトミラーリング

オブジェクトミラーリングは，複数のノード間でオブジェクトのふるまいを同期するものであり，TeaTime で提案されている複製計算モデルに基いている．複製計算では，各ノードはそれぞれ同じオブジェクトの複製を保持する．そして，あるノード上でオブジェクト間のメッセージパッシングが起こると，それが他ノードに伝播され，同じメッセージパッシングが各ノード上で発生する．このように，メッセージパッシングを共有することで，各ノードのオブジェクトのふるまいを同期する．TeaTime を用いたシステムとして Croquet[3)] がある．Croquet は複製計算モデルに基づいてユーザの描画処理などのアプリケーションの処理をリアルタイムに共有することができる．

CUBE では，オブジェクトミラーリングの対象となるオブジェクトをミラーオブジェクトと呼ぶ．そして，Java 上で，このミラーオブジェクトの生成とメソッド呼び出しを複製することによって，複製計算を実現する．

2.3 オブジェクトの生成の複製

オブジェクトミラーリングの仕組みを図 2 に示す．CUBE では，ミラーオブジェクトを ObjectManager クラスを用いて管理する．ミラーオブジェクト生成メソッドは，ミラーオブジェクトと代理オブジェクトを生成し，その参照を返す (この代理オブジェクトは，メソッド呼び出しの複製を実現のために利用するものであり，3.3 節で説明する)．その後，ミラーオブジェクトをそのオブジェクトを識別するためのオブジェクト ID と対応づけて ObjectManager に登録する．このとき，さらにミラーオブジェクトの生成情報と ID をネットワークを通して他ノードへ通知する．他ノードの ObejctManager はこの通知を受け取ると受け取った情報を元に，ミラーオブジェクトと代理オブジェクトを生成し，ミラーオブジェクトに通知された ID を付与し，管理する．これにより，各ノードは同じ状態のミラーオブジェクトの複製を保持する．

2.4 代理オブジェクトを用いたメソッド呼び出しの複製

CUBE では，メソッド呼び出しを複製するために，Java が提供するリフレクション機能の Proxy クラスを利用している．この Proxy クラスを利用し，代理オブジェクトを生成する．この代理オブジェクトは，ミラーオブジェクトと一対一対応しており，対応するミラーオブジェクトと同じインタフェースを実装する．代理オブジェクトのメソッドが呼び出されたとき，Proxy クラスの機能により，呼び出されたメソッドの情報 (メソッド名，引数，引数の型) が InvocationHandler に渡される．InvocationHandler は，呼び出されたメソッドの情報を他ノードに通知する．これにより，あるノード上で起こったメソッド呼び出しが，

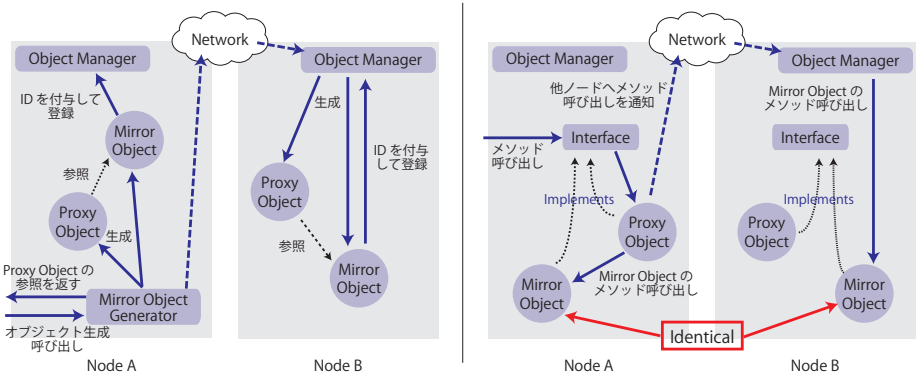

図 2 オブジェクトミラーリングの仕組み

他ノード上でも複製される．

2.5 CUBE を用いたプログラミング

これまでに述べたように，オブジェクトミラーリングはミラーオブジェクトの代わりに，代理オブジェクトのメソッド呼び出しを行うことで実現される．開発者は，プログラミングのさいに，CUBE が提供するメソッド (SynchroObjectProxy.newInstance()) を用いて，ミラーオブジェクトと代理オブジェクトを生成する．図 3 は，CUBE を用いたプログラミングの例である．代理オブジェクトは，ミラーオブジェクトと同じインタフェースであるため，開発者はミラーオブジェクトを扱うのと同じ要領で代理オブジェクトを扱うことができるため，オブジェクトミラーリングに必要な処理を意識せずにプログラムを書くことができる．

2.6 CUBE の利用例

CUBE を利用して構築したアプリケーション例として SnowBoy を紹介する．SnowBoy は，初等教育においてコンピュータを用いた協調学習を支援するためのソフトウェアである．児童は複数人でグループを作り，共同して 1 つの作品を作ることができる．作品は，まず 3D グラフィックスのオブジェクトを制作し，そのオブジェクトに対してプログラミングを行うことで，オブジェクトに対して動きを与えることができる．SnowBoy において，児童が各々操作する PC 上には，オブジェクトを描画するための「オブジェクト描画領域」，プログラムを編集するための「プログラミング領域」などが存在する．これらの空間は，CUBE を用いて同期されている (図??)．これにより，児童は自分用の PC を操作しながら，ひと

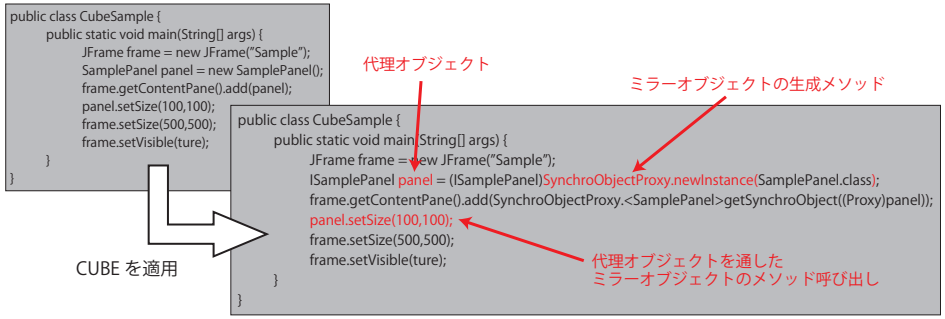

図 3 CUBE を用いたプログラミング例

つの作品を他の児童と協力しながら創作することができる．

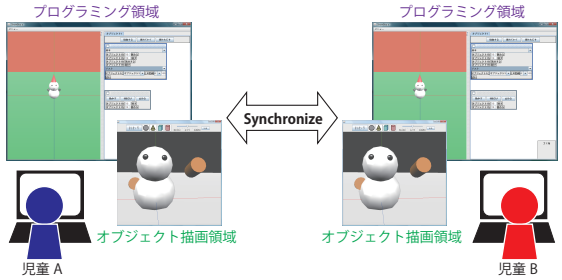

図 4 SnowBoy における作業空間の同期

3. 異種端末環境への適用

CUBE のオブジェクトミラーリングは，異なる端末上で動作しているアプリケーション間で，それぞれのアプリケーションがミラーオブジェクトを保持し，そのミラーオブジェクトのふるまいを同期するものである．しかし，異種端末が混在し，各々の端末上のシステムがその端末に適した実装方式を採る必要がある場合には，オブジェクトミラーリングをそのまま適応することができない．これは，端末ごとに適した実装を行うことによって，アプリケーション内のオブジェクトの実装が異なるからである．ここで，もし，オブジェクトの実装が異なっても，アプリケーションの中で果たす役割は同じであり，ただその実装方法が

違う場合，それらのオブジェクトをミラーリングする対象とすることができれば，利用端末に適した実装のアプリケーションを用いる協調作業支援システムを構築することができる．本節では，オブジェクトの実装の違いを吸収し，システム内で同じ役割を果たすオブジェクト同士をミラーリングする手法について述べる．

3.1 実装が異なるオブジェクトのメソッド呼び出しの複製

2 節で述べたように，ミラーオブジェクトのメソッド呼び出しは，その代理オブジェクトを介して行われる．図 5 はひとつのミラーオブジェクトとその代理オブジェクトの組がある機能を果たすと考えた場合の, 両者の関係を示す図である．このとき，代理オブジェクトはその機能のインタフェース，ミラーオブジェクトはその機能の実装と考えることができる．この場合，各々のアプリケーションでそれぞれの代理オブジェクトを同様に扱えるなら，この機能のインタフェースは同じであると言える．つまり，機能の実装であるミラーオブジェクトの実装は隠蔽され，同じ役割を果たす機能を，実装の異なるアプリケーション間でも，同じインタフェースで扱うことができる．

CUBE において，代理オブジェクトは，自身と対応するミラーオブジェクトと同じインタフェースを実装する．また，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，参照される．そのため，各ノード上のミラーオブジェクトが同じインタフェースを実装していれば，それぞれの代理オブジェクトも同じインタフェースを実装することになる．これは，各々の代理オブジェクトが同じインタフェースを実装することを指し，各々のアプリケーションは，代理オブジェクトを同様に扱うことができる．つまり，各々のノードに存在するミラーオブジェクトのインタフェースが同じなら，アプリケーションに対して，その実装は隠蔽され，同じ役割のオブジェクトを同様に扱うことができる．

代理オブジェクトが，ミラーオブジェクトのメソッド呼び出しを他ノードへ通知するさい，他ノードへ通知する情報は，

- オブジェクトの ID
- 呼び出されたメソッドの名前
- メソッドの引数
- 引数の型

である．オブジェクトは ID で判別され，メソッド呼び出しに使われる情報は，すべてメソッドのインタフェースに関する情報であるため，メソッドの実装には関わらない．これを利用すると，各端末に適した実装のオブジェクトのメソッドの呼び出しを複製することができる．

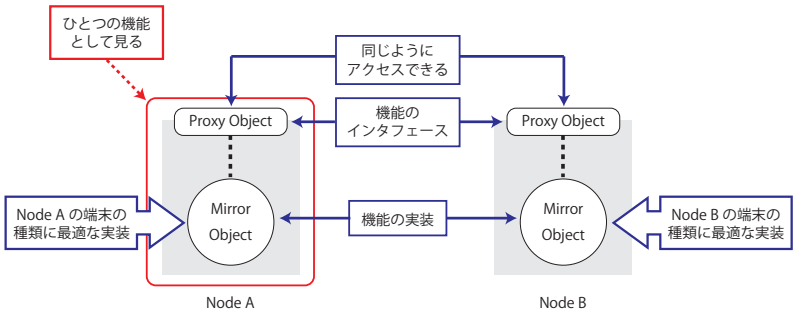

図 5 ミラーオブジェクトとその代理オブジェクトの関係

3.2 実装が異なるオブジェクトの生成の複製

オブジェクトミラーリングにおいて，実装の異なるオブジェクトの生成を複製する場合を考える．

あるオブジェクトが生成されたとき，その生成を他ノードに通知し，同じ役割のオブジェクトを生成しなければならない．このとき，生成すべきオブジェクトの情報としてクラス名を他ノードに通知しても，他ノードでは，その役割を果たすオブジェクトのクラスは違うものであるので，オブジェクトを生成することができない．そのため，ミラーリングしたいオブジェクトの実装が異なる場合は，まず，各々の端末の異なるクラス同士の名前を対応付ける必要がある．

そこで，各端末に実装の違うクラスの対応を表すクラス対応表 (Class Map List) を持たせる．このクラス対応表は，

- 端末のタイプ
- 端末ごとの使用クラス名

を保持している．各ノードは，オブジェクト生成のメッセージを受け取ったときに，クラス対応表を参照し，通知されたクラス名を自ノードに対応したクラス名へと変換する．そして，変換後のクラス名をもとに新たなミラーオブジェクトを生成する．そのさい，生成されたミラーオブジェクトにそれぞれ共通の ID を付与し，以後 ID を用いて扱うことを可能にする．

3.3 クラス対応表の実装

クラス対応表を用いたクラス変換のフローを図 6 に示す．まず，Message Receiver は他ノードからオブジェクト生成のメッセージを受け取る．次に，Message Translator が，自身の端末のタイプ，およびクラス対応表を参照して，通知されたクラス名を自身の端末に適

したクラス名に変換する．そして，変換後の情報を元に Mirror Object Generator によってミラーオブジェクトとその代理オブジェクトが生成され，ID と共に Object Manager によって管理される．

このクラス対応表は，アプリケーションを開発するさいに，開発者が作成する必要がある．そのため，計算機が処理しやすいフォーマットであることに加えて，開発者の負担を増加させないためにも，人が理解しやすいフォーマットであることが必要である．そこで，クラス対応表には XML を用いる．図 7 にクラス対応表のフォーマットを示す．クラス対応表は，terminal と message という要素を持つ．まず，terminal 要素の中で，変換が必要な端末の種類を定義する．

図 7 の例では，pc，netbook，mobile の 3 つの端末が定義されている．次に，message 要素中において，クラスの対応を定義する．class タグの属性に最初に定義した物の中から適切な端末タイプを設定し，そのコンテンツとして対応するクラス名を保持する．たとえば，図 7 の最初の message 要素では，自身の端末のタイプが pc の場合は PCObject，netbook の場合は NetbookObject，mobile の場合は MobileObject というクラスを使用することを定義している．

XML 形式で書かれたクラス対応表は，アプリケーションが起動されるさいにパースされ，各端末のメモリ上に格納される．図 8 は，図 7 の XML をパースした場合の例で，自身の端末のタイプが mobile であった場合のメモリ上のテーブルを表している．他ノードから受信したクラス名をキーとして検索すると，自身の端末タイプに対応するクラス名が得られる．

4. 異種端末間でのアプリケーション例と評価

4.1 パズルアプリケーションの例

本稿の手法を用いた異種端末間で動作するアプリケーション例として，スライドパズルアプリケーションを実装した．図 9 は，そのスクリーンショットである．携帯端末版は J2ME の CDC で開発した．また，エミュレータに Sun が提供している，Sun Java Toolkit for CDC 1.0 を使用した．

このアプリケーションでは，携帯端末用と PC 端末用でユーザインタフェースが異なっている．携帯端末版では，ユーザはカーソルキー用いて，青色のカーソルフレームを動かす．そして，動かしたいピースにカーソルをあてた状態で決定キーを押すと，ピースがブランクスペースへスライド移動する仕組みになっている．一方 PC 端末では，ユーザはマウスを用いて，パズルのピースをクリックして選択し，移動させる．

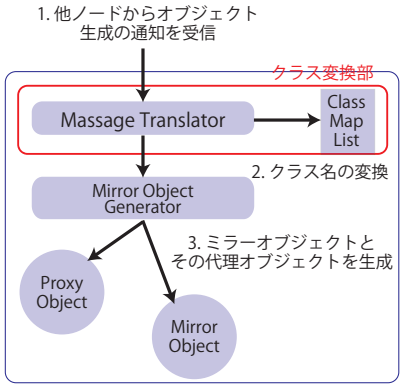

図 6 クラス情報変換の処理フロー

```
<?xml version="1.0" encoding="UTF-8"?>
<ConvertList>
    <terminal>
        <type>pc</type>
        <type>netbook</type>
        <type>mobile</type>
    </terminal>
    <message type="class">
        <class type="pc">PCObject</class>
        <class type="netbook">NetbookObject</class>
        <class type="mobile">MobileObject</class>
    </message>
    <message type="class">
        <class type="pc">3DObject</class>
        <class type="netbook">3DObject</class>
        <class type="mobile">2DObject</class>
    </message>
    <message type="class">
        <class type="pc">FullOject</class>
        <class type="netbook">HalfObject</class>
        <class type="mobile">HalfObject</class>
    </message>
</ConvertList>
```

図 7 Class Map List のフォーマット

Key	Value
PCObject	MobileObject
NetbookObject	MobileObject
MobileObject	MobileObject
3DObject	2DObject
3DObject	2DObject
FullObject	HalfObject
HalfObject	HalfObject

図 8 メモリ上に格納されたクラス対応表

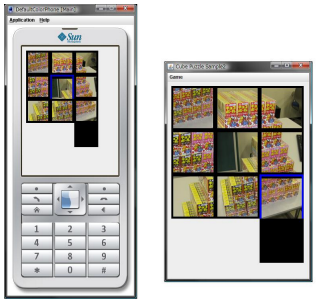

図 9 異種端末間で動くパズルアプリケーション

アプリケーションの実装では，パズルのピースが各々ミラーオブジェクトとなっており，Piece オブジェクトが持つ，setLocation() メソッドの呼び出しを複製している．

このように，本稿の手法を用いることで，ユーザが各端末に適した実装のアプリケーションを用いて，協調作業を行うことができる．開発者はアプリケーションを開発するさいに，

- ミラーリングするオブジェクトの選択
- ミラーオブジェクトの各端末での実装

を適切に判断することで，異種端末混在環境において動作する協調作業支援システムを構築することができる．

4.2 関連研究

分散プログラミングを支援するフレームワークとして SOR がある[4]．SOR では，オブジェクト共有空間を定義し，そこに各ノード上のオブジェクトを共有オブジェクトとして格納することで，複数の JVM 間での仮想的なオブジェクト共有と格納されたオブジェクトの永続化を実現する手法が提案されている．しかし，このオブジェクト共有空間では，格納されたオブジェクトの複製されない．つまり，実際には，共有オブジェクトは 1 つのノードにしか存在しない．そのため，ノードの状態が頻繁に変化し，ノードが断線する可能性がある P2P ネットワークには適していない．CUBE では，各ノードに同じ状態のオブジェクトが配置されるため，他ノードの状態の変化による影響が非常に少ない．

また，携帯端末での利用もサポートし，P2P 技術を用いたリアルタイムな協調作業の支援を行うシステムのフレームワークとして，SOBA フレームワークがある[5)6)]．SOBA フレームワークでは，コンピュータネットワーク上にセッションと呼ばれる情報共有のための仮想空間を生成する．セッションは分割，統合など，柔軟に変化させることが可能である．また，携帯端末からのセッションへの参加を可能とするために，携帯端末用のプロキシサーバ (ゲートウェイ) を設けている．このゲートウェイは，端末から受け取ったデータを各々の端末向けに整形・変換，さらに処理の一部を負担することにより，PC 端末と携帯端末間での仮想的な P2P 通信を可能としている．さらに，SOBA はユーザが多様なメディア (映像、音声、アプリケーション画面やテキストなどのデータ) を用いて，多様なコミュニケーションを行うことに焦点を当てたフレームワークである．また，SOBA は拡張された MVC モデルに基づくイベント同期によって Model を共有するもであり，オブジェクトを複製する CUBE とは異なっている．

5. おわりに

本稿では，協調システム開発基盤 CUBE の構成と，その代表的な機能であるオブジェクトミラーリング，さらにオブジェクトミラーリングを異種端末間で実現するための手法について述べた．CUBE では，各ノードのオブジェクトのメソッド呼び出しを複製することで，オブジェクトのふるまいを同期する．さらに，実装とインタフェースの分離によって，各端末に適した実装のオブジェクトの状態を同一に保つことができる．また，クラス対応表の利用により，各端末に適した実装のオブジェクトを異種端末の各ノード上に同様に配置することが可能である．これらを利用することにより，各ノードでのユーザの操作の共有や，データの同期を可能し，さらに異種端末混在環境でも使用できるアプリケーションを構築することができる．

今後は，オブジェクトミラーリングの順序付けを行う一貫性保持や，ノードの途中参加，またミラーオブジェクトの永続化などの機能を追加していく予定である．さらに，携帯端末などの端末において，処理能力の低さや，通信プロトコルの違いによる処理の遅延などが起こった場合，それを吸収し，システム全体に影響を与えずに利用可能とすることも今後の課題である．

参考文献

1) David P. Reed, “Designing Croquet’s TeaTime - A Real-time, Temporal Environment for Active Object Cooperation”, OOPSLA 2005, 2005.
2) 柿内 達真, 取越 翔太郎, 桜打 彬夫, 大東和 忠幸, 野口 尚吾, 高田 秀志, “SnowBoy: 教室内でのプログラミング作品共有による共同創作が可能な初等教育向け協調学習支援システム”, 第 72 回グループウェアとネットワークサービス研究発表会, GN72-1, 2009.
3) Hideyuki TAKADA, “A 3D Collaborative Creation Environment with Tile Programming on Croquet”, c5, pp. 125-130, Fifth International Conference on Creating, Connecting and Collaborating through Computing (C5 ’07), 2007.
4) 前田直人, 上田和紀, “オブジェクト共有空間を利用した分散プログラミング支援フレームワーク SOR, ”情報処理学会第 58 回全国大会, No.3N-2, 1999.
5) 林 良生, 角田 誠, 篠田 直樹, 中島 玲二, “SOBA フレームワークにおける P2P ネットワーク上の同期機構の実現”, 情報処理学会 65 回全国大会, No.3A-2, 2003.
6) 緒方 敏博, 角田 誠, 玉垣 裕, 中島 玲二, “SOBA フレームワークにおけるモバイル端末を利用するための構造”, シンポジウム「ケイタイ・カーナビの利用性と人間工学」, 2005.