WhiteDog – アスペクト指向に対応したネットワーク共有アプリケーション支援システム
（契約件名:XML 通信を基盤としたオブジェクト共有・オーナー管理システム）

WhiteDog – Network-shared application supporting system corresponding to Aspect-Oriented

中口 孝雄 [1)]　和田 健之介 [2)]　熊坂 妥仁 [3)]
Takao NAKAGUCHI　Kennosuke WADA　Yasuhito KUMASAKA

1)（有）アントラッド（〒630-8101　奈良県奈良市青山４丁目４番地１２３）　E-mail: takao@untrod.keihanna.ne.jp）
2)（有）アントラッド（〒630-8101　奈良県奈良市青山４丁目４番地１２３）　E-mail: kwada@hera.eonet.ne.jp）
3)（有）アントラッド（〒630-8101　奈良県奈良市青山４丁目４番地１２３）

ABSTRACT. This software development project is supported by the Exploratory Software Project. The product of fruits provides an application framework which steps into application level more than the other existing frameworks and a toolkit which enables developers to apply this product to their software systems. Using this product, developers could reduce the cost of programming Network-shared application.

１．背景

インスタントメッセージ、オンラインゲーム、共同作業対応 CAD など、現在たくさんのネットワーク共有アプリケーションが開発され普及している。ブロードバンド時代の到来に伴ってその利用範囲はさらに拡大し、様々な機能を持ったシステムが登場することが予想される。

一方で、これらネットワーク共有アプリケーションの開発には莫大なコストがかかる。これは、インターネットが日常的に利用される現在に至っても、未だネットワークプログラミングが高度で複雑なものであるからである。

現状でも多くのネットワーク共有アプリケーションや、それを支援するネットワーク・プログラミング・ライブラリが開発されており、既に公開されているライブラリもいくつか存在する。しかしいずれも汎用性に限界があり、依然として開発コストは高い。

筆者も今まで、様々な分野のネットワーク共有アプリケーションを開発してきた。クライアントとサーバのソケット操作、送信データ構築、受信データ解析といった通信部分を、毎回一から設計・開発した。通常それは当然のことだが、繰り返し開発する中で、通信部分の実装コストを削減するミドルウェアの必要性を感じるとともに、その部分の高い汎用性も強く認識した。

２．目的

本プロジェクトでは、筆者が、様々な分野のネットワーク共有アプリケーションを開発することで得られた経験とノウハウを結集して、既存のものよりも汎用性が高く、よりアプリケーションレベルに踏み込んだ支援システムを開発し、ネットワーク共有アプリケーションの開発コストを削減することを、その目的としている。

３．成果物

上記の目的を達成するため、本プロジェクトでは、次の４つのソフトウェアを開発した。

（１） 汎用 XML セッションサーバ
Java で実装された、汎用のマルチセッションサーバ。通信データのフォーマットに XML を採用した。複数のクライアント間で作成される、仮想的なセッションを管理する。拡張性に優れ、様々なサービスをプラグインできる。また、このサーバにアクセスするための Java 用、Delphi/C++Builder[1]用クライアントライブラリも開発した。Macromedia Flash[2]からこのサーバを利用することも出来る。

（２） オブジェクト共有システム
Java 言語向けに実装。オブジェクト共有モデルを導入して、通信プロトコルを意識せずに多人数参加型システムを構築することを可能にする。Java のリフレクション機能などを利用している。汎用 XML セッションサーバのクライアントライブラリとして提供される。

（３） オーナー管理システム

---

[1] Borland 社の開発ツール。http://www.borland.co.jp/
[2] http://www.macromedia.com/jp/software/flash/

Java 言語向けに実装。同じセッションに参加しているクライアントの中に、一つのオーナークライアント設け、その動的な移動を管理する。これにより、オブジェクトのロックや負荷の移動、分散といった機構の実装が容易になる。汎用 XML セッションサーバの拡張サービス、クライアントライブラリとして提供される。

（4） アスペクト指向プログラミング支援ライブラリ
Java 言語向けに実装。アスペクトを記述することで、オブジェクト共有システムへの対応部分を実装することができる。AspectJ[3]に対応した、クライアントライブラリとして提供される。

4．オブジェクト共有モデル

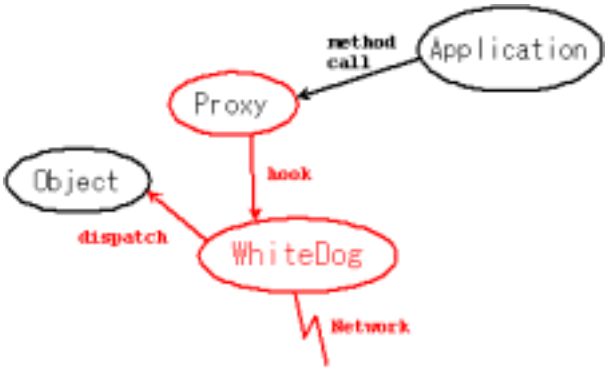
本プロジェクトでは、よりアプリケーションレベルに踏み込んだ機能を提供するため、オブジェクト共有モデルを作成し、導入した。これは、次に上げる制約を満たすオブジェクトに対して、その状態を共有する機能を提供するものである。

- メソッドを実行するタイミングが変化しても、状態が変化しない。
  I. 生成(1)され、あるメソッドが呼び出された(2)後のオブジェクトの状態は、(1)から(2)までに経過した時間に左右されない。
  II. あるメソッドが呼び出され(1)、別あるいは同じメソッドが呼び出された(2)後の状態は、(1)から(2)までに経過した時間に左右されない。
- 戻り値を持つメソッドでは状態は変化しない。

これらの条件を満たすオブジェクトに対して、メソッド呼び出しを記録し、同じ順番で再生すれば、そのオブジェクトの状態を復元することができる。本システムではこの性質を利用し、ネットワークを介してこれを行うことで、オブジェクトの状態共有を実現する。

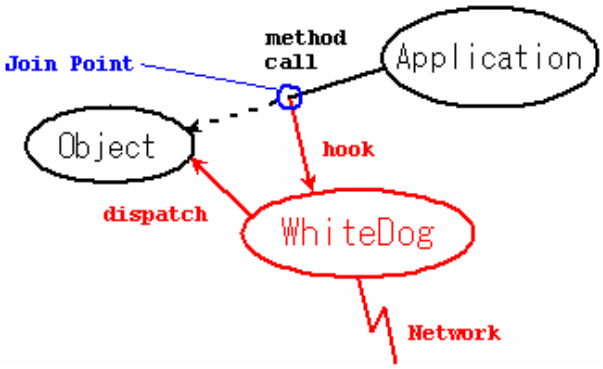
オブジェクト共有システムにおいては、代理オブジェクト(java.lang.Proxy)によるメソッドフックと、リフレクション機能(java.lang.reflection)によるメソッド再生を行う。代理オブジェクトによりフックされたメソッドは、XML 形式にエンコードされ、一旦サーバへ送られる。その後セッションに参加しているクライアント全てに配信され、クライアントでは、XML データをデコードしてメソッドを再生する。一旦サーバを介すことにより、オブジェクトへのメソッド実行順が全てのクライアントで同じになるため、複数のクライアントでオブジェクトが共有できる。

[3] Xerox PARC で開発され、現在は eclipse プロジェクトへ移管されている。http://www.eclipse.org/aspectj/



5．アスペクト指向プログラミング支援ライブラリ

代理オブジェクトを使う代わりに、アスペクト指向プログラミングツールである AspectJ を使ってオブジェクトのメソッド呼び出しをフックすることにより、オブジェクト共有を実現する。本プロジェクトでは、AspectJ とオブジェクト共有システムを接続するライブラリ、一般的なオブジェクト共有処理を実装した汎用基底アスペクトを提供する。この方法では、代理オブジェクトを使う方法と比べ、同期対象メソッド選択方法の柔軟性が得られる。



次に、神経衰弱アプレットに対して本システムを適用した際のアスペクトコードを示す。

```
-- ShinkeiSuijakuAspect.aj --
import java.applet.Applet;
import jp.whitedog.aspect.AspectContext;

public aspect ShinkeiSuijakuAspect extends
AbstractAppletAspect{
  private static final String APP_NAME =
"object.ShinkeiSuijaku";

  //アプレット初期化前にセッション、アプレットを登録する。
  before(): execution(
    void ShinkeiSuijaku.init()
  ){
    applet_ = (Applet)thisJoinPoint.getTarget();
    registerObject(APP_NAME, applet_);
  }

  // アプレット開始時に接続してアプレットの共有を開始する
  after(): execution(
    void ShinkeiSuijaku.start()
  ){
    connect(applet_.getCodeBase().getHost(),
12539);
  }

  // アプレット停止時に接続を解除
```

```
  after(): execution(
    void ShinkeiSuijaku.stop()
  ){
    disconnect();
  }

  // 同期対象メソッド呼び出しポイントのオーバーライド
(*1)
  pointcut concernedMethodsExecution():
    execution(
      void ShinkeiSuijaku.mouseReleased(..)
    ) || execution(
      void ShinkeiSuijaku.mousePressed(..)
    );

   Applet applet_;
}
```

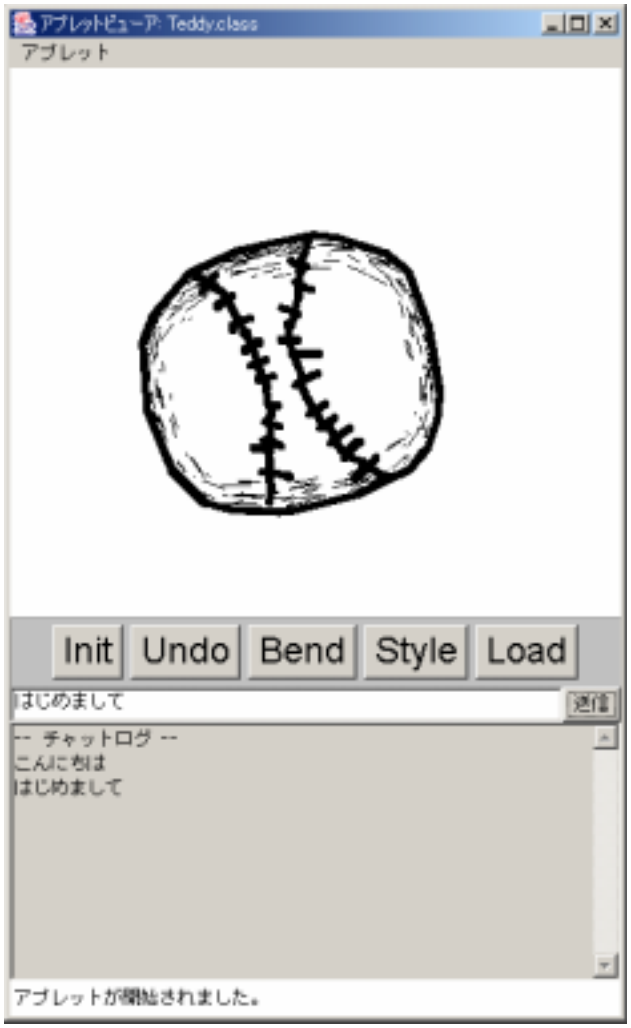
---

*1 では、基底アスペクトで定義された抽象ポイントカット concernedMethodsExecution をオーバーライドして、オブジェクトの同期対象メソッドを指定している。基底アスペクトでは、ここで指定されたメソッドの呼び出し時に、パラメータを XML 形式にエンコードして、サーバへ送信する。

6．適用例

システムの有効性を示すため、いくつかのアプリケーションにこのシステムを適用した。

（1） 五十嵐氏の Teddy[4]

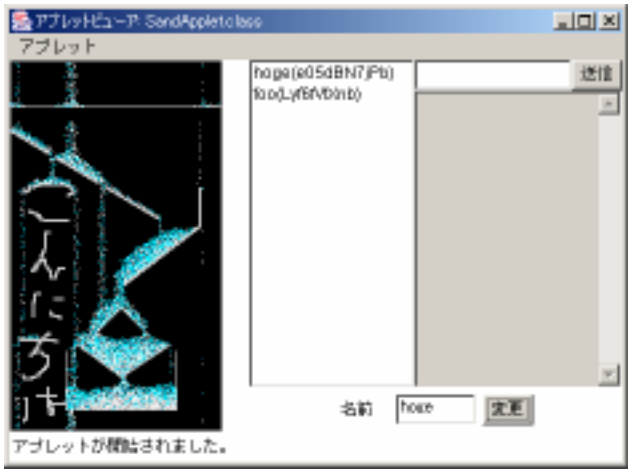


ネットワーク共有化に加え、アスペクトコードによりチャットパネルを挿入した。チャットパネルは、オブジェクト共有システムを使ってメッセージを共有するように作られた部品。

（2） 藤原氏のナンバープレイス[5]

ネットワーク共有化 + チャット UI + オーナー管理。オーナー管理 UI の付いたチャットパネルを追加した。[オーナーを変更]ボタンが有効な間だけ、盤を変更できる。チャットは常に行える。

（3） 四井氏の砂[6]



ユーザ名のリスト表示付きチャットパネルを追加した。砂粒は同期対象としておらず、各クライアントで個別にシミュレートされる。

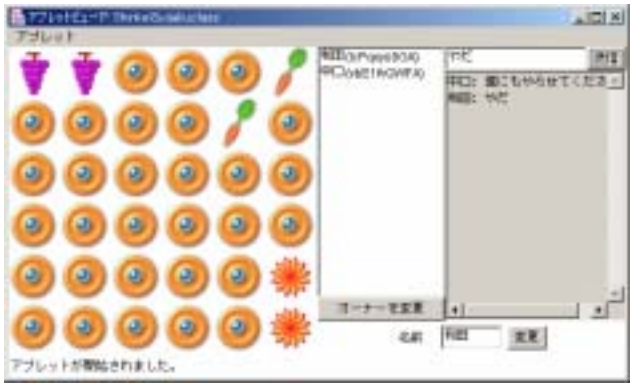
---

[4] http://www-ui.is.s.u-tokyo.ac.jp/~takeo/teddy/teddy-j.htm

[5] http://www.pro.or.jp/~fuji/java/

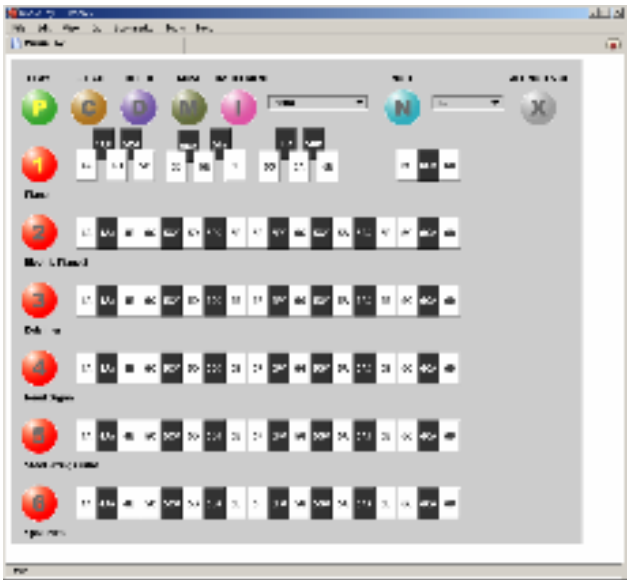
[6] http://www.kdn.gr.jp/~shii/sand/index-jp.html

（４） 神経衰弱アプレット



各パネルの状態同期と、チャットパネルの追加、失敗時の自動オーナー移動を実装した。不正解時にオーナーが移動し、オーナーのユーザが間違い、自分に権限が回ってくるまでは、パネルを操作できない。オーナーの場合、”オーナーを変更”ボタンが有効になる。

（５） MusicToy（楽器作成/演奏アプレット）



ネットワーク共有化。各楽器の編集や演奏を同期。

システムの適用時には、各アプリケーションのオリジナルコードには手を加えていない。

７．今後の課題

実運用に向けて、システムの高速化、メモリ使用量の削減、フレームワークの簡素化、データベース対応、特定用途向けのライブラリ作成などの課題が上げられる。

アスペクト指向によるシステムの適用は、実験的に採用している面が残っているため、今後さらにその手法自体を成熟させて行く必要がある。

また、ビジネス化に繋げるために、エンドユーザ/開発者に向けたドキュメント作成、Web ページの充実に取り組むとともに、明確なマーケティング戦略を作成し、実行して行く。