3Q–10　**SMAF システム :適応的ユーザインタフェース移送の実現**

中澤 仁[1]　喜多山 卓郎[2]　徳田 英幸[1,3]

[1]慶應義塾大学政策・メディア研究科　[2]慶応義塾大学 SFC 研究所　[3]慶應義塾大学環境情報学部

1　はじめに

SMAF プロジェクトでは，モバイルコードを用いた異機種混在環境を対象とした動的適応可能な組み込みシステム用基盤ソフトウエアを開発している．

本研究では，視覚的状態共有機能と，ホスト間でのインスタンス複製および移動機構を持ったウィジェット群を提供する，位置透過型分散共有ツールキットライブラリ (Mobile Graphical User Interface Library : Mogul) を開発した．

Mogul を用いることによってアプリケーションプログラマは，視覚的状態共有機能や複製機能，移動機能を有する分散マルチユーザアプリケーションを容易に開発できる．さらに，ソケット記述子やファイル記述子をはじめとした，ホストに依存するオブジェクトインスタンスへの継続的アクセス機構は，移動透過的なアプリケーション開発を可能とする．また，ホスト間移動に際する計算機環境への適応機構によって，アプリケーションサービス品質の一貫性が保たれる．

2　Mogul

本節では，Mogul の機能，特徴およびユーザインタフェース移送に伴うホスト位置依存データの取り扱いについての問題点を述べる．

2.1　機能

Mogul は，GUI のホスト間移動機構およびウィジェット識別子の管理機構を提供するユーザマネージャと，ウィジェットに対する各種操作機構をユーザコマンドとして提供するコマンド群，および標準の java.awt.Component クラスの代替とから構成される．アプリケーション利用者は，各ウィジェットに対して以下に示す 6 つの操作を行える．

move	ウィジェットを遠隔ホストに移動する．
copy	ウィジェットを遠隔ホストに複製する．
delete	遠隔ホスト上のウィジェットを削除する．
dump	ウィジェットをディスクに保存する．
restore	ウィジェットをディスクから活性化する．
acl	アクセス制御ポリシを変更する．

上記の操作は GUI の構成とは無関係にあらゆる粒度で行える．図 1に，ウィジェットのホスト間移動の概念図を示す．アプリケーションウィンドウ全体のホスト間移動や，プッシュボタン 1 つのホスト間移動など様々な操作が可能である．

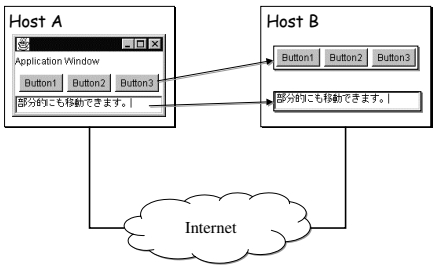

図 1: Mogul ウィジェットの移動性

2.2　特徴

プログラミングの容易性，ホスト位置透過性が Mogul の特徴として挙げられる．

Mogul の提供する各機構は，java.awt.Component クラスに追加することで完全に隠蔽されている．従って，GUI のホスト間移動を実現するためのアプリケーションプログラマによるコードの追加は全く必要ない．

さらにプログラマはアプリケーション記述段階で，Mogul ウィジェットのホスト位置を考慮する必要はない．

2.3　ホスト間移動に伴う問題点

GUI のホスト間移動に際して，以下に示す 2 つの問題が生じる．

- ホスト位置依存データおよび資源への継続アクセス
- ホスト間の性能差に起因するサービス品質の変化

ホスト位置依存データの例としてはソケット記述子やファイル記述子が挙げられる．ホスト間を移動するウィジェットインスタンスが，ソケットやファイルに関連するオブジェクトインスタンスをクラス変数として所有する場合，移動後に当該変数を用いることはできない．

また，表示解像度や処理能力の異なるホストにウィジェットが移動した際には，移動先ホストの計算機環境への適応機構が必要となる．

3　設計

本節では，Mogul における位置依存データへの継続アクセス機構および計算機環境変化への適応機構について設計について述べる．

SMAF System: Adaptive migration of GUI widgets in Mogul

[1]Graduate School of Media and Governance, Keio University
5322, Endo, Fujisawa, Kanagawa 252, Japan
E-Mail: <jin@sfc.wide.ad.jp>
[2]SFC Research Institute, Keio University
[3]Faculty of Environmental Information, Keio University

3.1 設計方針

Mogul は，アプリケーション記述時にプログラマに対する負担を軽減することを目標としている．モバイルエージェント環境 [2] をはじめとした既存のシステムにおいては，それぞれの機能を実現するためのクラスライブラリが提供されており，プログラミング時の制約を高める結果となっている．本研究では，Mogul を用いることによって発生する通常の Java アプリケーションに対する追加的コードを最小限にし，プログラミング時の制約を抑制することを設計方針とする．

3.2 位置依存データへの継続アクセス機構

特定ホスト上で取得した入出力ストリームへの，ホスト位置透過的な継続アクセスを行うための機構である．Mogul では，図 2に示すように遠隔ホストへ移動したウィジェットインスタンスが特定ホスト上でオープンしたソケット記述子やファイル記述子に対する入出力を，当該ホストへリダイレクトすることによって，継続的な入出力を実現する．

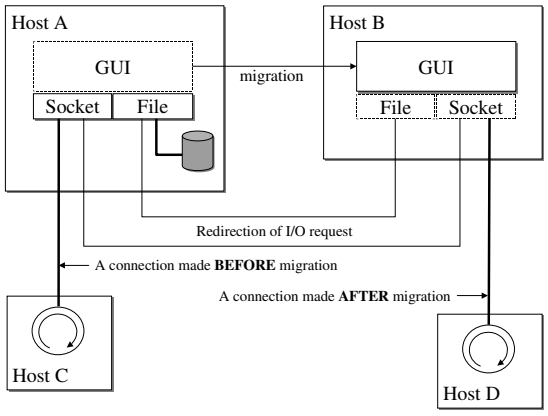

図 2: 入出力リダイレクション

3.3 計算機環境への適応機構

ウィジェットインスタンスが遠隔ホストへ移動した際，移動先ホストの計算機環境を検知し動的に適応するための機構である．計算機環境の変化をイベントとしてアプリケーションに通知し，ユーザマネージャの管理する環境情報を提供する．環境情報の例として，CPU アーキテクチャ，ディスプレイ解像度，メモリ容量などが挙げられる．図 3，アプリケーションプログラム中から本機構を利用する例を示す．

4 実装

Mogul の実装は Java 言語を用いて行った．JDK(Java Development Kit) のマルチプラットフォーム性から，ユーザは自分自身の使い慣れたオペレーティングシステムやユーザインタフェース上で，Mogul を用いて構築された分散マルチユーザアプリケーションを利用できる．

```
import java.awt.*;
import JP.ac.keio.sfc.ht.mogul.event.*;
public class TestWindow extends Frame
implements EnvironmentEventListener {
  public TestWindow(){
    add(``Center'', new Button(``button''));
    pack(); show();
  }
  public void environmentChanged
  (EnvironmentEvent e){
    e.getArchitecture();
    e.getMemoryAmount();
    e.getDisplayResolution();
    e.getOSName();
    e.getOSVersion();
  }
}
```

図 3: サンプルコード

4.1 位置依存データへの継続アクセス機構

Mogul ウィジェットからのソケット記述子やファイル記述子へのホスト透過的な継続アクセス機構は，java.net パッケージ，java.io パッケージに含まれるクラスをサブクラス化して実現した．本クラス群を用いることによって，ホスト移動後の I/O 要求は，移動元ホストへリダイレクトされる．また，移動先ホスト上で新たに作成されたソケット記述子やファイル記述子への I/O は直接行われる．

4.2 計算機環境への動的適応機構

Mogul ウィジェットへ，計算機環境の変化を通知することを目的として，EnvironmentEvent クラスを導入した．本クラスは，Mogul ウィジェットのホスト間移動時にユーザマネージャから当該ウィジェットの environmentChanged メソッドの引数として渡される．ウィジェットは，同クラスのメソッドを呼び出すことによって，新たな計算機環境を構築する種々の項目の情報を適宜取得できる．

5 まとめ

本稿では，位置透過型分散共有ツールキットライブラリ Mogul に関して，動的適応性および位置透過性の観点から設計と実装を記述した．現在プロトタイプの実装を終了し，いくつかのアプリケーションモデルに基づいた応用評価を行っている．

参考文献

[1] 中澤 仁, “Mogul :位置透過型分散共有ツールキットライブラリ”, 情報処理学会システムソフトウエアとオペレーティング研究会，78-16，May 1998．

[2] G. Karjoth, D.B. Lange, M. Oshima, “A Security Model for Aglets”, IEEE Internet Computing, July-Aug 1997.