ユビキタス環境におけるモバイルエージェントを用いたソフトウェアの動的更新手法

佐伯 智之† 河口 信夫‡† 稲垣 康善*

†名古屋大学大学院情報科学研究科
‡名古屋大学情報連携基盤センター
*愛知県立大学情報科学部

Dynamic Software Update using Mobile Agent for Ubiquitous Environment

Tomoyuki Saeki † Nobuo Kawaguchi ‡† Yasuyoshi Inagaki *

†Graduate School of Information Science, Nagoya University
‡Information Technology Center, Nagoya University
*School of Information Science and Technology, Aichi Prefectural University

1 はじめに

近年，情報機器の小型・軽量化，低価格化が進み，携帯端末や情報家電が急速に普及しつつある．それに伴って，小型の計算機が埋め込まれた機器がいたるところに存在する，ユビキタスコンピューティング環境の実現が期待されている．そのような環境においては，多くの情報機器が身の回りのさまざまな場所に存在するため，新しい機能の追加や，不具合の改修のたびにおこなわれるソフトウェアの更新作業に多大なコストが必要となる．また，そのような環境で動作しているソフトウェアは，更新の際に機器の再起動などの必要がないことが望ましい．このようにユビキタス環境では，ネットワーク環境内の各機器で動作しているソフトウェアを，それが動作中にもつ内部変数などのデータを保持したまま，機器を再起動することなく更新できるようなシステムが望まれる．動作中ソフトウェアを更新する手法として，OS にプログラムの状態監視など新たな機能を実装し，プロセスレベルでの動的更新を実現する手法がある [1][2]．しかし，よりシンプルで，容易に他端末上で動作中のソフトウェア更新への応用が可能な更新手法が必要であると考えられる．

我々は，モバイルエージェントに基づいてソフトウェアの動的な更新を実現する手法を提案してきた [3][4]．ソフトウェアの動的な更新とは，動作中のソフトウェアのコードを，それが動作中に持つ内部変数などのデータを保持したまま，新たなコードへと更新することである．このような動的更新を行う際，更新前後のソフトウェア間で，その実行状態のデータ構造が異なる場合に正しく更新されないという問題があった．この問題に対し，コードの更新に合わせてデータも更新することにより，データ構造が異なるコードへの更新も可能にする手法を提案した [5]．本稿では，それらの手法について簡単に説明し，ネットワーク内にある各端末上のソフトウェアに対して更新を行う方法について述べる．また，この手法に基づいて動的な更新を行うシステムを実装し，実際に作成したソフトウェアに対して動的更新を行い，本手法の実現可能性および有効性を示す．

2 動的更新手法

ここでは，我々が提案してきた，モバイルエージェントを用いたソフトウェアの動的更新手法について述べる．

モバイルエージェントとは，その実行状態を保持したまま，自律的に計算機間をネットワークを介して移動し，移動先の計算機上で処理を継続するソフトウェアである．このモバイルエージェントは，プログラムコードと，実行状態などのデータから構成されているが，移動の際にはこれらを一旦通信可能なデータ形式に変換（シリアライズ）してからネットワーク上に転送され，移動先の端末に到着したときに逆の変換（デシリアライズ）を行い，復元される．

動的更新は，このエージェントの移動を同一端末上で行うことで実現する．すなわち，単純なコードのみの更新であるなら，コードだけを更新したものと入れ替え，データはそのままシリアライズし，同一端末上でデシリアライズすることで，更新されたコードからなるエージェントが更新前の実行状態を保持したまま復元され，動的な更新が実現できる（図 1）．

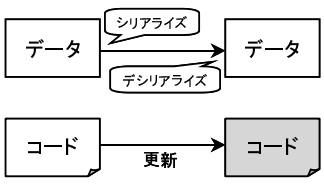

図 1: モバイルエージェントの動的更新

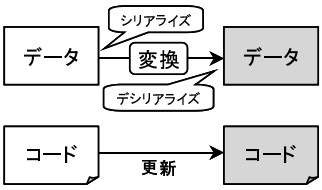

図 2: データ構造の変更に対応した動的更新

3　データ構造の変更

前節で提案した手法では，更新前後のソフトウェア間でそのデータ構造が異なる場合，更新前のソフトウェアのデータが，更新後のソフトウェアのコードに適合せず，動的な更新が正しく行われないことがある．そのため，前節の手法で，矛盾の起こらないように更新を行う場合，プログラムコードの更新を，ある程度の制限内で行う必要がある．それは例えば，変数追加などのデータ構造の変更を伴わないような更新である．しかし，そのような制限内での更新では，ソフトウェアの満足な更新はできない．そこで，データ構造の変更を伴うような更新も可能とするための手法として，データ変換を伴った動的更新手法を提案した．その更新手順を以下に示す．

1. エージェントをシリアライズする
2. コードを更新したものと入れ替える
3. シリアライズされたエージェントのデータを，更新先コードに適合するように変換する
4. 変換後のデータをデシリアライズし，エージェントを復元する

このように，コードのみを更新するのではなく，データもコードの更新に合わせて更新することにより，更新前後のソフトウェア間でデータの構造が異なっている場合にも，正しい更新を実現する（図 2）．

4　他端末上のソフトウェアの更新

提案した動的更新の仕組みと，コードなどの必要な情報を持って移動するエージェントを組み合わせて用いることにより，他の端末上のソフトウェアを，ネットワークを介して動的に更新することが可能となる．すなわち，エージェントの更新を行う端末に，更新されたコードや必要なデータ変換コードを持ったコード移送エージェントが移動し，提案した動的更新手法によって移動先端末のエージェントの動的更新を行うことができる．以下，図 3 を用いて更新手順を説明する．

まず，①更新を指示する端末が，ネットワーク内の各端末で動作しているエージェントに関する情報（エージェントの名前，エージェントの ID，エージェントの現在のバージョン番号）および端末自体の情報を取得するため，情報取得エージェント (AcquireInfo) を各端末に移動させる．これらエージェントが獲得してきた情報と，更新指示端末に存在するプログラムコードおよびデータ変換コードより，更新可能なエージェントをピックアップする．更新不可能と判断される例として，更新先バージョンへの更新に必要となるデータ変換コードが存在しない，端末情報からそ

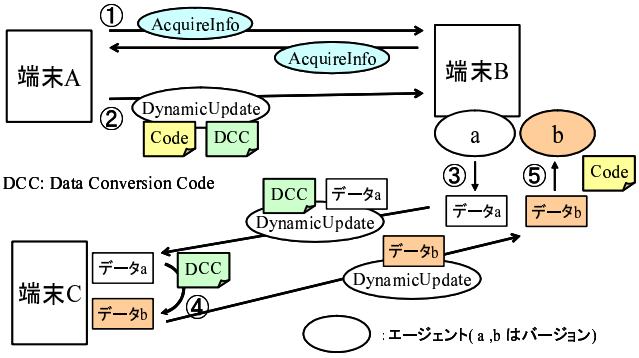

図 3: ネットワークを介したソフトウェアの動的更新

の端末は更新が明示的に禁止されている，端末情報からメモリなどの実行環境とプログラムコードのサイズなどの理由から更新ができない，などが挙げられる．次に，②更新を指定したエージェントが動作する端末に，更新先バージョンのコード (Code) と必要となるデータ変換コード (DCC) を持った移送エージェント (DynamicUpdate) を移動させる．そして移動先の端末で，同一端末上の動的更新手法を用いて指定エージェントの動的更新を行う．移動先の端末に，データ変換を行う機構が存在しない場合には，他の端末にデータ変換を委託する．その場合は，③更新対象エージェントをシリアライズしたデータと必要なデータ変換コードとともに委託先端末に送り，④その端末でデータ変換を行う．その後，変換後のデータを，更新を行う端末に送り返し，⑤そのデータを，更新されたコードを用いてデシリアライズし，エージェントを復元する．

5　データ変換の応用

前節でも述べたように，ソフトウェアが動作している機器の実行環境の制約など何らかの理由により，新しいバージョンへと更新できない端末が存在することが考えられる．この場合，複数のバージョンのソフトウェアが同時に混在する環境において，それらがデータをやりとりする状況が生まれる．データ構造が異なるバージョンのソフトウェア間でデータのやりとりを行う場合には，一般にそのままでは正しく通信が行われない．そこで，データ構造の違いを補うため，前節で提案したデータ変換をここでも利用する．すなわち，送信先のソフトウェアに合わせたデータ変換により，データ構造の異なるバージョンのソフトウェア間で矛盾の起こらないようなデータのやりとりを可能とする．データの変換処理を行う場所としては，(1) データ送信元の端末で行う，(2) 送信先の端末で行う，(3) やりとりをしている端末以外の端末で行う，の 3 通りが考えられる（図 4）．(3) は，ソフトウェアが動作している機器にメモリの制限があるなどして，余分な処理が行えない場合に有効である．(1) の実現には，送信前にデータの変換を行わなければならないため，あらかじめデータ送信先ソフトウェアのバージョンを知っている必要がある．(2) の実現には，送信データに送信元ソフトウェアのバージョンを情報として付け加える必要がある．(3) の実現には

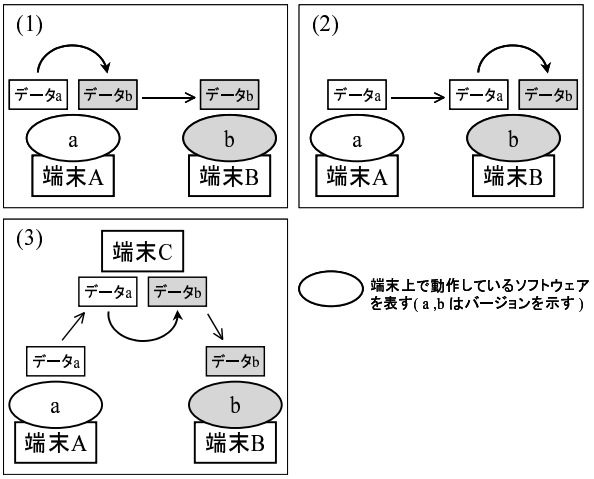

図 4: データ変換を行う場所の種類

(1)，(2) 双方の情報に加え，変換を行う端末を決定する必要がある．また，(1)〜(3) 全てにおいて，それぞれ必要となるデータ変換コードを獲得する必要がある．

このような通信の際に必要となるデータ変換には，以下の 2 通りが考えられる．

1. 更新の行われなかったバージョンのソフトウェア (旧バージョンとする) から，更新の行われた新しいバージョンのソフトウェア（新バージョンとする）へとデータが送られた場合に，旧バージョンのデータを新バージョンに適したデータに変換する
2. 新バージョンから旧バージョンへとデータが送られた場合に，新バージョンのデータを旧バージョンに適したデータに変換する

1 については以下のようにデータ変換を行う．新バージョン側の端末は，旧バージョンからのデータを受け取ったら，必要となるデータ変換コードを獲得して，データ変換を行う．2 については，新バージョンが旧バージョンにデータを送る前に，あらかじめ先ほどと同様にデータ変換コードを獲得し，データ変換を行ってから，データを送る．この実現のために，エージェントの更新が行われた端末には，更新が行われなかったエージェントが存在する端末の情報（ノード ID，そのエージェントのバージョンなど）を更新を行う際に記録する．また，そのとき利用したデータ変換コードおよび，逆の変換に必要となるデータ変換コードが存在する端末の情報 (更新指示端末のノード ID) も記録しておく．なお，この場合，データ変換場所は基本的には新しいバージョンのソフトウェアが動作している端末ということになるが，その端末で行うことに問題がある場合には，他の端末に委託して変換を行うことも考えられる．

6 実装

前節で述べた手法に基づいて，動的更新を行うシステムを，Java で実装されたモバイルエージェントシステム cogma[6] を利用して実現した．

コードの動的な更新は，更新されたコードを，現在のバージョンのコードを読み込んだクラスローダとは異なるクラスローダで読み込むことによって実現する．Java ではこのようにクラスローダを用いることにより，コードから必要なクラスを動的にロードすることができる．この新しいコードへの入替えが行われた状態で，シリアライズされたエージェントをデシリアライズを行えば，更新されたコードからなるエージェントが復元される．

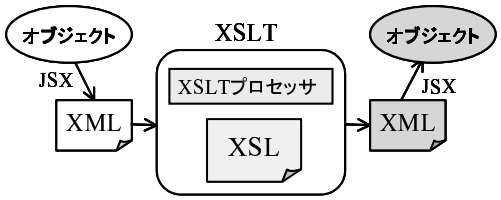

図 5: データ変換

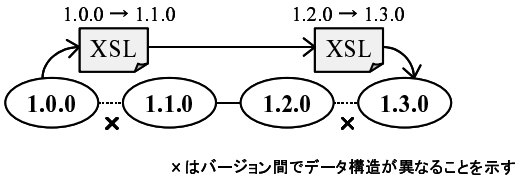

図 6: 複数バージョンをまたぐ更新

データ構造が異なるコードへの更新の場合には，データ変換が必要となる．本実装では，エージェントのデータであるエージェントクラスのオブジェクトを，JSX[7] を用いて XML に変換し，どのように変換するかを記述した XSL ファイルを用いてその XML データを XSLT[8] によって更新先コードに適合する形にデータ変換し，再び JSX を用いてオブジェクトへと戻す（図 5）．このようにしてデータ変換を実現する．XSL ファイルは，データ構造の異なる隣接バージョン間ごとに，双方向分用意する．複数バージョンをまたぐ更新をおこなう場合には，必要となる XSL ファイルを順次用いることによってデータ変換を行う（図 6）．

7 図形エディタの動的更新

ここで，実際に行った動的更新の例を紹介する．更新対象として，簡単な図形エディタをモバイルエージェントとして作成した．この図形エディタの作成過程である以下の 3 バージョン間で更新（アップグレード・ダウングレード）を行った．

- バージョン 1.0.0： 直線・長方形の描画機能，描画した図形の選択・削除機能を持つ
- バージョン 1.1.0： バージョン 1.0.0 が持つ機能に加えて，図形の移動機能を持つ
- バージョン 1.2.0： バージョン 1.1.0 が持つ機能に加えて，図形の回転機能を持つ

バージョン 1.2.0 では，図形の回転機能を付け加えるにあたって，長方形を表すクラスに，長方形の回転角度を表す変数を新たに追加した．これにより，バージョン 1.1.0 と 1.2.0 の間でデー

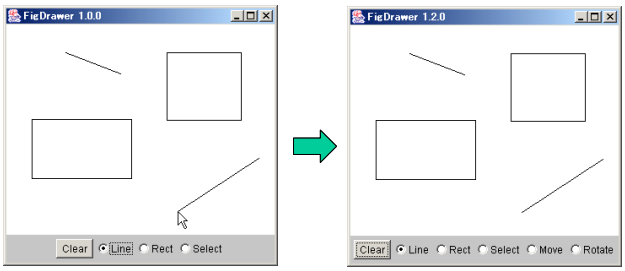

図 7: バージョン 1.0.0 から 1.2.0 への更新

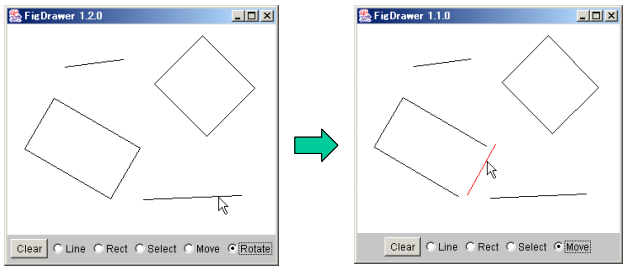

図 8: バージョン 1.2.0 から 1.1.0 への更新

タ構造に違いが生じるため，その間をまたぐ更新を行う際には，データ変換を行う必要がある．

まず初めに，バージョン 1.0.0 から 1.2.0 への更新を行う．この更新では，バージョン 1.1.0 から 1.2.0 へのデータ変換処理，すなわち，新しく追加した変数をデータに追加する処理を記述した XSL ファイルを用意する．

次に，バージョン 1.2.0 から 1.1.0 への更新を行う．この更新では，バージョン 1.2.0 から 1.1.0 へのデータ変換処理を記述した XSL ファイルを用意するが，ここで，この更新の方針として，バージョン 1.2.0 で回転させられた長方形を，回転した長方形の描画ができないバージョン 1.1.0 でも見た目が変わらないようにすることとし，バージョン 1.2.0 における回転した長方形を，バージョン 1.1.0 では 4 本の直線で表すようにした．

初めの更新の様子を図 7 に示す．描かれた図形の状態を保持したまま，move，rotate の機能が追加されていることがわかる．2 番目の更新の様子を図 8 に示す．更新後のバージョン 1.1.0 で，回転した長方形が 4 本の直線から成るようにデータ変換されていることがわかる．

2 番目の更新において，変換対象となるデータを JSX によって XML 化したデータの一部を次に示す．

```
<org.cogma.codget.FigDrawer_-Rect
 x="34" y="135" width="118" height="70" rotation="30"/>
<org.cogma.codget.FigDrawer_-Line
 x1="72" y1="73" x2="141" y2="64"/>
    ...
```

この XML データを，XSL を利用して，回転した長方形データを 4 本の直線データへと変換している．以下に変換に用いた XSL の一部を示す．

```
<xsl:template match="org.cogma.codget.FigDrawer_-Rect">
 <xsl:if test="@rotation[.='0']">
  <xsl:copy>
   <xsl:copy-of select="@x"/>
        ...
  </xsl:copy>
 </xsl:if>
 <xsl:if test="@rotation[.!='0']">
  <xsl:element name="org.cogma.codget.FigDrawer_-Line">
   <xsl:attribute name="x1">
    <xsl:value-of select="result:getRX(number(@x),number(@y),
     number(@width),number(@height),number(@rotation),0)"/>
   </xsl:attribute>
        ...
 </xsl:if>
</xsl:template>
```

以上より，提案した手法を用いることによって，単純なコード更新に加えて，データを変換することにより，データ構造の異なるコードへの動的更新も行えることを示すことができた．

8 まとめ

本稿では，モバイルエージェントに基づいたソフトウェアの動的更新手法，およびデータ構造の変更にも対応したソフトウェアの動的更新手法について述べた．また，ネットワーク上の各端末で動作しているエージェントを各端末の状態に合わせた動的更新を行う手法を提案した．また，モバイルエージェントシステム cogma 上に，提案手法に基づいて動的更新を行うシステムのプロトタイプを実装し，実際に作成したソフトウェアに対して動的更新を行い，その実現可能性や有効性を示した．

今後の課題としては，他のソフトウェアと連携して動作しているソフトウェアを動的に更新する手法の検討や，データ変換のために必要となる XSL ファイルの半自動生成を行う手法の検討などが挙げられる．

参考文献

[1] 谷口秀夫，伊藤健一，牛島和夫：プロセス走行時におけるプログラムの部分入れ替え法， 電子情報通信学会論文誌，Vol.J78-D-I，No.5，pp.492-499 (1995)

[2] 小林 良岳，佐藤 友隆，唐野 雅樹，結城 理憲，前川 守: 彩：コンパイル時に自動生成される Portal をもとに動的再構成可能なオペレーティングシステム，電子情報通信学会論文誌 VOL.J84-D-I No.6, pp.605-616 (2001)

[3] 河口信夫，外山勝彦，稲垣康善：モバイルエージェントに基づく動的拡張可能なソフトウェアシステム，情報処理学会夏のプログラミング・シンポジウム，pp.71-78 (1999).

[4] 植田智，河口信夫，稲垣康善：モバイルエージェントシステムにおけるオンデマンドコード移送とバージョン管理，情報処理学会第 6 4 回全国大会，pp.509-510 (2002).

[5] 佐伯智之，河口信夫，稲垣康善：データ構造の変更に対応したソフトウェアの動的更新手法， 第 6 回プログラミングおよび応用のシステムに関するワークショップ (SPA2003), (2003)

[6] 河口信夫，稲垣康善：cogma:動的ネットワーク環境における組み込み機器間の連携用ミドルウェア，情報処理学会コンピュータシステム・シンポジウム，pp.1-8 (2001).

[7] JSX(Java Serialization to XML)：(http://www.csse.monash.edu.au/~bren/JSX/)

[8] XSLT(XSL Transformations) Version1.0：(http://www.w3.org/TR/xslt)