ヒューマンインタフェース 81－4
（1998. 12. 11）

移動エージェントユーザーインターフェース:Bali

美馬 義亮

日本アイ・ビー・エム 東京基礎研究所

移動エージェントは最近実用化が進みつつある技術であり、これはネットワークで接続されたコンピュータ上において、さまざまなシステムレベルで適用可能なソフトウエア技術である。この技術が適用されたシステム上では、プログラムの自律的な移動・実行が行われ、移動エージェントの振る舞いをユーザが直接操作するような状況が生まれうる。また、移動エージェントのプログラムの開発においては、その振る舞いを視覚化したり、デバッグのためユーザの直接操作によってエージェントの振る舞いをコントロールしたりするということが要求される。Bali はそのような要求を満たすためのユーザ環境となるべく、試作された移動エージェント用のデスクトップシステムである。

User Interface for Mobile Agent: Bali

Yoshiaki Mima

IBM Research, Tokyo Research Laboratory

On personal computers and workstations, the GUI with desktop metaphor is widly accepted. We have been developped a mobile agent system. As one of the tools for mobile agents, we developped a desktop-like visual shell: Bali on top of the mobile agent system. The purpose of developing Bail is to enable users natural and intuitive handling of mobile agents. In this paper, we will introduce a way of extending the concept of desktop metaphor to agent based desktop. We are describing the outline of GUI and inside of its structure as an example together with introduction of our own mobile agent system.

1. はじめに

これまで、我々はオブジェクト指向言語 Java をベースにした移動エージェントシステム Aglets [1]を開発してきた。これらの移動エージェントとの対話のためのインターフェースは、単純なプログラム開発を前提にしておりなるべくシンプルなインターフェースをもったものを提供するようにしてきた。しかし、移動エージェントの応用範囲が広がり、移動エージェントをさまざまな用途に利用するためには、移動エージェントシス

テムの監視やデバッグが重要になるとともに、ウィンドウシステム上に構築されたビジュアルシェルであるデスクトップ[2]と同等に位置づけられているようなインターフェースをもったビジュアルシェルが必要となることは明らかである。

移動エージェントを前提としたデスクトップが今までのデスクトップと異なるのは、移動エージェント自体が自律的な存在であり、ユーザの直接的な指示がなくとも自分で移動してしまったり消滅してしまうという点にある。また、別のサーバからエージェントが作業中のデスクトップに移動しきて勝手に行動することもある。これらはこれまでのユーザがコンピュータ上の処理を主導して行っていた環境とは異なるものである。本稿では、移動エージェントを制御するための自然なユーザインターフェースの一例として、移動エージェント用のビジュアルシェル Bali[3]を試作したのでこれについて紹介と関連した考察を行う。

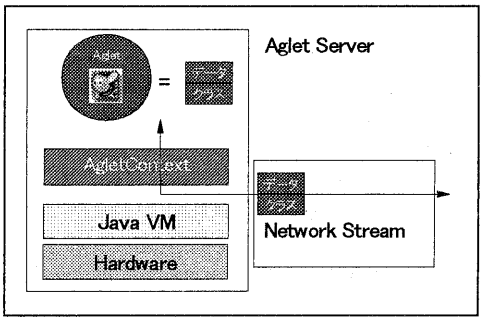
2 章では移動エージェントシステムの概要を紹介し、3章でBaliについての紹介、4章で考察を行い既存デスクトップと移動エージェント用のデスクトップの違いについて述べ、最後に5章で今後の課題について延べる。

2. 移動エージェントシステム Aglets

移動エージェントには多様な実現方法があり、規格化などの努力は行われているものの、それに対する操作方法についても、必ずしも一定の規則が存在するわけではない。ここで紹介するのは我々自身が開発してきた移動エージェントシステム上のエージェントを操作するためのユーザーインターフェースである。したがって、最初に対象となる移動エージェントシステムの概要を紹介する。

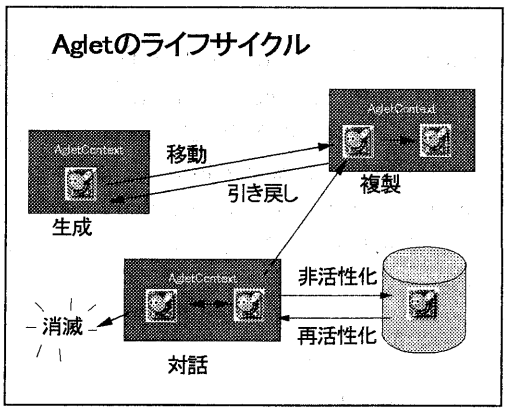
①. 移動エージェントフレームワーク Aglets

Aglets は移動エージェントを実現するためのフレームワークであり、Java 言語で記述されたライブラリとして実現されている。Aglets はオブジェクトのデータを Serialize すること、ならびに転送先でオブジェクトのクラスをロードして実行させることにより、そのオブジェクト自体が別の計算機に移動して再実行可能な仕組みを提供している。このとき転送先の計算機上にはエージェントサーバと呼ばれるプロセスが実行されており、他のエージェントサーバからエージェントが到着するのを監視している。



Aglet は生成、複製、非活性化、活性化、移動、対話、消滅からなる状態を遷移するライフサイクルをもつものであり、AgletContext と呼ばれる環境の中で管理され、また AgletContext の中だけで実行可能な Java のオブジェクトである。

Aglets では、エージェントがこれらの移動に本質的な行動を行うときにはシステムはそのイベントの前後にイベントの発生を受け取る仕組みを持っている。



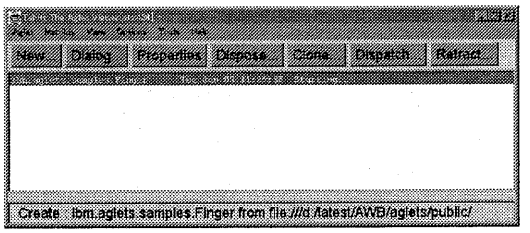
AgletContext には名前をつけることが可能であり、一つのエージェントサーバ中に複数の AgletContext を持つことも可能である。エージェ

ントが移動するときには、その移動先としてはAgletContextのアドレスが指定される。

エージェントにとってはAgletContext間の移動とは、そのAgletContextから利用できる計算機資源の量や質が異なる事、ならびに同一のコンテキストに存在するエージェントの構成が変化するため異なった能力や情報を持つエージェントとのコミュニケーションが可能になるという変化がおこるということである。

②. エージェントビューア

Aglets Software Development Kitでは Tahitiという Viewerが提供され、エージェントに対してこれらの状態遷移を外部から強制する事ができる。実はTahitiは実行スレッドのビューアのようなものであるが、特に人間との対話に重点を置いて開発したものではなく、エージェントを扱うための最低限の機能を実現したものと位置づけられる。



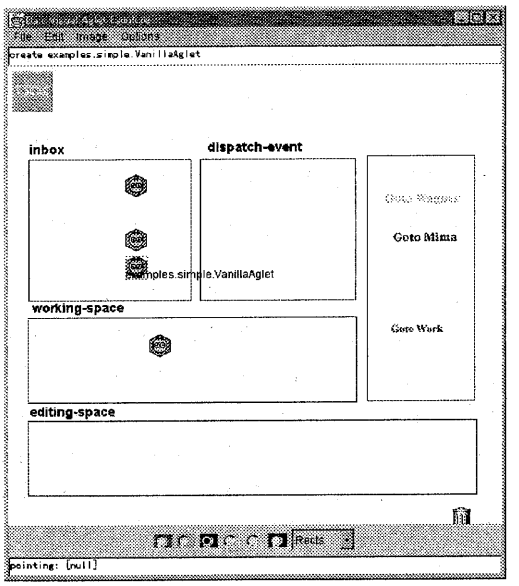
Tahitiは存在する一つのエージェントを一項目として表示するリストをもち、ユーザ操作により、選択されたおのおののアイテムに対して、消滅、複製、非活性化、活性化、移動などの変化を起こす事ができる。

3. エージェント用デスクトップ:Bali

今回試作したデスクトップBaliはTahitiを置き換えることにより、従来のウィンドウシステム上で提供されているビジュアルシェルとしての機能を拡張したデスクトップインターフェースを提供するものである。

このデスクトップでは画面上がいくつかの長方形の領域に別れており、それぞれの領域に対してAgletContextが割り当てられている。実現上の視覚的なバリエーションとしては、これらの領域にデスクトップのフォルダのようなものを割り当てることが可能である。個々の領域にはAgletContextの名前が割り振られている。移動エージェントAgletが外部から、Baliを用いてこのユーザが管理するサーバーに移動するときは特定のAgletContext（複数設定する事が可能、この例の場合はinboxただ一つ）にのみ移動を許可するように設定を行うするものとする。

下図の右にある"Goto ..."といったアイコンのひとつ一つは主としてリモートサイトにある別のAgletContextを意味するアイコンとなっている。ユーザはいったん（この例ではinboxに）受け入れたエージェントに対し、これらの領域間におけるアイコンの直接的な移動操作、または別のAgletContextを意味するアイコンへの重ねあわせによってさらに別のAgletContextに移動させる事が可能である。のこりの領域の意味について以下で簡単に説明を行う。



Inbox:これらのAgletContextにはさまざまなエージェント実行上の制限が加えられている。たとえば、(この例における inbox のような) 他の場所からエージェントが到着する場所では、エージェントの自律的実行を制限するため、スリープ状態になるようにする事が可能である。この inbox という名前はは電子メールの inbox から借用したものである。よそからきたエージェントはすべてこの inbox という場所を経由して処理がなされることになる。

Dispatch-event というAgletContextは、エージェントが他へ移動しようとした場合にそれらの動きを捕捉して、一旦そのエージェントを活動停止状態にした上で、ユーザーによって許可をもらいで別のサーバーへ送るというユーザにとっては制御の機会が与えられる。電子メールならば自分自身で勝手に配送されるメールは存在しないが、移動エージェントではそういう存在が普通である。この仕組みは、外へ行くエージェントの関所となる。だから、ユーザにとって不審な「出エージェント」はその素性をチェックされた上でその後の行動を決定する事になる。

Working-Space とは、現在のところ、セキュリティ要件を満たす範囲で、エージェントが移動以外の行動を自由にできる場所である。このサーバ上にそのような AgletContext を複数おいても一向に差し支えない。(ただし、移動に関わる動作を行うとシステムにトラップされdispatch-event に送られる事になる。)

Editing-spaceはユーザが、個々のエージェントに対し、それが誰の権限で生成されたのかとか、誰が作ったコードなのか、どこからやってきたコードなのか、などを見る事によって素性をチェックしたり、エージェントに直接メッセージを送ることが可能になったりできるような作業場である。ここで提供される機能についてはまだ考察の余地がある。

各AgletContextに対応する領域には自動的にアイコンのレイアウトを行うプログラムが割り当てられており、おのおのの領域におけるレイアウティングポリシーによりそのアイコンにより表示を動的に変化させるようになっている。

4. 考察

Bali においては個々の移動エージェント、転送先のサーバがアイコンとして表現されている。移動エージェントが別のサーバから送られてくると、画面上に対応するアイコンが自動的に生成され表示される。逆に、エージェントが消滅したり、他のサーバに移動するとアイコンは消滅する。

一般的なデスクトップにおけるオブジェクトの操作と移動エージェントとしてのオブジェクト操作の本質的な差異は移動エージェントだけがもつオブジェクトがもつ自律性にある。これらの差異を箇条書きにしてみると以下のようになる。

- 従来型のデスクトップではユーザの操作のみによりアイコンが生成・コピー・消去されるが、エージェント対応のものでは、ユーザ操作なしにこれらの事象が発生しうる。
- また、従来型のデスクトップではアイコンの配置はユーザの手によって決定されたが、エージェント対応のデスクトップではシステムによってその配置が行われることになる。

最初に述べられている性質はエージェントシステムの自律性の存在から考えると当然のものではあるが、ユーザからエージェントをコントロールすることに大きな意味がある Bali においては本質的な問題であった。Tahiti 上での操作を行う場合、人間の対応時間と比較してエージェントの動作が早い場合には、(エージェント側がダイアローグを出して操作者に対して問いかけを行なわない限り) エージェントサーバー上の出来事が起こった事さえ気づかないということもあった。この点の改良がBaliの存在意義としてあげられる。

また、二つ目にあげた問題も小さな問題ではない。Tahitiでははじめから画面上でモニタできるエージェントの数はせいぜい数十止まりであり、

それ以上多くのエージェントをモニタする事は実質上不可能であった。システムのテストを行う場合には、大量のエージェントの生成消滅のモニタを行う要求があり、一度に多数のエージェントを表示することが求められる。もとより、Baliにおいては Tahiti よりも多くのエージェントを表示する事は可能ではあるが、それでもエージェントの数があまり多くなると効率的な管理を行っているとはいえず、それらを表示する合理的な手法についてはまだ研究の余地がある。このため、ここでは、動的なエージェントの表示機構の重要さを指摘するにとどめたい。

5. 今後の課題

ここで述べられているユーザーインターフェースに関連したテーマ、とくにエージェントの勝手な動きをどう人間の管理下に置くかに関する問題は、分散され自律的に実行される移動エージェントを用いたプログラム全般に適用されうる。

今回あげた、エージェントを複数の AgletContext に分けて分割統治するという考え方は、一つの解決策ではあるが、必ずしも唯一の解決方法であるとはいえない。たとえばエージェントの振る舞いはその移動だけではなく、メッセージ通信や資源管理という側面がある。エージェントの管理者としては、どんなエージェントがどれぐらいの頻度でメッセージ通信をおこなっているのかについて視覚化がなされれば、システムのパーフォーマンスの改善に貢献する可能性があるし、どんなエージェントごとにどんな資源を必要としているのかを視覚化することにも同様に重要な意味がある。

エージェントの動作、そしてその動作の意味は、エージェントを使う人の要求ごとに違ったものになる。その多様な要求に従ってエージェントの振る舞いを評価し、制御できるようなシステムを設計する事が次なる目標だと思われる。

考察で述べたの二つの問題のどちらにも関連する事だが、現在は一つのタスクに利用されるエージェントの数がそれほど多くはなくても、将来においては一つのタスクごとに数百から数千のエージェントを巻き込んで作業を進めることが起こると仮定してみよう。これらは、仮想商品取引などでは十分に起こりそうなシナリオであるが、このとき、もはや管理者が一つ一つのエージェントを眺め、トレースしているということは必ずしも意味がなくなり、エージェントのオペレーションはまさに化学反応の状況を呈することも起こりうる。サーバー間におけるエージェントの平衡状態とか最終成果情報を持ったエージェントの抽出といった化学メタファーなどもまた、表現手段の候補として現実味をおびた解決策であるようにも思われる。

6. まとめ

移動エージェントシステムを構築するにあたっては、移動エージェントシステム自体やその試験を行うにあたって、これらのエージェントを視覚的に表現し、直感的に扱うことが要求される。エージェントシステム自体が持つ自律性とユーザの意志を反映したエージェントのコントロールという二つの問題を解決しようとしたアプローチとしての試作システムBaliの概要の紹介を行い，本デスクトップに対する新たな考察を行った。

7. 辞謝

この研究は IBM 東京基礎研究所における、Aglets Project の中で大島満氏が試作したデスクトップ Bali にヒントを得て始めたものである。彼の本研究における初期の貢献に感謝したい。

参考文献

[1] D.Lange, M. Oshima "Programming and Deploying Java Mobile Agents with Aglets," 1998, Addison-Wesley

[2] D. C. Smith, et. al, "Designing the Star User Interface," BYTE, April/1982

[3] Y.Mima, "Bali: A Live Desktop for Mobile Agents," Proc. Of IEEE APCHI '98, July, 17, 1998