ソフトウェア工学 77−14
（１９９１. ２. ８）

ＵＩＭＳの試作とそのニュースリーダへの応用

佐藤 豊　　真野 芳久

電子技術総合研究所 情報アーキテクチャ部 情報ベース研究室

開発中のユーザインターフェース管理システムｖｉａと、これをニュースリーダに適用した例について述べる。ここでは対話システムを並行動作するエージェントの集合体として構成する。任意の言語で記述された各エージェントはｖｉａプロトコルによってｖｉａシステムに接続し、エージェント間はメッセージ通信により結合する。ｖｉａプロトコルはネットワーク透明なバス型の接続メカニズムであり、メッセージのブロードキャストや、エージェントの動的な接続や入れ換え機能を提供する。この上に応用部品、対話部品、およびユーザインターフェースの記述と開発を支援するためのUIMS部品が、それぞれエージェントとして接続される。本システム上に、各種ニュースリーダが共通に持つ機能を集めた抽象ニュースリーダを接続し、いくつかのニュースリーダの機能とインターフェースの記述を試みた。

Design and Preliminary Implementation of a User Interface Management System and its Application for Electronic-news Readers

Yutaka Sato　Yoshihisa Mano

Electrotechnical Laboratory
1-1-4 Umezono, Ibaraki 305, Japan

This paper discusses the design of a User Interface Management System named Via and its application for electronic-news reader tools. We organize an interactive system as a collection of concurrent agents. Description language for each agent is not restricted to a specific language, and agents communicate network-transparently with each other over a Via-protocol. The Via-protocol supports message broadcasting, and agents can participate in or leave Via dynamically. Several UIMS agents are currently under development to support description and development of interactive systems. An abstract news reader with common news reader functions is connected to this system, and several experimental interfaces and functions for news readers are created.

1 まえがき

対話システムの利用者インターフェースは、利用者がシステムの持つ機能を使いこなし、容易に、かつ感覚的にも快適に利用するための重要な要素である。このために従来から良質の利用者インターフェースを効率的に開発し、改良して行くための支援環境が必要とされてきた。近年では、対話システムにおけるプレゼンテーションのマルチメディア化や、複数の対話が同時進行する(マルチスレッド)対話モデルに対応する必要性から、対話インターフェースの開発を系統的に支援するシステムへの要求がさらに高まっている。

この目的のために、利用者インターフェース管理システム(User Interface Management System:UIMS)は生まれた。UIMSでは、対話システムのアプリケーションに依存する部分(応用部)と、インターフェース(対話部)とをできる限り疎に分離して、対話部と応用部との独立な開発を支援し、結合機能を提供する。このような、対話部と応用部の分離、すなわち対話独立性(dialogue independency)が、UIMSにおいて最も重要な概念であることは、これまでに開発された多数のUIMSを通じて共通の認識となっている[Hart89, Cock89, Cout89, Salv89, Manh89]。

しかし、いかに分離するかは、UIMSに限らずソフトウェア開発においては常に設計者を悩ませる、答えの得にくい問題である。実際、UIMSにおいても、対話部と応用部をいかに分離すべきか、またそれらをいかに結合して管理するかについては、合意は得られていない。1985年に提唱されたSeeheimモデル[Gree85]は、対話独立性の概念をきわめて簡潔にモデル化した(図1)。その後このモデルをベースにして多数のUIMSが作られ、多くの問題点が明らかになった。ここでの問題もやはり、分離の方針を明確に与え得るのか、分離によって記述がより複雑に(困難に)ならないか、結合のための実行時コストが高くならないか、などである。

一方、一旦分離された対話システムを実行するアーキテクチャとしては、分離した対話部と応用部をさらに機能単位の部品に分割し、それぞれの部品を並行動作する複数のエージェントとして実行し、エージェント間をメッセージあるいはイベントの交換により結合するマルチエージェントモデルが自然であり、共通の基盤となりつつある[Cout89]。

さて、筆者は3年ほど前から、UNIX上で動作する電子メイル・ニュースリーダvinを開発している[佐藤88]。このツールは、多数のユーザからの要求を受けて頻繁に変更され、改良されてきた。しかし、機能と利用者インターフェースが明確に分離されていないため、インターフェースの改良や大幅な置き換えが、困難になってきた。現在数百人の利用者が利用している既存のインターフェースは、保持しなければならない。この制約の中で、全く新たな形式のインターフェースを実験するために、また既存のカスタマイズ機能に収まらないインターフェースの作成を利用者に委ねるために、「対話独立性」が切実な要求となっている。

そこで本稿では、2.で対話独立性の実現によって得られる利点とUIMSへの要求を考察する。3.では、マルチエージェント型のシステムの開発改良支援および実行環境として設計した、viaシステムについて述べる。4.ではviaシステムの適用例として、幾つかのニュースリーダの利用者インターフェースを作成した例を示す。

2 対話独立性への期待

対話部と応用部を独立させ、両者の結合をUIMSによって系統的に管理することにより、様々な利点が得られると期待される。

(1) 多様なインターフェース

一つのアプリケーションに対して多様なインターフェースを実現できる。アプリケーションが提供する機能は同一であっても、そのプレゼンテーションのための資源や技術は進歩する。近年では特に、マルチメディアへの対応が課題となっている。対話独立性はこのような進歩に追随して行くことを容易にする。逆に、古いインターフェースしか使えない(例えばキャラクタ端末とモデムでコンピュータにアクセスするような)状況に対応することも、現実の場面では必要になる。

(2) 一様なインターフェース

多様なアプリケーションに対して一様なインターフェー

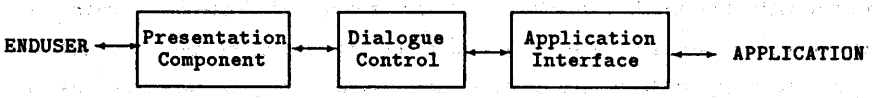

図 1: Seeheim Model [Gree85]

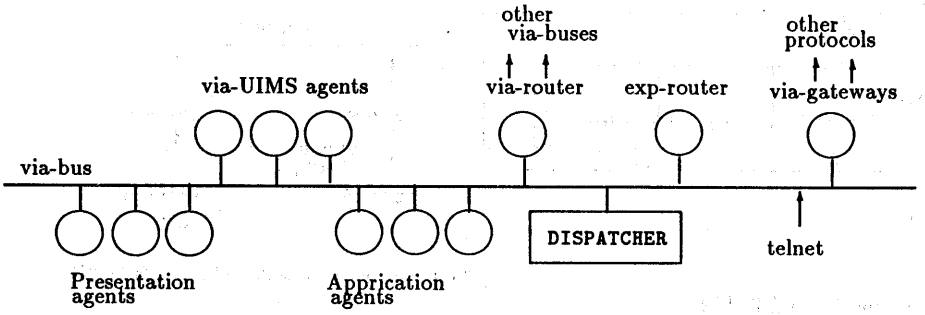

図 2: via-UIMS のアーキテクチャ

スを実現できる。統一された操作モデルに基づくインターフェースは、利用者の学習の労力を減少させ、使い易さを向上させる [Shne87]。分離された対話部の構成法に対して共通の枠組を与えることで、この統一化を支援することができる。

(3) 対話部のみの独立した開発と改良

対話システムの使い易さは、実際に使ってみることなしに評価するのは難しい。使いやすさの評価を、繰り返しフィードバックしながら洗練して行くうえで、分離された対話部のみの効率的な修正ができることで、開発効率は高まる。さらに応用部が実現されていない段階で、対話部だけを試作して、対話部を評価するプロトタイピング手法も試みられている [Hart89][Burg89]。

(4) 記述言語の分離

多様な応用部と対話部の全ての部分を記述するのに最適な、単一の言語があるとは考えにくい。特定の言語の機能を前提にしない分離と結合を行なうならば、両者は別々の言語で記述することが可能になる。また、対話部の記述を、非手続き言語、図的言語などにより記述できるようにすれば、計算機の専門家以外による対話部の開発と改良、あるいはエンドユーザによるカスタマイズを支援できる。

(5) 再利用

注意深く分離された対話部品や応用部品は、共有し再利用することができる。既存の (対話部と応用部が一体として作られた) システムに対しても、UIMS へのインターフェースを組み込んで、応用部を再利用しつつ新たなインターフェースへ対応できるよう、支援することができる。

(6) 機種独立性および分散透明性

Xwindow システムに代表される、プレゼンテーション部品と応用部品を分離して、クライアントサーバモデルによって分散透明に結合する方式は、ハードウェアや OS の環境を越えて統一されたインターフェースを提供し、また負荷を分散するために有効である。

実用の UIMS を考えるとき、(4),(5) にあげたような、複数の言語に開かれたシステムであること、既存の部品を再利用しやすいこと、などは無視できない重要な要求である。

3 via-UIMS の設計と試作

UIMS のアーキテクチャは、分離された対話部と応用部、さらに分割された対話部品と応用部品を結合し、実行を制御する機構を与える。以下に、現在試作中の via-UIMS のアーキテクチャの概要を示す。

3.1 via-UIMS のアーキテクチャ

via-UIMS は、エージェント間を接続する機構 via-bus と、via-UIMS エージェントの組からなる (図 2)。分割された応用部品、対話部品はそれぞれ、UIMS 部品とともにエージェントとして via-bus に接続する。このようなマルチエージェントモデルによる構成により、マルチスレッド型の対話の実現を支援し、利用者に見えない (バックグラウンドでの) 自動的な処理の記述を支援する。

エージェント間の接続機構である via-bus は、(論理的に) バス型の接続メカニズムである。これは、不定種類、不定数の構成要素の間の不定形態の接続を可能にするものである。エージェントは随時 via-bus と接続し、他のエージェントとメッセージを交換し、via-bus から離脱する。実際の実現では、エージェントが発したメッセージはディスパッチャに送られ、ディスパッチャが配送を行なう。exp-router は与えられたメッセージの式に従ってメッセージの経路制御を行なうことにより、間接的にエージェント間の接続を管理する。一つのエージェントは複数の via-bus に接続することができる。3.3 に述べる via-router はその例である。

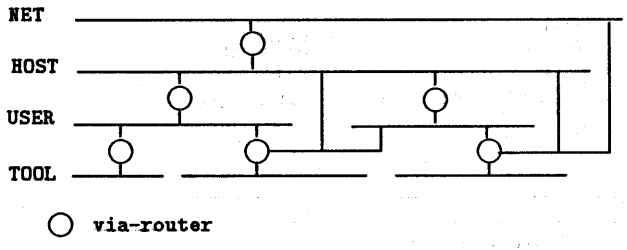

図 3: via-bus の階層型構成例

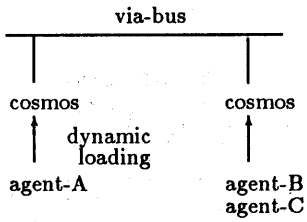

図 4: cosmos によるエージェント開発支援

viaではエージェント間の実行制御に関する主従関係を規定していないので、制御の中心は、対話部にあってもよいし、応用部にあってもよい。しかし基本的な枠組としては、制御の中心はこのどちらにもなく、対話部品と応用部品はともにサーバとして、それぞれ対話機能、応用機能を提供する。そして、具体的な対話の記述がクライアントとなって、これらのサーバを呼び出しながら実行を制御する。UIMS 部品は、これらの要素の構成や状態を管理し、実行を制御する。 これは [Hart89] において均衡型制御 (balanced control) と分類されている方式にあたる。

3.2 via プロトコル

via-bus は、コネクションレスな 1:n の双方向通信を提供している。拡張の容易な柔軟な UIMS を実現する上で、このような動的な結合機能は重要である [Cout89]。

(1) メッセージ形式

メッセージは任意長のバイト列である。ヘッダ部分にメッセージ長、戻り先のスタック、返信の要/不要などの情報を含み、本体はコマンドおよびデータからなる。ヘッダ中の戻り先スタックは、多段のサブルーチンコールと同様な、多段のエージェント間のメッセージ送信/返信のために用いられる。本体のデータの構造は via-bus レベルでは透過的であり、一方エージェントはヘッダ部分には基本的にアクセスしない。

(2) ブロードキャスト

エージェントは、送信するメッセージの送り先エージェントを指定しない。したがって、送信メッセージはすべてブロードキャストされる。ただし、エージェントは受信したメッセージに対して返信メッセージを送ることができ、この場合には宛て先のエージェントの ID が自動的にヘッダに挿入される。

(3) メッセージのフック

エージェントが接続している via-bus の中で、指定したメッセージを排他的に受信するよう指定できる。エージェントは、受信メッセージを加工してから再送出する、あるいは消滅させる。これにより、他の受信者に変更を加えることなく、新たな機能を挿入することができる。

(4) エージェントの置き換え

via-bus の中でエージェントは、エージェント名を名乗り、同一の名前のエージェントが接続されているとき、以前のものを切断する機能がある。これは、エージェントを新しい版で入れ換える場合などのために使用される。

3.3 複合型の via-bus 構成

「利用者環境」全体を統合的に管理するには、一つの対話ツールを制御するだけでなく、複数のツール間、あるいは複数ユーザ間での環境の共有や同期の制御を行なう必要がある。このために、via-bus は、図 3 に示すように階層的に構成することができる。

via-bus 間の接続は via-router エージェントにより行なう。via-router は、この via-bus から他の via-bus に対して送出すべきメッセージ、および他の via-bus からこの via-bus に取り込むべきメッセージを判断して、中継する。via-router 以外のエージェントは、via-bus の階層構造やネットワーク上での分散を意識しなくてよい。

3.4 開発支援環境

各部品 (エージェント) の開発を支援するエージェントとして、cosmos プログラミングシステム [佐藤 89] を利用することができる (図 4)。cosmos は、C 言語インタプリタや動的リンク機能により、C プログラムによるエージェントの開発を支援する。また、対話システムでは、利用者の扱う外部表現 (言語) と、アプリケーションで扱う内部表現の変換が頻繁に行なわれる。cosmos では、ごく小規模の言語処理系を多数使用するような対話システムにおいて、言語処理系の作成支援環境および

実行時のパーサ / インタプリタサーバとして使用することができる。

3.5 via-UIMS エージェント

via-UIMS エージェントの中心となるものは、via-DB エージェントである。これは、対話システムのエージェント構成情報、現在の実行状態、利用者の属性などを管理する分散データベースである。

対話部品と応用部品の結合と実行制御は、任意の言語で記述することができるが、既存言語による開発を via-UIMS で充分に支援することは難しい。そこで現在、この記述専用の言語と処理系 (エージェントとして接続される) を設計している。この言語の基本的な設計方針としては、システムの実行状態は via-DB に格納し、システム全体の実行を制御するエージェントには状態を持たせず、メッセージからメッセージへの変換器 (フィルタ) の組み合わせとして記述する。

3.6 試作版 via

via プロトコルを、UNIX 上の socket ベースのプロセス間通信機能を用いて試作した。1 つの via-bus は、接続口となる 1 つの socket と、これにサーバプロセスとして接続されたディスパッチャで実現した。各エージェントは、接続したい via-bus を表す socket に対して、クライアントプロセスとして接続する。自力で socket への接続を記述できない言語 (例えば shell や awk のスクリプト) のプログラムは、viattach と呼ぶエージェントの子プロセスとして接続することにより、標準入出力を介して via-bus に接続することができる。

エージェントは、受信を希望するメッセージがあれば、その形式をディスパッチャに登録する。ディスパッチャは、どのエージェントがどのメッセージの受信を希望しているかを管理するテーブルを持ち、これに従ってエージェントから送られたメッセージの転送や返送を行なう。

via プロトコルは、ディスパッチャへのコマンドセットとして表されている。エージェントはディスパッチャにメッセージを送出し、ディスパッチャはメッセージがいくつのエージェントに転送されたかをメッセージとして返す。Sparc-Station/330 で測定したところ、ディスパッチャへのコマンド送出と応答の受信の性能は、約 200 回 / 秒となっている (現在はストリーム型で通信している)。

4 ニュースリーダへの応用

コンピュータネットワークの普及に伴って、電子メイルや電子ニュースを読み書きするための数多くのツールが作られてきた。例えば、readnews, rn, vn, nn, vnews, gnus/Nemacs, ucb-mail, elm, mh, mush, rmail/emacs などである。これらのうちの多くが、ソースプログラムが公開され、コンピュータネットワーク上で無料配布されている。まえがきに述べたように、筆者も vin と呼ぶツールを開発してきた。

これら幾多のツールは、プレゼンテーションは異なっても、機能の大部分は共通である。にもかかわらず、これらのツールは互いにほとんど無関係に作られてきた。これは明らかに労力の無駄である。プレゼンテーションと機能を分離して、機能を共有することが必要であることを示すよい例と言える。また、ニュースを扱うツールとメイルを扱うツールとは互いに、扱う対象も、プレゼンテーションの形式も類似である。にもかかわらず、やはりこれらを統合して扱えるようになっていない (vin は例外である)。これも、プレゼンテーションを共有すれば労力の重複がはぶけるだけでなく、ユーザに対しても共通の一貫したインターフェースを提供することができる。

この種のツールにおいて、機能とインターフェースの分離が全く行なわれて来なかったわけではない。個々のユーザの管理などを含まない、ニュースシステムへのアクセス機能に関しては、これをサーバ・クライアント方式でネットワーク透明に提供する nntp プロトコル [Kant86] が作られ普及している。また mh(message handler) では、メイル操作のためのプリミティブセットを unix のコマンド群として提供し、この機能セットを共有して多様なインターフェース (コマンドセット, 画面指向インターフェース, emacs への組み込み, Xwindow 対応, など) が作られている。いずれも、分離によって機能の共有が有効に働いている例といえる。

このような望ましい分離を、ニュースとメイルあるいはその他の情報に対して総合的に提供するにはどうしたらよいか？その方法の一つは、特定の利用者インターフェースを前提とせず、ニュースやメイルへのアクセスおよびユーザのアクセス履歴の管理機能を提供する、抽象ニュース (メイル) リーダを作成することである。そのアクセスインターフェースは、例えば nntp の上位互換にすることが考えられる。以下では、そのような抽象ニュースリーダの実現の試みについて述べる。

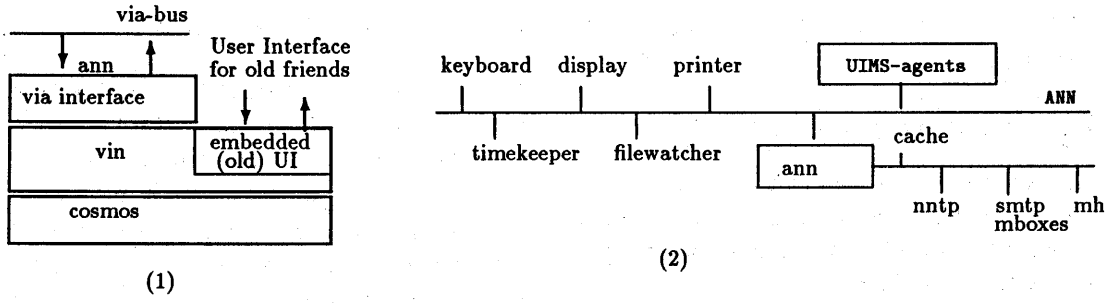

図 5: 抽象ニュースリーダ ann とその環境 ANN

4.1 抽象ニュースリーダ ann

試作中の抽象ニュースリーダ ann(Abstract Network Newsreader) は、vin に via インターフェースを加えたものであり、vin は cosmos システムの上で動作する (図 5-(1))。したがって ann の機能は、vin の機能のサブセットである。これは、vin の旧来の利用者インターフェースをそのまま残して、新たなインターフェースを付加する実験でもある。

ann は、ニュースとメイルに対する統一的なアクセスと、利用者履歴の管理機能を提供する。主なものは、ニュースグループあるいはメイルフォルダの (階層的) 一覧 (記事数や購読状況)、ニュース記事やメイルへのアクセスおよび検索、記事に対する印付け (未読/既読など) と、印された記事数のカウント、記事およびグループのソート、記事およびグループへの訪問のヒストリ、unix(MBOX) 形式および mh 形式のメイルフォルダの管理、記事のフォーマット変換 (画面表示用, 印刷用, 返信用など)、グループの階層構造の定義 (利用者による再定義)、利用者間の共有キャッシュで記事へのアクセスを高速化する機能、などである。

利用者インターフェースを ann の外で記述するために、keyboard や display などの要素は、ann とは完全に独立なエージェントとして実現されている (図 5-(2))。この他、ann の外部での事象の発生、ここではファイルの更新、を知らせるための timekeeper や filewatcher エージェントがある。この、ann と周辺要素の組を以下、大文字で ANN と記す。これらのエージェントはいずれも、LAN 上のどのマシンにあってもよい。また、複数ユーザ間でエージェントを共有するように via-bus を構成することができる。

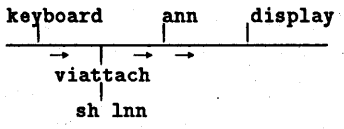

図 6: 行指向ニュースリーダの構成

```
#!/bin/sh
# rn LIKE NEWS READER INTERFACE
Sendmsg(){ echo $MSG; echo "."; read RESP; }
while read COM ARG; do
 case "$COM" in
 "INIT") MSG="MYNAME lnn";                    Sendmsg;
         MSG="SENDME KEYIN";                  Sendmsg;
         MSG="SENDME NEXTPAGE_ON_EOF";        Sendmsg;
M1="ANN FORMAT GROUP_LINE: '******** '%3<unread>"
M2="' unread article in '<group>'--read now? [ynq] '"
         MSG=$M1" "$M2;                       Sendmsg;;
 "KEYIN")
  case "$ARG" in
   "^L") MSG="DISPLAY_CONTROL CLEAR";         Sendmsg;
         MSG="EXP ANN PRINT|DISPLAY";         Sendmsg;;
    " ") MSG="DISPLAY_CONTROL PREVPAGE";      Sendmsg;;
    "b") MSG="DISPLAY_CONTROL NEXTPAGE";      Sendmsg;;
    "g") MSG="EXP GETLINE|ANN ARTICLE|DISPLAY";
                                              Sendmsg;;
    "n") MSG="EXP ANN NEXTARTICLE 1|DISPLAY";
                                              Sendmsg;;
    "p") MSG="EXP ANN NEXTARTICLE -1|DISPLAY";
                                              Sendmsg;;
    "q") MSG="CLOSE";                         Sendmsg;
         exit;;
  esac;;
 "NEXTPAGE_ON_EOF")
         MSG="EXP ANN NEXTARTICLE 1|DISPLAY";
                                              Sendmsg;;
 esac
done
```

図 7: shell スクリプトによる lnn の記述例

4.2 インターフェースの作成例

ann を用いて、いくつかのニュースリーダのインターフェースを部分的に実現してみた。

(1) 行指向ニュースリーダのインターフェース

ANN を用いて、まず、行指向のニュースリーダ (lnn) を記述した (図 6)。これは、ニュースリーダ rn の幾つかのコマンドを、sh コマンドのスクリプトで記述し、viattach により via-bus に接続したものである。lnn は、ANN が提供している各種のエージェントを接続するだけのフィルタであり、keyboard から入力を受けとり、ann にコマンドを送り、その出力を display に渡す。lnn は単体の sh スクリプトとして実行し、テストできる。

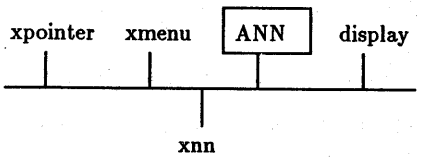

図 8: ウィンドウシステム対応の構成

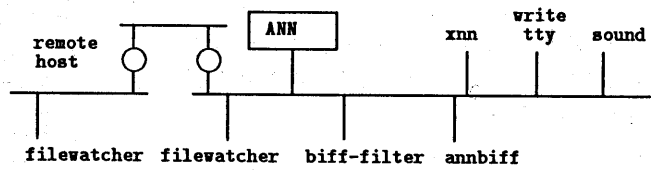

図 9: 新着メイル・ニュースを報知する構成

(2) 画面指向ニュースリーダのインターフェース

画面指向のニュースリーダでは、画面上に表示された情報をカーソルなどにより選択してシステムにフィードバックする点が、行指向のものとの最大の違いである。これを実現するために、display にカーソル移動機能およびカーソル位置報告機能を付加した。その他の機能やインターフェースは (1) のものを再利用している。

(3) ウィンドウシステムへのインターフェース

X ウィンドウシステムへのインターフェースを持つ xnn の構成を図 8 に示す。まず、画面上の文字カーソルと、マウスによるポインターを一致させて動かすために xpointer を加えた。xpointer は、コマンドで指定されたウィンドウ内で、ポインター位置が移動したときにメッセージを送出する。逆に、(キーボードからのコマンドなどによる) 文字カーソル位置の移動時にポインター位置を一致させる。xmenu は、メニューの生成消去コマンドを受け付け、作成したメニュー上でボタンが押されたときにメッセージを送出する。

xpointer, xmenu, および display は互いに依存しておらず、これらが送出するメッセージを xnn が仲介することによって、全体として動作する。したがってこれらのエージェントは他の応用、例えばテキストエディタなどにも再利用することができる。

4.3 外部環境との接続

新しいニュースあるいはメイルの到着を、利用者にいかに知らせるかを考える。利用者はログインしていないかも知れないし、逆に現在ニュースリーダを使用中かもしれない。また、LAN 上のどのマシンにログインしているか、どのような種類の端末を使用しているかも不定である。このような、対話システムの外の世界を含む総合的な利用者環境は、従来の UIMS のスコープの外にあったが、via ではこのような環境の総合的な管理を、重要な応用対象と考えている。

図 9 に、これを ANN を利用して実現した例を示す。

filewatcher は、ニュースやメイルのスプールに変更があったときに、それを知らせるメッセージ FILECHANGED を送出する。スプールがリモートホストにあったり、複数のホストに分散している場合には、filewatcher をそれに合わせて配置する。ann は FILECHANGED を受けとると、関連するメイルやニュースのフォルダを検査し、新着記事の ID を含むメッセージ (NEWMESSAGE) を送出する。annbiff は NEWMESSAGE を受けとり、利用者に知らせる。

利用者に知らせる方法は、様々なものが考えられる。ニュースリーダで ANN にアクセス中ならそのリーダが処理する、ユーザがログインしているならその端末にメッセージを出力する、音声出力が可能なら音声で知らせる、などである。

利用者は、特定の到着記事だけを知らせて欲しいかもしれない。この場合には、biff-filter エージェントを挿入する。biff-filter は、NEWMESSAGE を「フック」し、利用者に知らせるべきメッセージと判定したときに NEWMESSAGE を送出する。

4.4 対話の監視

利用者と対話システムの間の通信は全てメッセージとして via-bus 上を流れる。これを監視して管理することにより、利用者に対して有益な情報をフィードバックすることが可能である。

具体的な例をひとつ挙げる。vin の利用者のコマンド利用頻度を調べてみると、1 行単位のカーソル移動 (j,k) が、多くの利用者で全コマンド入力数の 8 割前後を占めている。ページ単位のカーソル移動機能 (Space,b) を知らないために、このような使い方をしているユーザもいる。そこで、コマンド入力パターンを監視して、k,j と Space,b の利用頻度がバランスしていないようなら、k,j が連続して使用されたときに、Space や b を利用するようアドバイスするといった機能を実現することができる。

また、システムから利用者に送られるさまざまな情報 (ヘルプメッセージやガイド、動作状況報告やエラー

メッセージ、など）を系統的に管理して、情報の提示形式を利用者ごとのレベルや利用環境に適合させたり、対話的にチュートリアルを行なうなどの機能の実現を、支援することできる。

5 おわりに

開発中のユーザインターフェース管理システム via のアーキテクチャと応用例について述べた。via はまだ開発をはじめたばかりの段階にあるが、例題のプログラムを作成してみてその有効性を感じることができた。そのひとつは、変更に対するターンアラウンドの短縮である。起動に時間のかかるタイプの応用 (vin もその一つ) では、via を用いることで、応用部は実行状態のまま、対話部だけを入れ換えてはテストできるので、ターンアラウンドが非常に短縮され、快適である。実際に利用される段階においても、応用部を実行状態のまま常駐させて対話部のみを起動・終了できるようにすることで、エンドユーザレベルに応答性の向上をもたらすことができる。

今回の試作では、各エージェントをプロセスとし、エージェント間をプロセス間通信で接続した。このような実現方法でも充分、実用に耐えるほど、現在のハードウェアは高速化していることに、一種の驚きを感じた。もちろん、タイマーからのミリ秒間隔のメッセージのように、通信が明らかなネックになる場合はある。このような場合には、通信部分をサブルーチンコールに展開して、エージェントを併合するようなトランスレータあるいはライブラリを作る必要はあると考えている。それは同時に、via 上で開発したツールを via から離れて利用したい場合にも役立つであろう。

全ての利用者にとって最も使いやすいようなインターフェースは存在し難い。利用者の過去の利用歴や知識、あるいは好みにより、使いやすさの評価基準は異なる [Kris90]。ひとりの利用者を見ても、徐々に変わっていくものであり、その変化に適応してゆくこと、さらに、より効率的な使い方の習得を助けることが必要である。via-UIMS はそのような適合型の利用者環境の実現のための基礎を提供できると考えている。

謝辞 本研究の機会を頂いている棟上昭男情報アーキテクチャ部長に感謝します。また、vin に関して有益なコメントやレポートを寄せていただいているユーザの皆さんに感謝します。

参考文献

[Gree85] Green,M.:
The University of Alberta User Interface Management System.
Proc. of SIGGRAPH '85, 12th Annual Conference (1985), pp.205-213.

[Burg89] Burgstaller,J., Grollmann,J. and Kapsner,F.:
A User Interface Management System for Rapid Prototyping and Generation of Dialog Managers,
in [Salv89] (1989), pp.533-540.

[Cock89] Cockton,G.(editor):
Engineering for Human-Computer Interaction,
Proc. of the IFIP TC 2/WG 2.7 Working Conference on Engineering for Human-Computer Interaction (1989).

[Cout89] Coutaz,J.:
Architecture Model for Interactive Software: Failures and Trends,
in [Cock89] (1989), pp.137-153.

[Hart89] Hartson,H.R. and Hix,D.:
Human-Computer Interface Development: Concepts and Systems for Its Management,
ACM Computing Surveys. Vol.21, no.1 (Mar.1989), pp.5-92.

[Kant86] Kantor,B. and Lapsley,P.:
Network News Transfer Protokol - A Proposed Standard for the Stream-Based Transmission of News, RFC977 (1986).

[Kris89] Krishnamurthy,B.:
BOSS: A Visual Interface for a Workstation Environment,
in [Cock89] (1989), pp.257-271.

[Manh89] Manheimer,J.M, Burnett,R.C and Wallers,J.A:
A Case Study of User Interface Management System Development and Application,
CHI'89 Proceedings (May.1989), pp.127-132.

[Salv89] Salvendy,G. and Smith,M.J.(editors):
Designing and Using Human-Computer Interfaces and Knowledge Based Systems (Proc. of the Third International Conference on Human-Computer Interaction), Elsevier Science Publishers B.V. (1989).

[Shne87] Shneiderman,B.:
Designing the User Interface,
Addison-Wesley Publishing (1987).

[Tell90] Telles,M.:
Updating an Older Interface,
CHI'90 Proceedings (1990), pp.243-247.

[佐藤 88] 佐藤、小島、田沼、植村:
内容を解釈実行するニュースメイルリーダ,
電子情報通信学会 データ工学研究会 DE88-23 (1988).

[佐藤 89] 佐藤:
多重言語を指向するプログラミングシステムの開発経験,
情報処理 Vol.30, No.4 (1989), pp.430-438.