Comic Chat を用いた情報提示手法の提案

Visualizing a Society of Proactive Agents with Comic Chat

柴山悦哉[†] 酒井隆行[†] 高橋伸[†]
Etsuya SHIBAYAMA Takayuki SAKAI Shin TAKAHASHI

†東京工業大学 情報理工学研究科 数理・計算科学専攻
Graduate School of Information Science and Engineering, Tokyo Insititure of Technology

概 要

Proactive なエージェントからなる社会として構成されたシステムを可視化表示し，さらに，ユーザもエージェントの社会の一員として対話に参加できるようなユーザインタフェースの構成法を提案する．具体的には，各エージェントをコミックのキャラクタとして表示し，エージェント間およびユーザとエージェントの間の対話を，チャットのインタフェースを用いて表現する．このような考え方を実現するために，Comic Chat を利用した．

1 はじめに

今日では多くのコンピュータシステムがデスクトップメタファと WIMP(ウィンドウ，アイコン，メニュー，ポインティングデバイス) に基づく GUI を採用している．MacOS や Windows に代表されるこれらの GUI は，(1) コンピュータシステムが管理する資源 (resource) をビジュアルなオブジェクトと捉え，(2) これら各オブジェクトに，画面上でのアイコン表現を与えるとともに，(3) その表現に対してポインティングデバイスを用いた直接操作を行なうことにより，(4) コンピュータシステムが管理する資源に対するさまざまな操作を実現している．

今日のコンピュータシステム (特にソフトウェアシステム) をオブジェクトの集まりとして捉えるのは自然な考え方である．そして，ユーザからコンピュータシステムへの入力を，オブジェクトへの直接操作により実現するのも，これもまた自然な考え方であろう．

一方，近年の傾向として，コンピュータのソフトウェアシステムを自律的なエージェントの集まりとして構成する方式が注目を集めている．ここでは，オブジェクトとエージェントの違いを，以下のように考えることにする*．

- オブジェクトとは，受動的なデータを抽象化したものであり，reactive に動作する．
- エージェントとは，能動的な動作を抽象化したものであり，proactive に動作する．

エージェントが注目される理由の一つは，実行環境からある程度独立した永続的な動作を表現する抽象として便利だからである [1].

2 メタファとしてのエージェントの社会

本稿で議論したいのは，デスクトップメタファと WIMP に代表される今日の GUI 構成法が，エージェントの集まりとして構成されたコンピュータシステムに対する操作のインタフェースとして有効であり続けるかという問題である．ファイルやフォルダに代表される受動的なオブジェクトと比べ，エージェントは，より能動的でより擬人的で，そして，その動作が重要な意味を持つ．単純に画面上のアイコンとしてエージェントを表現するのが良いとは限らない．

本稿では，エージェントの擬人性に注目し，コン

＊ この定義が常に正しいわけではないが，便宜的にこのような 2 種類の存在を考えると便利である．

ピュータシステムを擬人的なエージェントからなる社会と捉えることにする．そして，その社会の様子をユーザに提示するとともに，社会の活動にユーザが参加することにより，コンピュータシステムの操作を実現するメタファを提案する．

我々は，このメタファを実現するために Comic Chat [2] を用いたシステムを試作中である．エージェントの擬人性を表現するために，まず，エージェントの人格をビジュアライズしたい．これを，各エージェントにコミックのキャラクタを割り当てることにより実現する．そして，ユーザは，コミックとして表現されたエージェント間の対話を読むことにより，エージェントやエージェントの社会の状態を知ることができる．また，ユーザもチャットに加わることが可能であり，エージェントと同じ資格でエージェントの社会に参加することができる．

チャットシステムはもともと複数のユーザがおしゃべりを行なうためのものである．したがって，一つのチャットルームに，人間のユーザが複数参加するのは自然なことである．対話のための適切なプロトコルや言語さえ設定できれば，複数の人間と複数のエージェントによる対話的共同作業のためのプラットフォームとして利用できると考えられる．

3 Comic Chat を用いた表現の例

現在の時点では，まだ自律的なエージェントの集まりとして構成されたソフトウェアシステムはあまり多くない．しかし，エージェントの本質を永続的な動作やサービスを抽象化したものと捉えるなら，今日でも，以下のようなものはエージェントととして捉えることができる．

- ネットワークに接続された個々のコンピュータ
- サーバやデーモンが提供するさまざまな機能

今日の分散システムの状態や動作をエンドユーザに提示する場合，コンピュータ，サーバ，デーモンなどを擬人的に表現するのは，一つの有望な手法である．

このような考え方に基づき，システム管理の分野に対象をしぼったシステムを試作中である．システム管理の分野では，システムのモニタリングを行なう機能などが永続的な動作とみなせ，エージェントという抽象を用いて表現するのが妥当である．エージェントとユーザの間のインタラクションにはさまざまな形態が考えられるが，現在のところ，次のようなものに絞って，表現方法を検討している．

- お知らせ型の情報提示
- 実況中継型の情報提供
- 提案型の情報提供

最初のものは，notification に相当する．ユーザに対する notification は，今日の通常のデスクトップ環境では，メイルの到着や予定管理システムのアラーム機能などで利用されている．図 1は，システム管理で用いられる notification の一例として，ディスクのパーティションがいっぱいになったことを知らせている．

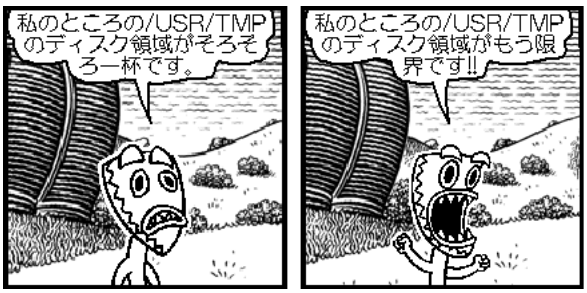

図 1　ディスク残量の警告

図中に現れるコミックのキャラクタがディスクの状態を監視するエージェントを表現している．この図では，事態の緊急度を表情で表している．左側のコマの表情は，パーティションがほぼ一杯になったが，まだ若干の余裕がある場合を表している．一方，右側のコマの表情は，本当にパーティションがあふれそうな状態を表している．通常の監視システムでよく用いられる，色やアイコンなどによる緊急度の表現†に比べ，より擬人的な手法が用いられている．

実況中継型の情報提供も，やはりキャラクタの表情とせりふを用いて行なわれる．お知らせ型と違うのは，定期的に新しいコマが生成される点だけである．図 2に示されているのは，システムの負荷を実況中継した例である．

この図の例は，通常なら Unix の xload や Windows NT のタスクモニタなどを用いてグラフ表示されるタイプの情報である．今日の通常のデスクトップ環境では，たとえば，株価の表示などにも実

† たとえば，緊急度の高い notification は赤，それほどでもなければ黄色の文字で表す方法などがある．

図 2　システム負荷の実況中継

況中継型の情報提示手法が用いられている．

ここまでに述べた２つの情報提供形態では，システムからユーザに対して事実が伝えられた．一方，提案型の情報提供では，ユーザが何らかの処置を行なわなければならないような事態が生じたときに，とるべき対策を提案したり，何らかのヒントを与えたりする．図 3は提案型情報提供の一例である．

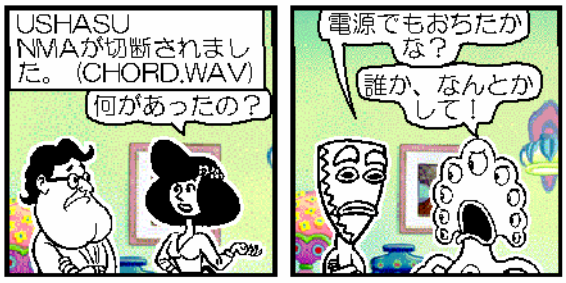

図 3　ユーザにヒントを与える

この図では，ushasu および nma という名前のホストとの通信ができなくなったことを伝える (左側のコマ) とともに，その原因を示唆する情報も伝えている (右側のコマ)．また，通信切断に伴うサービスの停止によって，被害を受けているエージェントが存在することも同時に伝えている‡ (右側のコマ)．

さて，この図に現れる 4 人のキャラクタのうち，一番左側の人物は，おそらく，システムを監視する (ソフトウェア) エージェントである．一方，残りの３人の発言内容をよくみてみると，これらがエージェントの発言なのか，チャットに参加している人間のユーザの発言なのか，必ずしも明らかでないことがわかる．たとえば,「誰か何とかして」という発言は，サービス停止にともない悲鳴をあげているホスト§ のものかもしれないし，同様の理由で悲鳴をあげている人間のものかもしれない．また,「電源でも落ちたかな」という発言は，ルールベースシステムにより故障診断を行なうエージェントのものかもしれないし，トラブルシューティングに一定の知識を持ったユーザのものかもしれない．

このような例から明らかなように，チャットを用いることにより，エージェントからの情報提供や提案とチャットに参加している人間のユーザからの情報提供や提案を統一的に表示するができる．このような環境では，故障診断エージェントからの提案があっても，熟練ユーザがその可能性を否定し，別のより適切な提案を行なうことも考えられる．

ここまでに紹介した内容は，基本的に，エージェントとユーザの直接対話を表現するものであった．一方，エージェント間の対話の状態をユーザに提示することにより，システムの状態を可視化することも考えられる．現在の実装ではまだ取り組んでいないが，プロトコルアニメーションのために Comic Chat を用いる可能性を検討している．これは，ホスト間 (あるいはサーバ機能とクライアント機能の間) でやりとりされるメッセージの様子をコミックにより表現するものである．たとえば，ルーティングの様子を可視化表示する場合なら，ホスト間をパケットが伝言のように伝わっていく様子を表すコミックにより表現する方法が考えられる．これは，通常なら traceroute コマンドを用いて獲得する情報を動作として示すことになる．

4　システム全体の構成

3 章で紹介した例は，チャットの断片であり，非常に局所的なものであった．まだ構想の段階で今後変更が行なわれる可能性もあるが，システムの全体像は，以下のようになる予定である．

- システム全体は，複数のチャットルームよりなる．話題の種類に応じてチャットルームを適切に選択し，対話を行なう，
- チャットルームの数が非常に多い場合は，階層的に組織化する．ネットニュースのニュースグ

‡　現在の実装では，提案のためのルールは，非常に単純なものしか導入しいない．つまり，インタフェースの妥当性を検証している段階であり，推論部が賢いわけではない．

§　たとえば，nfs … not responding というメッセージを表示するような状況を考えていただきたい．ちなみに，このような「悲鳴」をあげさせることで，機械的メッセージとは一味違った方法で，事態の深刻さのレベルを一般ユーザに知らせることができる．

ループのようなものと考えてほぼ間違いない．

- 各チャットルームには，エージェントと人間の対話の履歴が一つのコミックとして格納される．

チャットのログは一般にはかなり大きくなるし，内容も多岐にわたる．スケーラビリティの問題に対応するために，階層化は必須であろう．デスクトップメタファでも，机の上にのせられないくらい多くのオブジェクトを扱う場合には，フォルダを用いた階層化が行われている．

このような階層的チャットルームを構築すると，必要な情報を探し出す手間が問題となる．我々のグループでは，階層構造に対する探索のインタフェースとして HishiMochi [5] を提案している．HishiMochi は，複数焦点ズーミングインタフェース [6] に基づく検索インタフェースで，長方形の入れ子により図示された階層構造を対象とする．キーワードを入力すると，ヒットした箇所が拡大表示され，それ以外の箇所が縮小表示される．この技術を階層的 Comic Chat に適用すると，指定されたキーワードを含むチャットルームやコマだけを拡大表示することができるはずである．

5 他の方式との比較

エージェントの擬人性と社会性を表現するという観点からは，コミックではなく，Community Place [3] 等のような 3D のアバタ (擬人性) とダンジョン (社会性) を組み合わせる方式も考えられる．このような方式に対する Comic Chat の最大の利点は，履歴の一覧性である．過去ログの時系列をたどる時に，コミックなら斜め読みが可能である．一方，3D アバタとダンジョンを用いた場合，過去の履歴をたどるために 3D アニメーションの再生が必要になる．絵だけなら早送りで眺めることもできるが，せりふにも注目したい場合，高速の早送りは困難である．

コミックは 3D アニメーションに比べリアルでないという批判もあるかもしれない．しかし，リアルにした方が常に良い結果が得らるとは限らない．適度にデフォルメされている方が，万人にとって感情移入しやすい表現となる [4]．

今回提案した情報提供の形態は，通常のデスクトップ環境でもアドホックには実現されている．しかし，xload や株価の表示は，デスクトップ上で行われているものの，その内部までがデスクトップメタファにしたがっているわけではない．社会というメタファには，このようなものまで統一的に表現する能力があり，Comic Chat を用いると，完全にシームレスな表現が可能となる．

6 まとめ

本稿では，コンピュータシステムを proactive なエージェントの社会ととらえ，エージェントとユーザが効果的に対話するためのインタフェースに関する提案を行った．この方式は，以下のような特徴を持っている．

- 擬人的なエージェントをコミックのキャラクタにより表現する．
- エージェントによる永続的な動作を対話の履歴ととらえ，これをコミックとして表現する．
- (一般に複数の) 人間のユーザもチャットのプロトコルを用いることにより，エージェントと同等の資格でこの対話に加わる．

すなわち，擬人性，動作の永続性，複数ユーザと複数エージェントによる対話と協調作業を表現する枠組になっている．

参考文献

[1] J.-P. Briot, L. Gasser: Agents and Concurret Objects, IEEE Concurrency, Vol. 6, No. 4, pp. 74–77, 81, 1998.

[2] D. Kurlander, T. Skelly, and D. Salesin: Comic Chat, *ACM SIGGRAPH'96 Conference Proceedings*, pp. 225–236, 1996.

[3] R. Lea, Y. Honda, K. Matsuda, and S. Matsuda: Community Place: Architecture and Performance, In VRML97, pp. 41-49, 1997.

[4] S. McCloud: Understanding Comics: The Invisible Art, Kitchen Sink Press, 1993(邦訳：岡田斗司夫監訳:マンガ学, 美術出版社, 1998.).

[5] 豊田正史, 増井俊之, 柴山悦哉: HishiMochi: 非線形ズーミングを用いた動的検索システム, 日本ソフトウェア科学会 WISS'98 論文集 pp. 143–152, 近代科学社, 1998.

[6] M. Toyoda, and E. Shibayama: Hyper Mochi Sheet: A Predictive Focusing Interface for Navigating and Editing Nested Networks through a Multi-focus Distortion-Oriented View, *Proceedings of ACM CHI'99*, pp. 504–511, 1999.