97. 9. 24

情報処理学会　第５５回全国大会　公開パネル討論
「ソフトウェア開発新時代　ーネットワーク時代のソフトウェアテクノロジー」

エンドユーザ主導の時代

中所 武司
明治大学 理工学部 情報科学科
email : chusho@cs.meiji.ac.jp

今，インターネットやマルチメディアに代表される情報化社会の進展に伴い，エンドユーザが急激に増加しつつある．ネットワーク時代の主役はエンドユーザであり，エンドユーザ主導のアプリケーション開発の時代が迫っているという印象が強い．

業務の専門家（domain experts, business professionals）が利用するワークフローシステム，グループウェア，エージェントシステムなどの分散協調型ソフトウェアは，エンドユーザとなる業務の専門家自身が開発・保守しながら利用できることが望ましい．このようなアプリケーションは，規模的には小さいものが多いが，ネットワーク経由での複数のアプリケーションの連携などにより全体としては大規模化していく．最初は最小単位となる自律的個体から出発して，お互いの連携によって次第に大規模化していくような再帰的構成法がとられる．その過程でオブジェクト指向ベースのコンポーネントウェア，デザインパターン，アプリケーション・フレームワーク，ビジュアルプログラミング，エージェントなどの技術が利用される．

1．はじめに

今，新しい時代にふさわしい新しいソフトウェアの作り方が求められている．インターネットやマルチメディアに代表される情報化社会の進展に伴い，急激に増加しつつあるエンドユーザへの対応として，エンドユーザ主導のアプリケーション開発の時代が迫っているという印象が強い．

これまでは，ハードウェアに合わせてソフトウェアが開発され，そのソフトウェアに合わせてエンドユーザがコンピュータを利用してきた．この「初めにハードウェア（機械）ありき」という発想は，ハードウェアが高価だった時代にはそれを最大限に活用するために当然であった．

しかし，年間数百万台のパソコンが国内で売れているというインターネットの時代には「初めにエンドユーザ（人間）ありき」という発想へのパラダイムシフトが必要である．ネットワーク時代の主役はエンドユーザであり，筆者はこれを「コンピュータによる豊かな生活の実現」という意味で CS-life (Computer-Supported Life)と呼んでいる．

しかしながら，現在のエンドユーザのコンピュータ利用の形態は，電子メールやインターネットサーフィンあるいは文書処理のためのパッケージソフトウェアの利用が中心である．ここでもやはり人はソフトウェアに合わせてコンピュータを利用せざるをえない．

ネットワーク時代のエンドユーザは，自分の欲するアプリケーションを自ら開発できることが望ましい．豊かな生活の実現へのコンピュータ応用の代表例としては，身近な日常業務に関するルーティンワークの自動化がある．一般にエンドユーザコンピューティング

は幅広い概念を表わし，エンドユーザによるアプリケーションの利用を意味することが多いが，ここではルーティンワークからの解放のためのエンドユーザ主導のアプリケーション開発という視点で考察する．

2．エンドユーザ主導コンピューティング

2.1 エンドユーザの定義

一般に典型的なエンドユーザとして，以下の3種類が考えられる．

(a) 基幹業務の担当者
(b) 業務の専門家
(c) 一般のユーザ

最初は，ユーザ企業の基幹業務を担当している人達である．エンドユーザ部門に所属し，利用するソフトウェアはシステム部門が開発して提供してくれる．たとえば，銀行の窓口やチケットの販売窓口などでオンラインシステムの端末を操作している人達である．

2番目は，一般にオフィスワーカーといわれるような人達である．ネットワークに接続された専用のパソコンやワークステーションを持っており，業務内容に応じてDB検索や表計算やOLAPなどのために市販のアプリケーションパッケージを利用するが，自分でアプリケーションを開発することはしない．

最後は，オフィスや家庭における一般の人達である．たとえば，日常生活の中で銀行のATM端末を利用するような人達で，コンピュータ利用のための特別の訓練は受けていない．インターネットやマルチメディア時代の主要なユーザとなる．

ここでは，特に個々の業務専門家のルーティンワークのコンピュータ化という視点で，(b)のエンドユーザに注目する．

2.2 対象ソフトウェア

今後はオフィスで業務の専門家が利用するアプリケーションが重要になる．その多くは個人あるいは部門レベルで利用するワークフローシステム，グループウェア，エージェントシステムなどの分散協調型ソフトウェアが主となる．規模的には小さいものが多いが，ネットワーク経由での複数のアプリケーションの連携により全体としては大規模化することもある．

2.3 開発・保守形態

このようなアプリケーションの多くは市販のパッケージでは不十分だが，基幹業務用システムのように情報処理の専門家に開発を委託するほどのものではないことが多い．特に開発を委託する方法は，高価な開発コストに見合った効果が必須であり，少数で利用するようなアプリケーションの開発には向かない．また，このようなアプリケーションは稼働後も状況の変化に応じて頻繁かつ迅速な機能変更が必要なので委託開発にはなじまない．

そこでこの分野ではエンドユーザ主体の開発形態（図1）が必須となる．最終的にはエンドユーザが自らのアプリケーションを自ら開発し，自ら利用することを理想とするが，その実現に向けての技術課題は多い．当面はエンドユーザが主体でシステムエンジニアの助けを借りて開発するが，保守はエンドユーザだけで行うレベルが目標となろう．

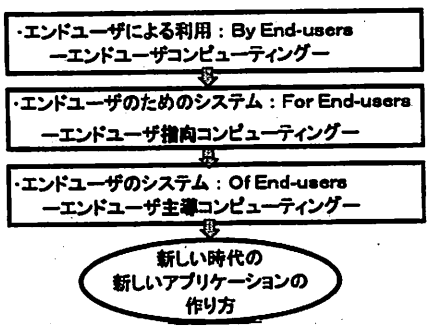

図1 エンドユーザコンピューティングの進展

3．ソフトウェア産業の形態

エンドユーザ主導の開発形態をソフトウェア産業の発展過程という視点でとらえるならば，図2に示すように労働集約型産業，知識集約型産業に続くものとして位置付けることができる．

ソフトウェア産業形態	主要な技術職	主要技術
労働集約型産業	プログラマ	自動化(CASE)
知識集約型産業	設計者	標準化(パッケージ)
ポスト知識集約型産業（知恵集約型産業）	業務専門家	エンドユーザコンピューティングツール

図2　ソフトウェア産業の形態

3.1　労働集約型産業とプロセスの自動化

かつてソフトウェア産業は労働集約型産業であった．それは1980年代の10年間の年間売上高の増加（8.8倍）が従業員（プログラマ）の増加（4.9倍）によって達成されてきたことに象徴される．生産コストあたりの生産量（ステップ数／人月）という生産性の向上のために，この時期のソフトウェア生産技術は，個別の開発手段（ツール）の高機能化からそれを用いたプロセスの自動化へと発展した．

3.2　知識集約型産業とプロダクトの標準化

ソフトウェアの生産性は本来は次のように定義されるべきである．

生産性＝生産物の価値／生産コスト

この「生産物の価値」はユーザの視点で決まるべきものであり，業務の知識と情報処理技術に基づくソフトウェア・パッケージが重要である．1990年代にはアプリケーションを個別に開発するメインフレーム文化に代わって，異機種ネットワーク間での移行性や接続性を重視したパッケージが流通するオープンシステムの文化が急激に広まった．ミドルウェアと呼ばれる基本的なソフトウェアが先行する形でソフトウェア・アーキテクチャの標準化が進み，プロダクトの標準化を促進した．

3.3　ポスト知識集約型産業とエンドユーザコンピューティング

情報処理システムが業務革新や経営戦略の実現手段として用いられるようになると，ソフトウェア開発の効率よりも効果が重要視される．何を作るかが最も大事であり，それを決めるのは業務の専門家である．業務の専門家が知恵をしぼって効果的なアプリケーションをタイムリーに作っていくためには，エンドユーザ自身が開発し，保守，拡張していく必要がある．

その実現のためには，自動化ツールや標準パッケージなどの情報処理技術を統合した上に，きめ細かく分類された応用分野対応の業務の知識に基づいたアプリケーション・フレームワークやコンポーネントウェアが必須である．分野別のアプリケーション・フレームワークの中にネットワーク上で流通する部品を取り込んでアプリケーションを作成するようになってきている．

潜在ユーザとしての業務の専門家の数を考えると，この市場は現在の情報サービス産業の数兆円規模の市場を大きく上回ることになる．ポスト知識集約型産業は，業務の専門家の知恵をいかにコンピュータ化していくかという意味において知恵集約型産業ともいえる．

4．アプリケーション構築技法

4.1　自律的個体の再帰的構成

先に述べたように，個人あるいは部門レベルで利用するワークフローシステム，グループウェア，エージェントシステムなどの分散協調型ソフトウェアは規模的には小さいものが多いが，ネットワーク経由での複数のアプリケーションの連携などにより全体としては大規模化していく．

次の式に示すように，このようなアプリケーション<A>は最初は最小単位となる自律的個体<a>から出発し，お互いの連携によって次第に大規模化していくような再帰的構成法がとられる．

<A> ＝ <a> | <A>の集合

たとえば，<a>と<A>をオブジェクトと複合オブジェクト，クラスやコンポーネントとデザインパターンやアプリケーション・フレー

ムワーク，あるいはエージェントとマルチエージェントなどに対応させてみるとよい．

本稿で注目するルーティンワークの自動化という観点からは，自分の日常的な業務を代行させるエージェントとエージェンシーに対応させたい．実際にはコンポーネントやオブジェクトなどの１個以上の組み合わせで最小単位のアプリケーションであるエージェントを構成し，さらにそれらのエージェントの連携でマルチエージェント化した分散協調型アプリケーションであるエージェンシーを再帰的に構成していく．

4.2 エンドユーザ向け開発環境とアプリケーション・アーキテクチャ

今後のエンドユーザ向け開発環境とアプリケーション・アーキテクチャの関係の例を図３に示す．アプリケーションは大まかには，ユーザインタフェース，モデル，コンポーネントウェアの３つの構成要素からなる．ユーザインタフェースは外界（ユーザ）との対話処理部分であり，モデルがアプリケーションに固有の処理を行う本体である．コンポーネントウェアには分野に共通の基本部品と特定業務分野向きの部品がある．後者が充実してくるとエンドユーザによるアプリケーション構築が容易になる．

再帰的構成法に対応させると，モデルとコンポーネントウェアが全体と部分の関係になる．ユーザインタフェース部分は外界とのインタフェースを有するときのみ必要となる付加的な部分である．

そのための開発環境の主な構成要素は，ビジュアルプログラミングツール，コンポーネント構築ツール，共通プラットフォームである．ビジュアルプログラミングツールはエンドユーザ自身が開発・保守に用いるもので，モデル構築ツールのほかにユーザインタフェース構築ツール，スクリプト言語，コンポーネントウェア利用支援ツールなどが考えられる．

コンポーネント構築ツールは主にシステムエンジニアが利用し，一般的なコンポーネントを開発して流通市場に提供する．

共通プラットフォームはミドルウェアと基本ソフトウェアの部分であり，オープンシステムや部品の流通の観点からは特にネットワーク透過性を実現する分散オブジェクト管理機能などが重要である．

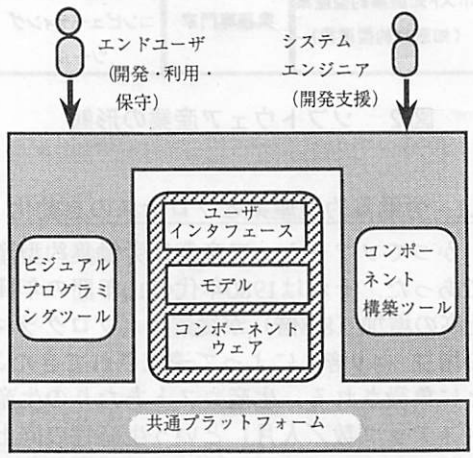

図３　エンドユーザ向け開発環境

4.3 ビジュアルプログラミングの動向

ビジュアルプログラミングの研究は以前から行われていたが，1990年代半ばから急速に発展した．その理由は次のような経緯でとらえることができる．

(1)　オープンシステム時代のソフトウェア・アーキテクチャの要請

1990年代に高まったオープンシステム化に対応するためにアプリケーションのアーキテクチャをできるだけ標準的に作る必要が生じた．そこで，まず基本ソフトウェアとアプリケーションの間にミドルウェアを挿入することによりシステムを階層構造化した．そして各階層を部品の集合で構成すると共に階層間のインタフェースを統一した．

(2)　オブジェクト指向技術による実現

このようなアーキテクチャはオブジェクト指向技術によって実現されている．オブジェクト指向技術が部品化に適している理由は，データ抽象化機能により機能部品を作りやす

いこと，クラス階層の継承機能により部品ライブラリを構築しやすいことなどがあげられる．

(3) コンポーネントウェアの出現

このようなオブジェクトの部品化はミドルウェアでの実用化が先行した．しかし，クラスライブラリで提供される部品の粒度は小さくて，プログラミングの知識が要求されるため，一般のアプリケーション開発への適用には限度があった．そこで，このギャップを埋めるために，分野を特定してアプリケーション・アーキテクチャの枠組みを与えるフレームワークや，部品の組み合わせ方を規定するデザインパターンなどが導入された．さらに，クラス部品に代わり，プログラムの定義内容を知る必要のないブラックボックス型のインスタンス部品がコンポーネント化されるようになり，インターネットを経由してコンポーネントが流通するような文化が形成されていった．

(4) ビジュアルプログラミングの実用化

しかし，このようなコンポーネントウェアが充実しても，プログラミングの本質的な難しさを克服できなければエンドユーザコンピューティングツールにはならない．そこで，個々の業務の分野対応に典型的な処理を粒度の大きなコンポーネントとして用意しておき，それらをビジュアルに組み合わせてアプリケーションを構築するためのビジュアルプログラミング支援ツールが登場した．

4.4 ネットワーク時代の課題

ネットワーク時代のアプリケーションに関する課題は種々のレベルのものがあるが，本稿の主題であるエンドユーザ主導の開発・利用・保守という視点では，インタフェースの不一致の問題がある．

分散協調型のアプリケーションの場合，たとえばクライアント側のシステムは，マルチプラットフォーム対応であることやユーザインタフェースにWWWブラウザを利用することが一般的になりつつある．そのため全体としては，インターネット文化の下に種々のレベルのインタフェースを標準化する動きが活発に行われているが，個々のケースでは必ずしもうまくはいっていない．

過渡期の現象としてはやむを得ない面があるにしても，常に”ユーザの視点”に立って解決していく必要がある．

5．研究アプローチの例

5.1 分散アプリケーション開発環境

業務の専門家が自ら作り，自ら使うようなエンドユーザコンピューティングの促進のためには，プログラミングの概念を排した新しいソフトウェアパラダイムが必要であるという認識から，我々は次のような基本的コンセプトを設定した．

A domain model ≡ a computation model
（業務モデルと計算モデルの一致）

Analysis ≡ design ≡ programming
（分析，設計，プログラミングの一体化）

この実現のために分散オブジェクト指向設計技法に基づくアプリケーション開発環境M-base [3al]を開発中である．

オフィスの日常的業務（ルーチンワーク）は，メッセージ駆動型の分散協調型モデルを用いて表現できる．本技法では，「1オブジェクト1業務」の原則に従って 一つの業務を擬人化したオブジェクト（エージェント）に割り当てることにより，メタファーベースのモデル化を行う．

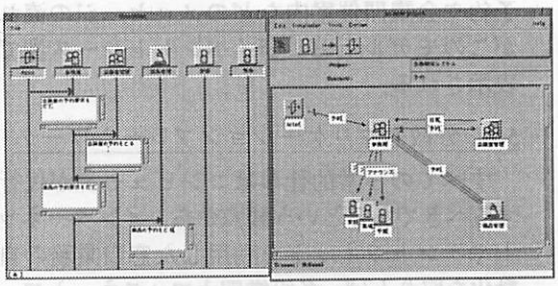

図4 Mbaseのモデリング&シミュレーション

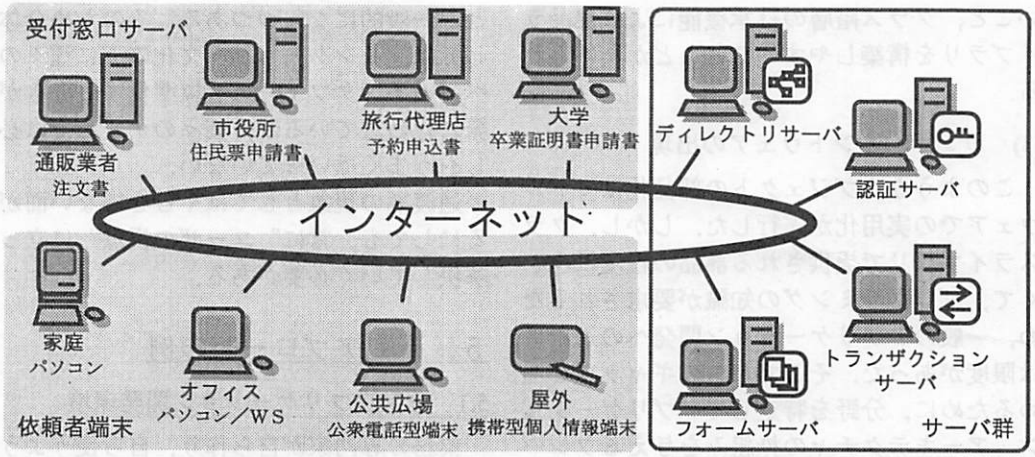

図５　多組織間オフィスネットワークMOONシステムにおける受付窓口業務

これらのオブジェクト間の業務の流れを示すメッセージフローについては，本来それを管理する立場の業務担当者が自分の代行オブジェクトの中でメッセージセット定義機能を用いて自然に記述できるようにした．従ってワークフローシステムにおいては全体のワークフローを管理するメタなシステムが不要になり，純粋な分散協調型モデルを構築できる．

さらに複雑な機能を再帰的構成法を適用した分散協調型オブジェクト群で実現するためにオブジェクトのネスト構造を導入した．

会議開催事務処理システムをモデリング＆シミュレーションツールで構築している画面例を図４に示す．この例では，分散協調型モデルの構築後，会議開催準備などのシナリオ対応にシミュレーションによる動作確認ができるようになっている．そのとき，事務局オブジェクトから他のオブジェクトへの会議室予約や会議開催案内などのメッセージの流れがこのモデル上およびイベントトレース図で表示される．

5.2　分散アプリケーションフレームワーク

すべての日常的仕事はコンピュータが代行するべきであるという観点から，インターネットやイントラネットを利用した窓口業務の自動化を取り上げ，多組織間オフィスネットワークシステム（図５）のための分散アプリケーションフレームワークwwHww (who-what-how with WWW) [4]を開発中である．

窓口業務のアプリケーションは，既にインターネット上に次々と実用化されているが，開発コストに見合った対象に限定されている．さらに個別に開発されているため，エンドユーザ（一般の利用者）から見てユーザインタフェースが統一されていないという不便さがある．

本研究では，アプリケーションフレームワークを提供することにより，エンドユーザ（業務の専門家）主導での開発・保守や多組織間でのユーザインタフェースの共通化およびエンドユーザ（一般の利用者）自身による自動記入エージェント作成・利用をめざしている．

参考文献

[1] 中所：ソフトウェア工学ーオープンシステムとパラダイムシフトー，朝倉書店 (97.5).

[2] 中所（主査）：エンドユーザ向けアプリケーション統合環境の研究開発報告書，日本情報処理開発協会 (97.2).

[3] 松本, 小西, 中所：「ドメインモデル≡計算モデル」を志向したアプリケーションソフトウエア開発環境M-baseにおける分析・設計技法，情報処理学会オブジェクト指向 '97 シンポジウム (97.7).

[4] 藤原, 斉藤, 中所：多組織間オフィスネットワークにおける分散アプリケーションフレームワークwwHwwのアーキテクチャと例題への適用、情報処理学会ソフトウェア工学研究会（97.7）.