リッチクライアントの現状と今後の動向

著者プロフィール
株式会社野村総合研究所　田中 達雄

リッチクライアントが注目される背景

2003 年頃から IT 関連の各種メディアで、リッチクライアントという言葉が頻繁に使われるようになった。その間、多くのベンダーから"リッチクライアント製品"なるものが提供され、徐々に導入事例も報告され始めている。

このように注目され始めたリッチクライアントであるが、その背景には Web アプリケーションの普及と Web ブラウザベースのクライアント環境の問題が挙げられる。詳細は後述するがインターネットの時代に入り、クライアント／サーバ型のシステムと比較した場合の開発コストや保守容易性の利点から Web ブラウザと Web アプリケーションサーバで構成される Web アプリケーションシステムへの移行が進んだ。

しかし、Web ブラウザベースのクライアントは、クランアント／サーバ型のクライアントと比較した場合、必ずしも利用者にとって使いやすいものではなかった。極端に言えば、利用者の操作性／利便性を犠牲にした上で Web アプリケーションを普及させてきたとも言える。これは利用者の生産性の低下を招き、多くのアプリケーションが Web 化される現在、無視できない問題として顕在化してきた。

これに対し、リッチクライアントは Web アプリケーションシステムであるにも関わらず、利用者の操作性／利便性を犠牲にしないクライアント（あるいは技術）であり、このことがリッチクライアントを注目させる理由となっている。

リッチクライアントの定義

登場して間もない頃は、リッチクライアントの概念すらまだ広く世間に認知されていなかった。リッチクライアントと聞いて、"従来型のクライアント／サーバ・システム"のクライアント（本連載では「ファットクライアント」と呼ぶ）をイメージする方も多くいた。

しかし、リッチクライアントという言葉が頻繁に使われ始めて約 2 年が経過した今では、ある程度その言葉の意味は理解され始め、ファットクライアントをイメージする方は少なくなっていると思う。しかし、リッチクライアントの正確な定義という点では、ベンダーによってその定義が多少異なっていたり、実現する技術も様々であったりすることから、わかりづらい状況にある。

そこで本連載では、リッチクライアントへの認識を統一するために、リッチクライアントとは表 1 の特徴を備えた「Web アプリケーションのクライアント（あるいは技術）」と定義する。以降、表 1 の定義を念頭において読み進めてほしい。

- 高い表現力を備える
- 高い操作性を容易に実現できる
- 容易に配布することができる
- クライアントリソースを有効活用できる（例えば、クライアントにプログラムやデータを保存したり、オフラインでのある程度の処理を実行することができたりするなど）

表 1：リッチクライアントの定義

過去のクライアント技術の問題点

ここでは、先に述べたリッチクライアントが注目される背景を十分に理解してもらうために、これまでのクライアント技術の歴史を振り返ることにする。これまでのクライアント技術の変遷は図 1 のようになる。図 1 に示すとおり、クライアント技術は大きく 4 つの世代に分けることができる。それぞれの世代の技術は、前の世代の技術の欠点を解消する形で登場、または進化している。その繰り返しの結果、現在のリッチクライアントに辿り着いたといえる。ではこれまでのクライアント技術には、どういった特徴があり、どういった欠点があったのかを順番に解説する。

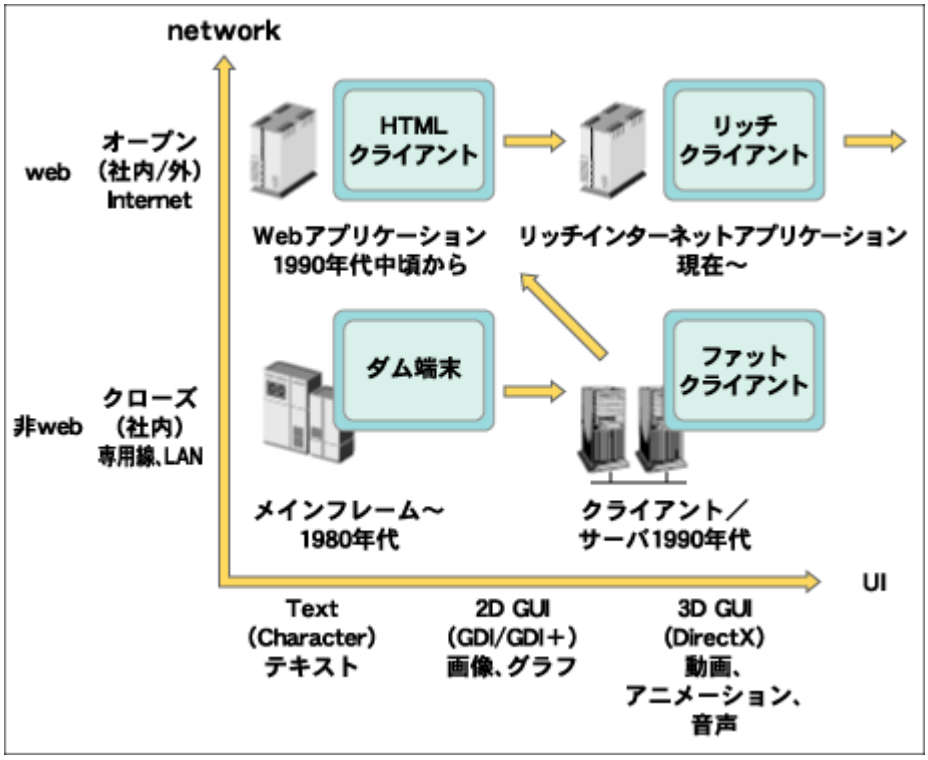

図 1：リッチクライアントの位置づけ

メインフレームの時代

この時代のクライアントマシンは CPU やメモリといったリソースが乏しく、処理はすべてメインフレーム上で行われていた。クライアント側ではデータの入出力しか行えなかった。

メインフレームのクライアントでは、基本的に文字列しか画面に表示できずにいた。しかも表示できる文字数も少なく制限があった。また、各機能はキーボードに割り当てられていたため、操作が直感的にはわからず、キーボードと機能の対応関係の教育を十分に受けていないと、そのアプリケーションを操作することはできなかった。つまり、メインフレーム時代のクライアントは、表 2 のような制約や欠点を抱えていたことになる。

- クライアントリソースが乏しく、入出力にしか使えない
- 表現力が乏しい
- 操作性も低い

表 2：メインフレーム時代の制約

クライアント／サーバ・システムの時代

1999 年代に入ると、コンピュータの性能が向上し、処理能力の高いクライアントマシンを比較的安価に購入できるようになった。そのため、この時代ではクライアントマシンのリソース（CPU やメモリ、ハードディスク、プリンタ、スキャナなど）を有効活用するようになり、いわゆる「クライアント／サーバ・システム」の開発が行われるようになる。クライアント側のリソースを有効活用したアプリケーションでは、サーバにいちいちアクセスしなくてもクライアント側で多くの処理を行える。例えば、入力チェックやデータ加工・集計処理、グラフ表示、帳票印刷などがそれに当たる。

また、クライアント／サーバ・システムでは、クライアント側の画面表示に、文字列だけでなく、ボタン、リストボックス、ドロップダウンリスト、タブ、ダイアログボックスなどにあたる 2 次元（2D）のグラフィカルユーザインターフェースが利用可能となり、クライアント側での表現力が向上した。さらに、操作用のインターフェースとして、マウスなどのキーボード以外のインターフェースが使えるようになり、複雑なユーザインターフェース（UI）を持つアプリケーションも容易に操作できるようになった。

このようにクライアント／サーバ・システム時代のクライアントは、メインフレーム時代のクライアントの制約や欠点を解消する形で登場し、表 3 のような特徴を持つ。

- クライアントリソースの有効活用（クライアントリソースが豊富になり多くの処理をクライアント側で行うことができる）
- 高い表現力（グラフィカルな UI を画面に自由に配置できるようになり、表現力が向上した）
- 高い操作性（マウスなどのキーボード以外の操作用インターフェースが使えるようになり、操作性が向上した）

表 3：クライアント／サーバ・システム時代の特徴

このクライアントを「ファットクライアント」と呼ぶ。図 2 は、次回に説明する「リッチクライアント評価」調査の結果から導き出した「ファットクライアントの優位性トップ 10」であるが、上記で挙げた特徴がやはり上位に挙がっている。

クラサバ（ファットクライアント）

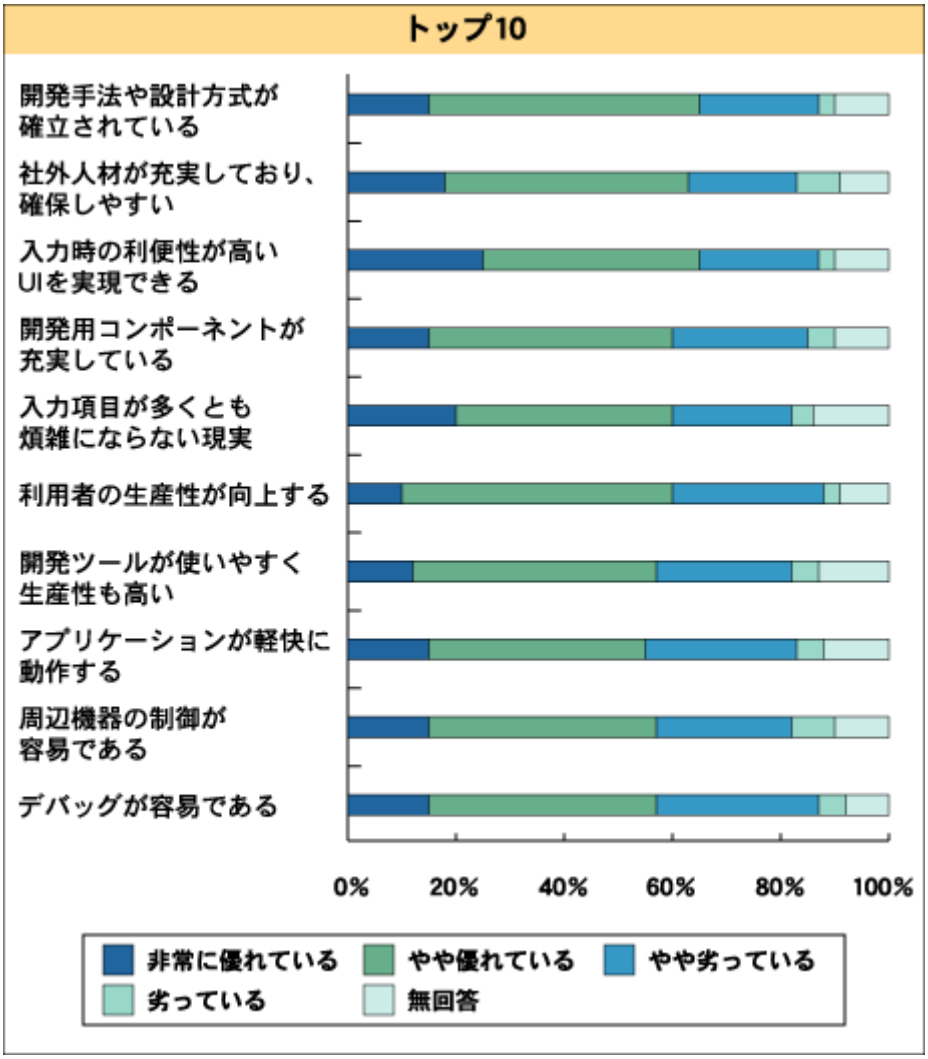

図 2：ファットクライアント優位点（トップ 10）

ファットクライアントの問題点
（クライアントプログラム配布の非効率）

しかし、ファットクライアントでは、クライアントプログラムの配布に関して無視できない問題があった。基本的にファットクライアントを利用する場合、すべての（必要な）クライアントマシンに対して、クライアントプログラムの配布やマシン環境の設定を導入時およびバージョンアップ時に逐次行う必要があり、運用に多大なコストがかかるという問題をはらんでいた。

例えば、Excel で開発したクライアントプログラムを利用する場合には表 4 のような問題が発生した。

1. すべてのクライアントマシンに Excel をインストールする必要がある
2. Excel で開発したクライアントアプリケーションプログラムを CD-ROM などで配布する（この時代、ネットワークを通してクライアントアプリケーションプログラムを配布できるほど、ネットワーク回線は太くなく、インフラも整備されていなかったため、通常 CD-ROM やフロッピーディスクで配布されていた）
3. 配布された Excel アプリケーションプログラムを動作可能とするためのデータベース接続設定（アダプタのインストールも必要）やネットワークの設定（ネットワーククライアントソフトウェアのインストールも必要）などのクライアントマシン側の設定をクライアントマシンごとに設定する必要がある
4. Excel で開発したアプリケーションプログラムのバージョンアップや修正パッチなどを行う場合も CD-ROM やフロッピーディスクなどでプログラムや修正パッチを配布する
5. Excel 自体がバージョンアップした場合、社内に複数の Excel バージョンが存在するようになり、その複数存在する Excel にそれぞれに対応した Excel アプリケーションを用意し、管理していかなくてはならなくなる（Excel 自体をすべてのクライアントマシンでバージョンアップすればよいとの判断もあるが、複数の Excel アプリケーションを利用している場合、すべての Excel アプリケーションをバージョンアップした Excel に合わせて開発し直す必要が生ずる）

表 4：Excel を使用したファットクライアントの問題点の例

このように配布や保守に関する手間が非常に多く、クライアント数が増加すればするほど、問題が大きくなる。図 3 は次回に説明する「リッチクライアント評価」調査の結果から導き出したファットクライアントの優位性ワースト 10 である。これを見ると、配布やクライアント環境に対する項目が欠点の上位を占めていることがわかる。

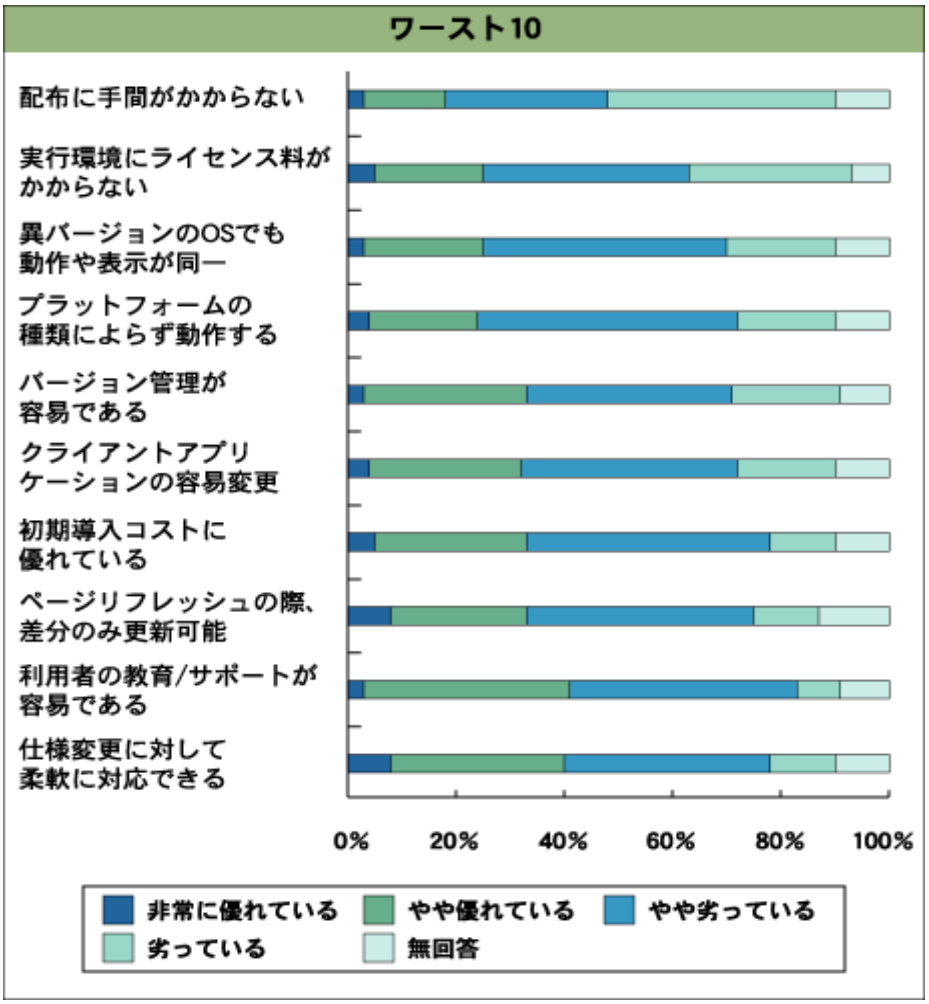

図 3：ファットクライアントの優位点（ワースト 10）

Web アプリケーションの時代

次にファットクライアントの欠点を解消するものとして普及し始めたのが、HTML と Web ブラウザを用いたクライアントアプリケーションである。このクライアントアプリケーションを HTML クライアントと呼ぶ。

HTML クライアントの最大の特徴は、「配布の容易さ」にある。Web ブラウザがインストールされたクライアントマシンがあれば、インターネット／イントラネットを通じて、どこからでも Web サーバから HTML ファイルをダウンロードし、クライアント画面を描画することができる。クライアント／サーバ・システム時代のときのように、アプリケーションプログラムを CD-ROM などで配布したりクライアントマシンの環境を再設定したりする必要はない。

図 4 に「リッチクライアント評価」調査の結果から導き出した HTML クライアントの優位性トップ 10 を示す。これを見ると、以下のようにファットクライアントではワースト 10 に含まれていた配布やクライアント環境に対する項目が優位性の上位を占めていることがわかる。

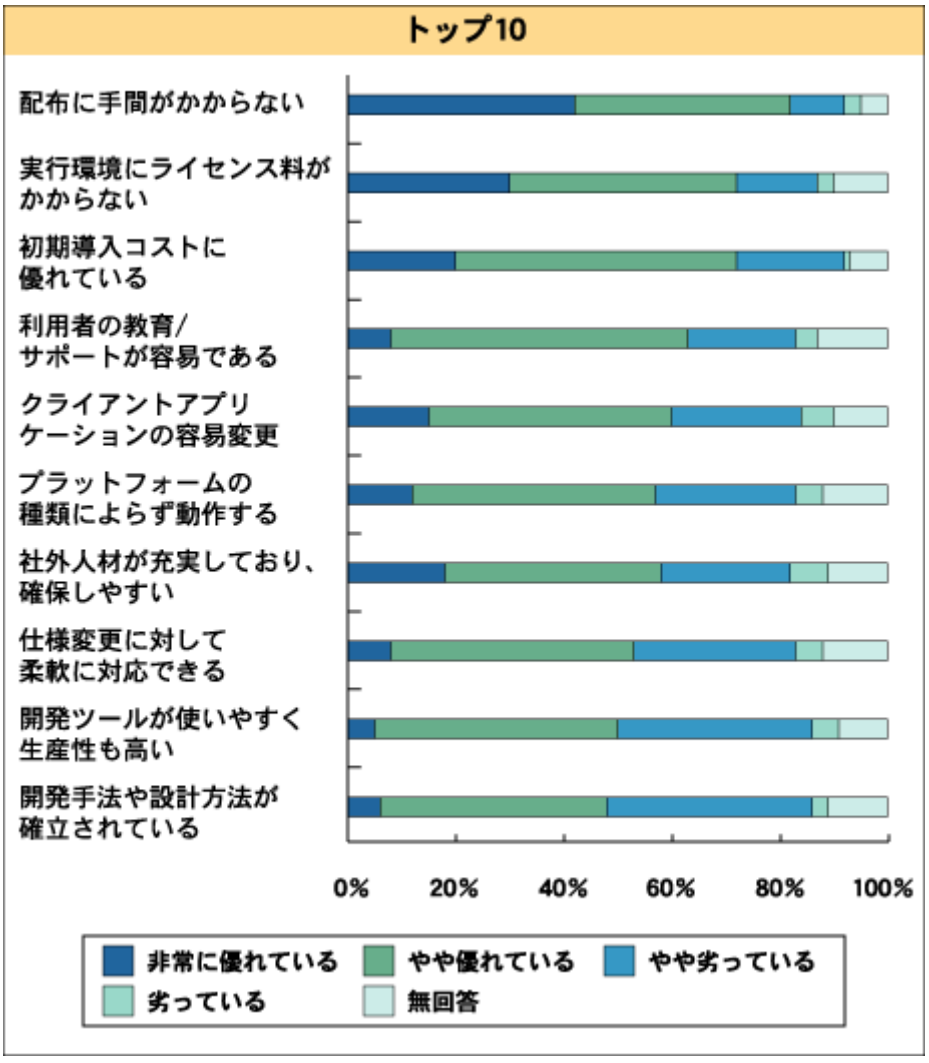

図 4：HTML クライアントの優位点（トップ 10）

HTML クライアントの問題点（表現力と操作性の乏しさ）

HTML クライアントは、クライアント／サーバ・システムの問題点であったプログラムの配布やクライアント環境設定に関する問題を解決したが、Web ブラウザをクライアントとして利用するが故のデメリットもあった。

Web ブラウザが描画する HTML ファイルは、もともとテキストデータをマークアップするために考案された言語であり、クライアントアプリケーションとして利用するには、ユーザビリティの点で問題があった。例えば、タブキーによる入力フィールドの自動移動や、入力項目のオートフォーマット（「30000」と入力したものを「¥30,000」に自動変換するような機能）などは、HTML だけでは不可能である。

その後、HTML を補完する技術として CSS（Cascading Style Sheet）や JavaScript などの言語が登場し、ユーザビリティの問題も多少改善されるようになるが、今度は 1 つの画面の開発に複数の言語を組み合わせることが多くなり、開発作業が煩雑化するといった問題が生じた。一方、これも HTML を補完する目的で、JSP（Java Server Pages）、ASP（Active Server Pages）、PHP、Perl などのサーバサイドの技術が使われ始め、アプリケーションの開発やメンテナンスはますます煩雑化することになった。加えて、HTML 以外の言語（CSS や JavaScript など）を HTML ファイル内に組み込むと Web ブラウザの種類やバージョンによって動作が異なったり、あるいは動作しなかったりするといった問題も生じた。そのため、アプリケーションを更新する度に複数の Web ブラウザやバージョンで動作確認テストを行わなければならず、テスト工数を増大させる原因になった。

図 5 は「リッチクライアント評価」調査の結果から導き出した HTML クライアントの優位性ワースト 10 である。これを見ると、以下のようにファットクライアントでは優位性トップ 10 に含まれていた UI やクライアントリソースの有効利用に関する項目が上位を占めていることがわかる。

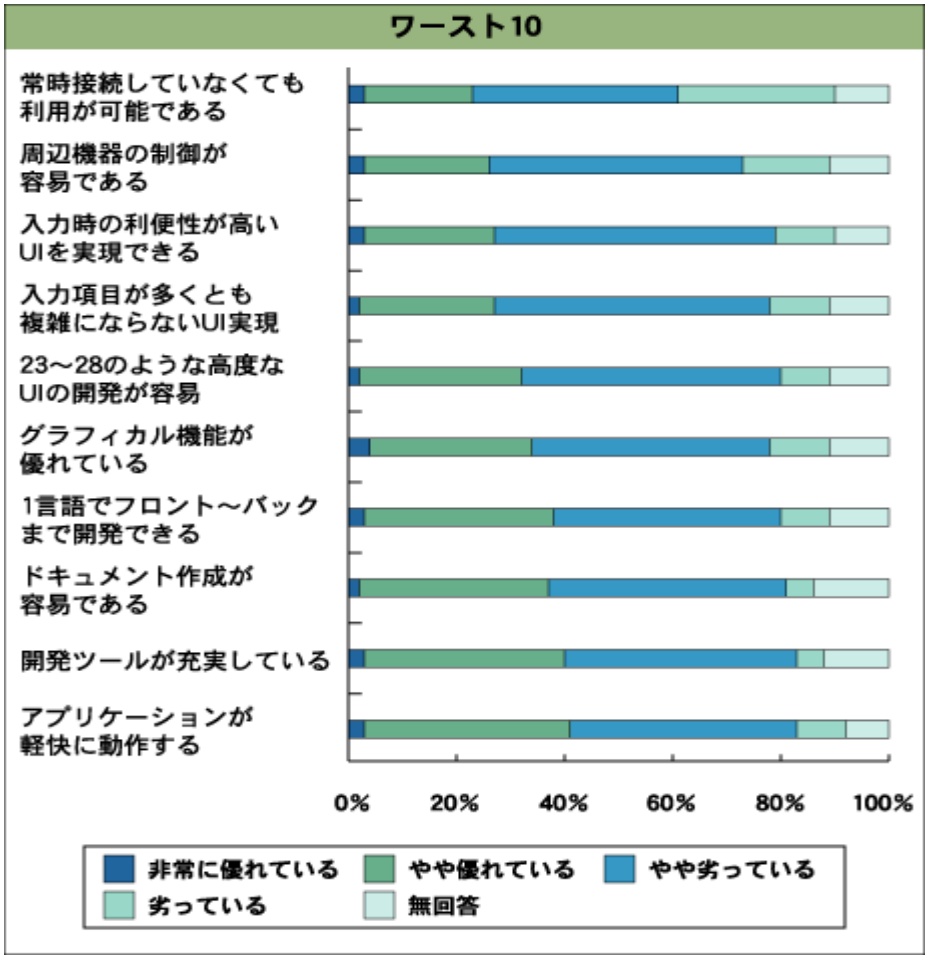

図 5：HTML クライアントの優位点（ワースト 10）

リッチクライアントは何を解決するのか

リッチクライアントは、前述したこれまでのクライアント技術がはらんでいた問題を踏まえ、ファットクライアントと HTML クライアントの双方のメリットを備える技術として登場した技術である。具体的には双方から以下のようなメリットを継承している。

ファットクライアントから継承したメリット

クライアントリソースの有効利用

リッチクライアントでは、ファットクライアントと同様にいちいちサーバにアクセスしなくともクライアント側だけで入力チェックやリスト表示、画面遷移など多くの処理を行うことができる。サーバへのアクセスは必要なデータの取得や送信時のみに行えばよい。

このように必要なときだけサーバにアクセスする処理形態をとれるリッチクライアントでは、オフライン環境でも、ある程度の処理を行うクライアントアプリケーションを開発することが可能となる。

高い表現力と操作性の作りこみやすさ

リッチクライアントでは、ファットクライアントと同様に利用者とのインターフェースである画面上に、多くのグラフィカルユーザインターフェースを自由に配置することができる。HTML クライアントでも HTML 以外の言語を駆使した場合、ファットクライアントに近いユーザインターフェースを作りこむことはできるが、作りこみやすいとは言い難い。

さらに最近のリッチクライアントでは、動画やアニメーション、音声や 3 次元（3D）グラフィカルユーザインターフェースなどを提供する技術や製品が提供され始めており、その意味ではファットクライアント以上の高い表現力や操作性をより簡単に作りこむことが可能になりつつある。これは Web アプリケーション利用者の幅（子供から高齢者、企業内であれば経営者、営業、技術者、コールセンターなど）が広がる中で、様々な利用者に合った使いやすいインターフェースを作りこむ自由度を高めることにつながる。

HTML クライアントから継承したメリット

プログラムの配布が容易

リッチクライアントでは HTML クライアントと同様に、Web サーバから動的にクライアントアプリケーションをダウンロードし起動することができる。そのため、ファットクライアントのように CD-ROM での配布やインストール、環境設定などの作業を行わなくてもよいといったメリットがある。なお、1 度ダウンロードしたクライアントアプリケーションは、クライアントマシンのローカルディスクに保存することで、2 度目以降からはローカルファイルとして保存されているクライアントアプリケーションを起動するといった方法をとることができる。これにより、Web サーバやネットワークにかかる負荷を減らし、またアプリケーションが起動するまでの時間を短縮することが可能である。

リッチクライアント独自の便利機能

リッチクライアントでは、ファットクライアントや HTML クライアントのメリットを継承し、問題点を解決するほかに、以下のような独自の便利な機能も実現されている。

Web ブラウザ特有の問題を解決

リッチクライアントの多くは、Web ブラウザに非依存な実行環境を用意している。そのため、開発者は、Web ブラウザの「戻る」ボタンや画面のリフレッシュなどに起因する"Web ブラウザ特有の問題"に悩まされることがない。

通信機能の充実

リッチクライアントの実行環境には、Web サービスなどの通信プロトコルのサポート、メッセージ到達保障、認証処理、ロギング、暗号化／復号化といった、クライアントアプリケーションがサーバと通信する際に必要となる"お決まりの機能"があらかじめ用意されている技術や製品がほとんどである。そのため、リッチクライアントの開発者は、業務ロジックの作り込みに集中でき、なおかつ配布するクライアントアプリケーションのサイズを小さくすることが可能となる。

まとめ

今回は、リッチクライアントが登場した背景やその特徴を解説した。次回は、野村総合研究所にて実施したリッチクライアント市場調査結果を紹介する。

リッチクライアントに関するのアンケート調査内容

第 1 回で報告したような特徴を備えたリッチクライアントに対する評価や市場ニーズを確かめるために、2004 年 3 月 15 日から 4 月 2 日にかけて、「リッチクライアントに関する市場調査」というアンケート調査を実施した。

アンケートは、無作為に抽出した国内企業約 2,000 社に郵送で送付し、275 社から回答を得た。業種や企業規模（従業員数）の分布は図 1 のとおりである。参考として回答者のリッチクライアントに対する理解度を示しているが、リッチクライアントを「内容まで深く理解している」、「ある程度の内容は理解している」の合計が約 29.5%である。

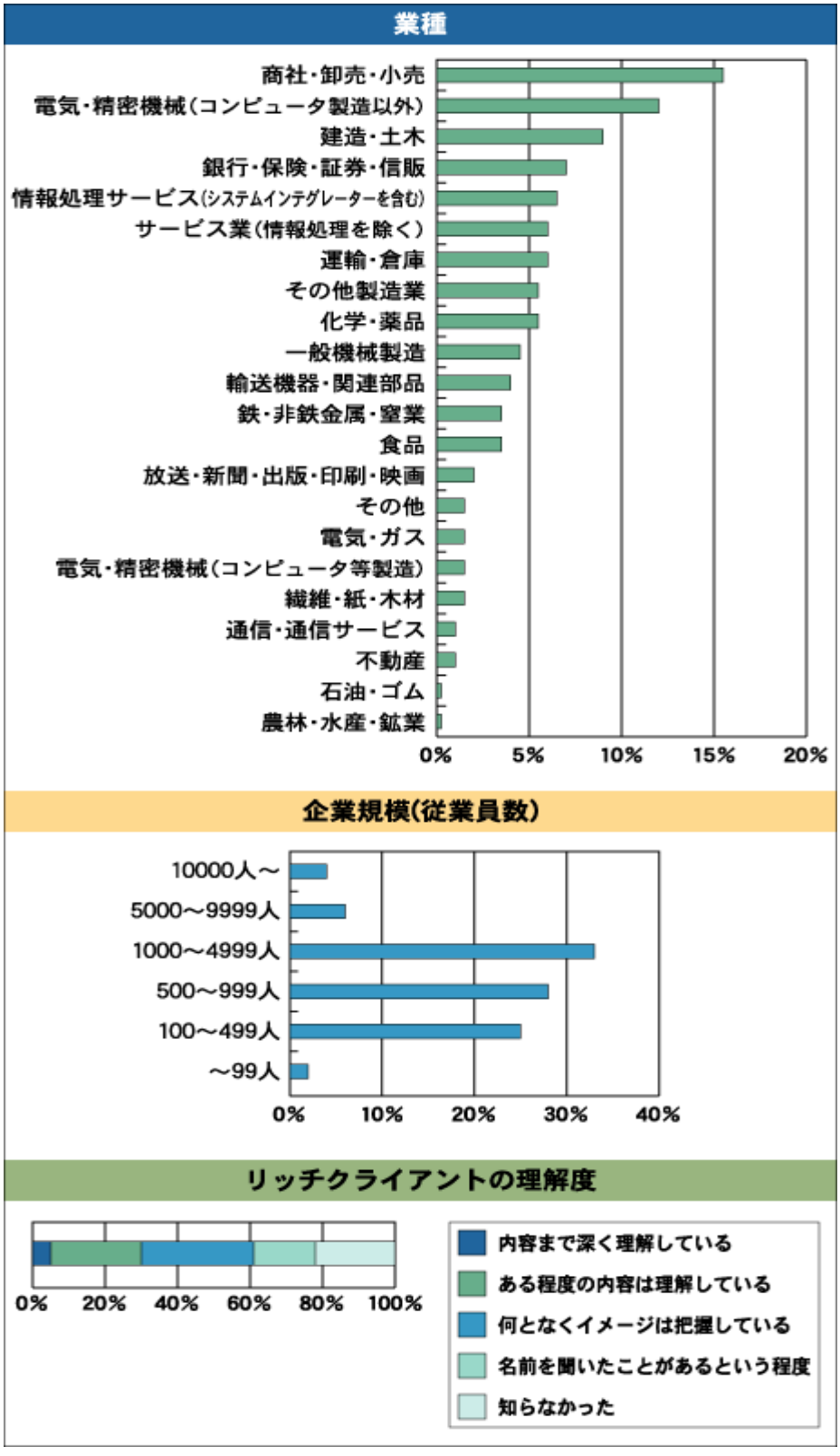

図 1：アンケート結果の概要

以降、この調査結果の内容をもとに、今後のリッチクライアントの方向性について考察する。

何をリッチクライアントにしたか？

まず、現在稼動中のシステムで利用しているクライアントアプリケーションの形態について聞いたところ、全体の約 14.4%がリッチクライアントを、約 30.5%が HTML クライアントを、そして約 39.4%がファットクライアントを利用しているという結果が得られた（図 2）。このことから、依然としてファットクライアントの比率が高いことがわかる。

また、現在リッチクライアントを利用しているシステムで、以前にどのような形態のクライアントアプリケーションを使っていたのかについては、HTML クライアントからリッチクライアントに移行したケースが 12.9%、ファットクライアントからリッチクライアントに移行したケースが 24.0%という結果が得られ、ファットクライアントからの移行が、HTML クライアントからの移行の約 2 倍に達していることがわかる（図 3）。

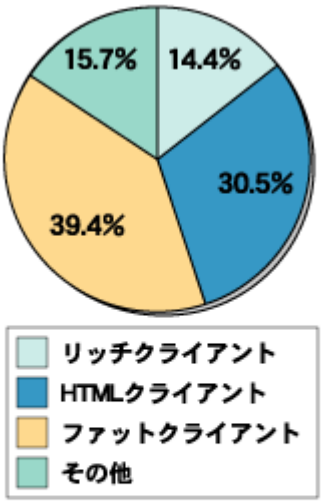

図2:現在利用しているクライアント技術

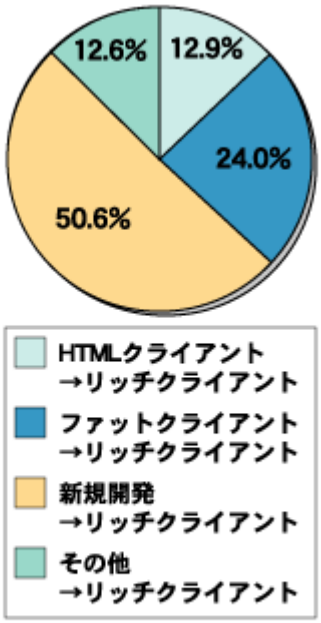

図3:リッチクライアントへ移行したシステムの種類

さらに、リッチクライアントに移行したシステムの種類については、第 1 位が「非 Web 系の社内基幹系業務システム」で約 17.1%、第 2 位が「非 Web 系の社内情報系システム」で約 13.5%であった（図 4）。

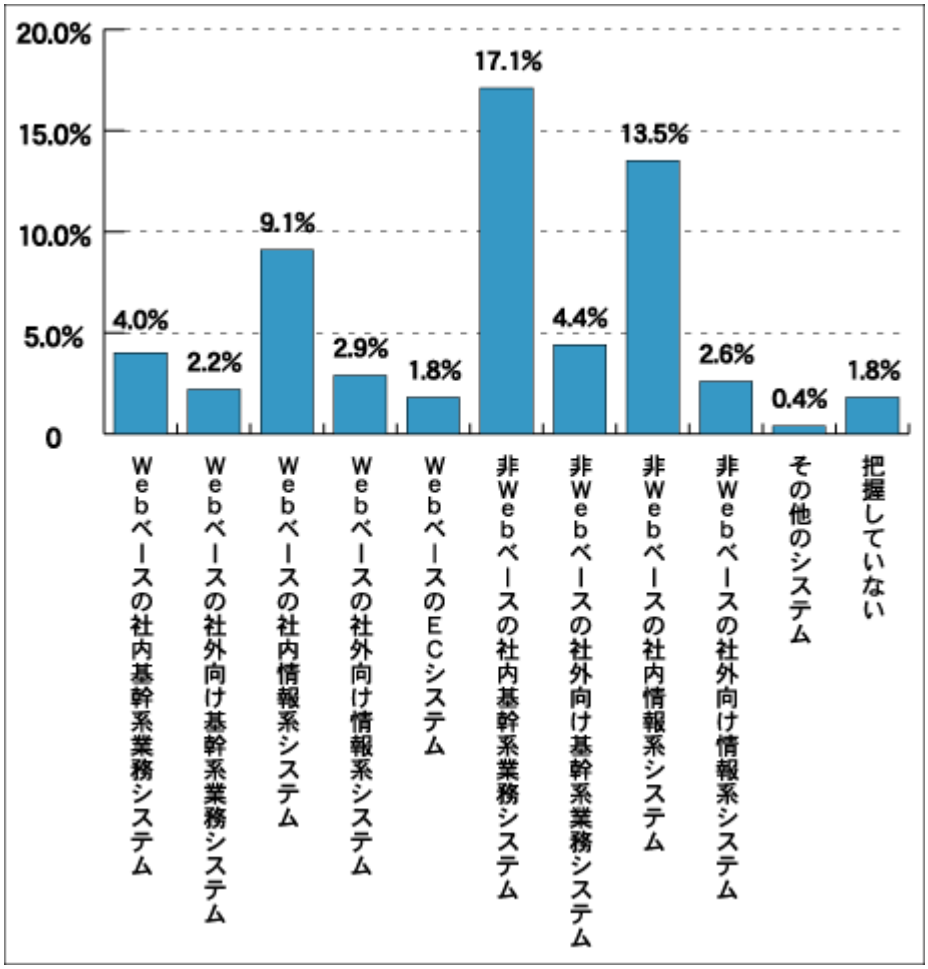

図 4：リッチクライアントへ移行したシステムの詳細

HTML クライアントのユーザビリティの問題を解決するものとして、注目されることが多いリッチクライアントであるが、実際には HTML クライアントよりもファットクライアントから移行する場合のほうが多い。さらに、その内訳を見ると、社内基幹系業務システムの割合が最も高い。

これらの結果から、Web アプリケーションの利便性については認識していたが、ユーザビリティを犠牲にしてまで HTML クライアントに移行できなかった社内基幹系業務システムを、リッチクライアントを使って Web 化しているという傾向がうかがえる。

これから何をリッチクライアントにしたいのか？

前述まで、過去から現在におけるリッチクライアント移行実績を解説したが、ここからは現在から将来にかけてのリッチクライアントの移行ニーズについて解説する。

HTML クライアントの移行ニーズ

現在使用している HTML クライアントの移行ニーズを調査したところ、将来リッチクライアントへ移行したいと回答したのは全体の約 18.8%であった。また、リッチクライアント理解者に至っては、21.7%であった（図 5）。

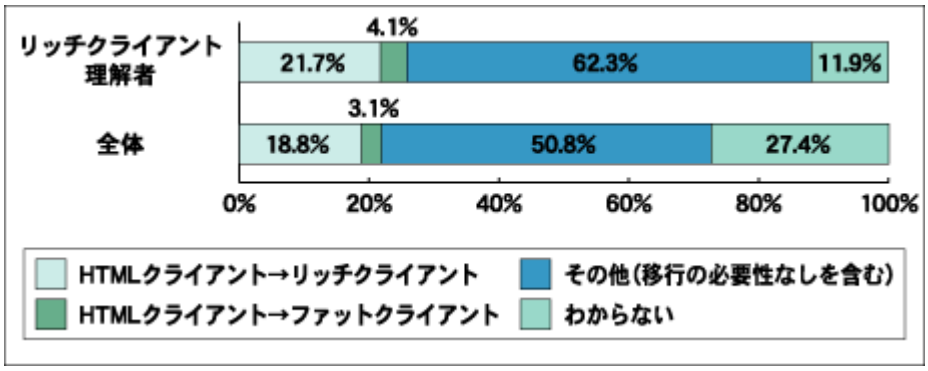

図 5：HTML クライアントの移行ニーズ

リッチクライアントを理解している方が移行を希望する率が高く、リッチクライアントの特徴を理解しているほど、リッチクライアントへの移行ニーズが高くなると見ることができる。

逆に、筆者らの予想に反して「現在の HTML クライアントを将来もそのまま移行しなくてもよい」と考えている回答者が 50%強と多く、「HTML クライアントのユーザビリティでも十分」、もしくは「我慢して使う」という企業が多くを占めた。そもそも高度なユーザビリティを必要としないシステムのみを HTML クライアント化していたとの見方もできる。

では、HTML クライアントからリッチクライアントへ移行したいと考えるシステムの内訳はどうなっているのだろうか。それについ調べたところ、第 1 位が「Web システムの社内情報系システム」で約 20.7%、第 2 位が「Web システムの社内基幹系業務システム」で約 15.6%であった（図 6）。これは、リッチクライアントへ移行したシステムの詳細（図 4）の傾向と同じである。

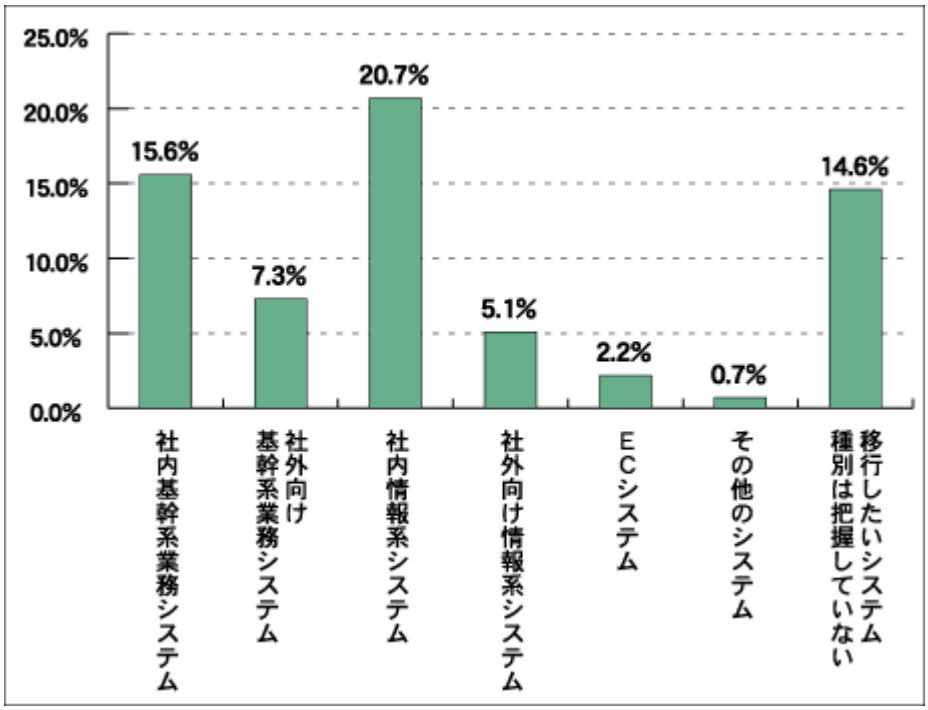

図 6：HTML クライアントからリッチクライアントへ移行したいと回答したシステムの詳細

通常、利用者の生産性を厳しく求められるはずの社内基幹系業務システムが第 2 位である理由は、そもそも HTML クライアントが利用されているシステムとして母数が多いためではないかと考えられる。

ファットクライアントの移行ニーズ

一方、現在使用しているファットクライアントの移行ニーズについて調査したところ、将来リッチクライアントへ移行したいと回答したのは、全体の約 24%であった。また、リッチクライアントのメリットを理解している方の中で、ファットクライアントからリッチクライアントへ移行したいと考えている方は 34.7%に達し、HTML クライアントからの移行ニーズの約 1.5 倍となった（図 7）。

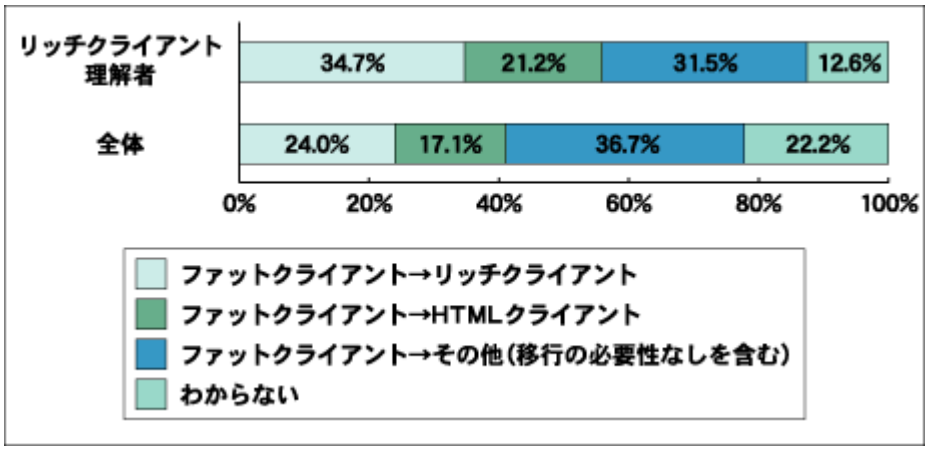

図 7：ファットクライアントの移行ニーズ

ファットクライアントをそのまま使うという割合は約 30%強と HTML クライアントの結果（約 50%強）より大幅に下回り、現在のファットクライアントをリッチクライアントもしくは HTML クライアントの何らかに移行したいと考えている回答者が多いとの結果が得られた。

リッチクライアント製品を提供しているベンダーもファットクライアントをターゲットに拡販活動を行うべきであろう。

次にファットクライアントからリッチクライアントへ移行したいシステムの種類であるが、第 1 位が「非 Web システムの社内基幹系業務システム」で約 26.6%、第 2 位が「非 Web システムの社内情報系システム」で約 23.3%であった。これもリッチクライアントへ移行したシステムの詳細（図 4）の傾向と同じ結果となっている（図 8）。

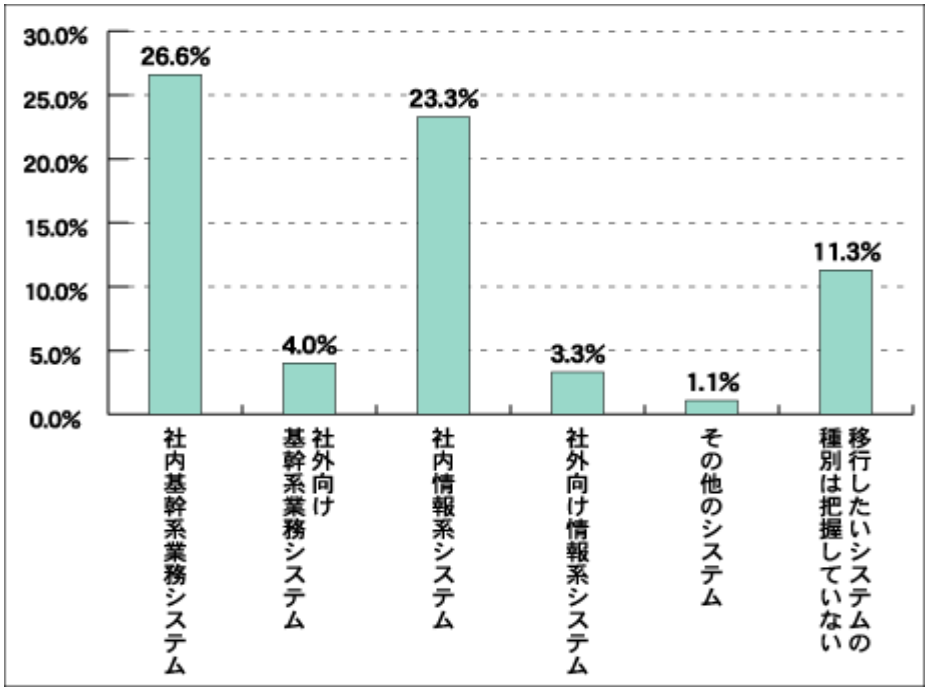

図 8：ファットクライアントからリッチクライアントへ移行したいと回答したシステムの詳細

これらの結果からも、ファットクライアントを使っているアプリケーションシステムは現在の Web 化という大きな流れがあるにも関わらず、ユーザビリティを重視するがゆえに HTML クライアントへ移行できなかったアプリケーションシステムが多く、リッチクライアントを機に Web 化を進めたいとの潜在的ニーズが多くあることがうかがえる。

将来のリッチクライアント比率

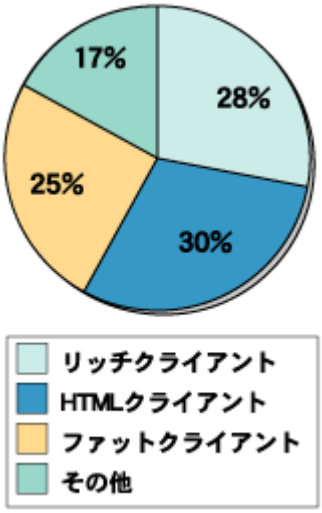

図 9：将来のクライアント技術

以上の結果から、リッチクライアントへの移行ニーズは、HTML クライアントよりファットクライアントの方が高く、若干ではあるが社内基幹系業務システムの移行ニーズが社内情報系システムと比べて高いことがわかる。一般に、基幹系業務システムは情報系システムと比べ、短時間で多くの処理をいかに効率よく行うかを問われるシステムが多い。この結果は、その傾向を反映しているとも言える。

また、これまでの結果を総合して、将来（2～3 年後）における各クライアント技術の割合を推定したものが図 9 である。リッチクライアントの割合は、現在の 14.4%から約 28%へと増加し、HTML クライアントは約 30%と変わらず（これは、ファットクライアントからの移行分とリッチクライアントへの移行分が相殺されるため）、ファットクライアントは約 14%減少して約 25%になると予想される。

今回の調査結果から、リッチクライアントへの移行ニーズは高く、リッチクライアントは今後ますます普及していくものと考えられる。

リッチクライアントのメリット／デメリット

このように、今後ますます増加すると考えられるリッチクライアントであるが、実際に導入する際には、他のクライアント技術に対するメリットとデメリットをよく吟味しなければならない。特に、新しい技術が登場する際には、そのメリットばかりが強調される傾向があり、デメリットについてはあまり表に出てこないので注意が必要である。

では、リッチクライアントのメリット／デメリットについて、ユーザおよび開発者はどのように見ているのだろうか。以下では、その調査結果を紹介する。

リッチクライアントの特徴

図 10 は、アプリケーション開発において重要と思われる 33 項目に対して、回答者に「重要と思うか」という質問と「リッチクライアントに優位性があるか」という質問を同時に質問した結果であり、横軸に「アプリケーション開発における重要性」、縦軸に「リッチクライアントの優位性」を配置した図である。

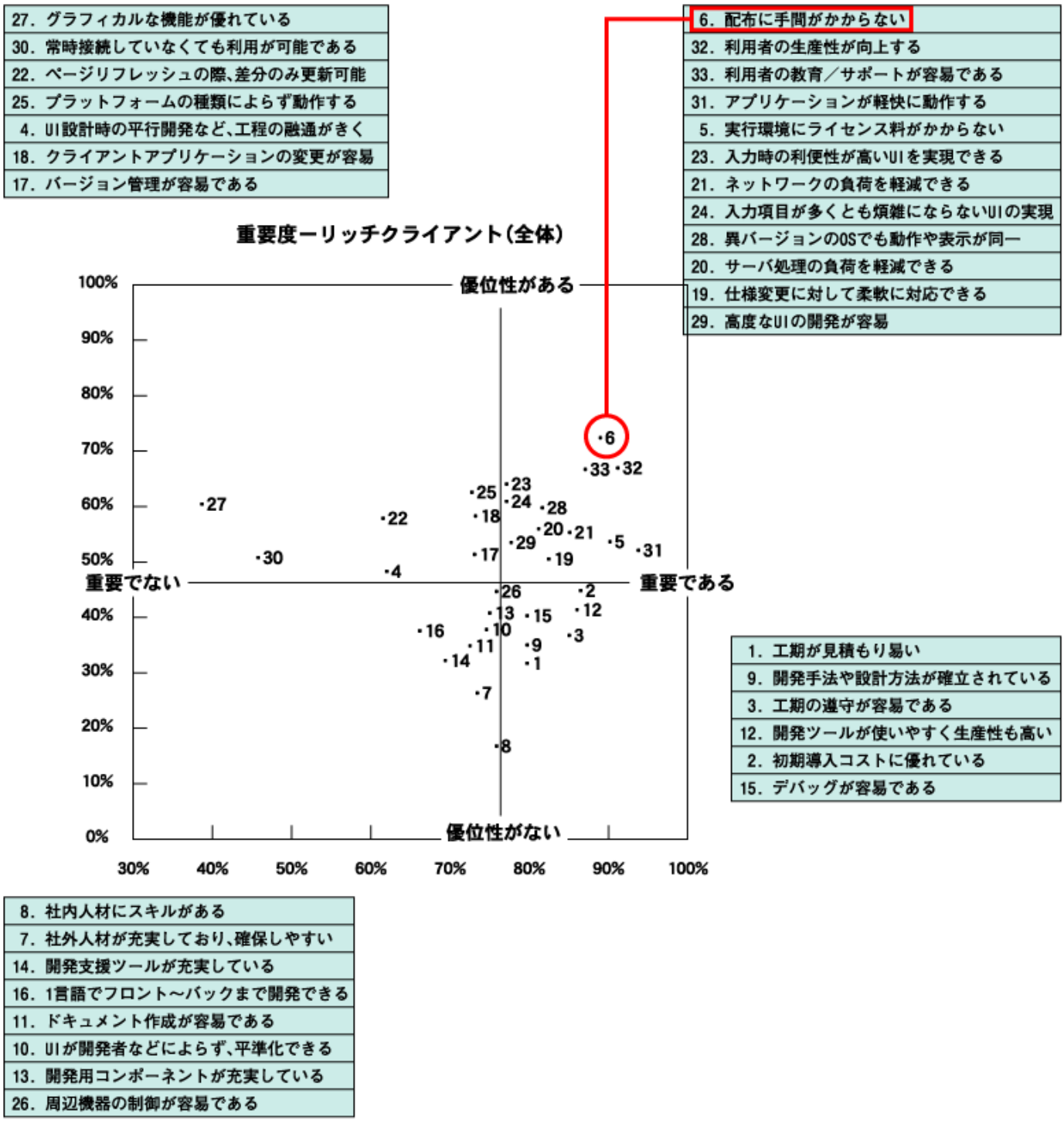

図 10：リッチクライアントの優位性

例えば、「6. 配布に手間がかからない」という項目の場合、"アプリケーション開発において重要視されており、かつリッチクライアントに優位性がある"ということになる。

図 10 では、アプリケーション開発においてあまり重要視されていない項目については、リッチクライアントに優位性があってもあまり意味がないとの見方をし、アプリケーション開発において重要視されている項目だけを重点的に評価することにする。

この調査の結果、リッチクライアントは、「配布に手間がかからない」、「利用者の生産性が向上する」、「入力時の利便性が高い UI を実現できる」など、まさにリッチクライアントの特徴とも言える項目で優位性があると評価された。

また、この結果から、クライアントアプリケーション開発時に重要視されやすいリッチクライアントが劣っていると考えられている項目として、「工期が見積もりづらい」、「開発手法や設計方式が確立されていない」、「開発ツールが整備されておらず、（開発の）生産性が低い」といったものがあることがわかった。

2004 年 5 月発表時点のリッチクライアント調査結果で、リッチクライアントは確かに高度な UI を実現でき、利用者の生産性を向上させ、配布の手間もかからないといったメリットを備えるが、その反面、現状では開発手法や設計方式が未整備であり、また、開発ツールやデバックなどの開発環境も未成熟であると評価されている。「現時点ではどうなっているのか？」という点については、現在主要ベンダーへのヒアリングを予定しており、第 4 回で後述したいと思う。

ファットクライアントからリッチクライアントへ移行した事例

会社 A の販売管理システム

	移行前	移行後
クライアント	① Visual Basic で開発した Windows ネイティブアプリケーション ② Magic8 実行	.NET Framework 上で動作する Windows Form
サーバ	Windows Server2003	IIS（Windows 2008 Server）
開発環境	① Visual Studio ② Magic8 開発	① Visual Studio .NET ② Magic uniPaaS V1
C/S 接続	Windows Network Pervasive PSQL	XML ベース Web サービス（SOAP）