情 報 シ ス テ ム 52－2
（１９９５．１．１７）

クライアントサーバー環境における アプリケーション開発と、 開発ツールの役割

池田賢一

日本グプタ　製品開発本部　技術支援担当

クライアントサーバー環境における、開発ツールの選択方法と、そのカテゴリー分類について述べる。
まず、現在のシステムニーズを概略的に図示し、その中での必要機能概略を述べることで、開発ツールに求められているニーズの明確化を図る。
さらに、定義された機能を利用してカテゴリー定義を行う場合の視点についても考察し、いくつかの開発ツールの定義を行うこととする。
現在、数１０種類存在しているＰＣ上の開発ツールについて、その評価項目の設定を行うとともに、提供される機能をいかにしてシステム開発に適合させていくかについての前提知識の明確化が目的である。

1　はじめに

現在のWindowsを中心としたPC環境には、数１０種類に及ぶ、開発ツールが存在しており、そのツールのいずれを選択するかは、非常に混沌とした状況になっていると言って良い。

コンピューター関係の雑誌では、○×表を元にした評価が中心となっているが、システムからのニーズを直接反映しているとはいいがたく、実際は、機能の有無が判別できても、それらがシステムのニーズに対してどのような影響を与えるかの判断が難しいケースが一般的である。

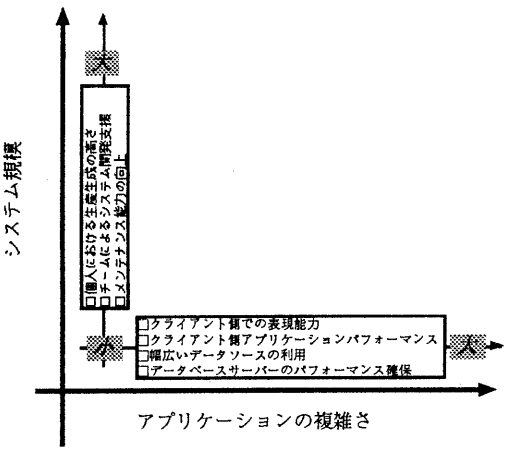
本文では、現在のシステムニーズを分類し、そこでのニーズを明確化するとともに、そこから導き出される必要機能を考えてみることにする。

最後に、それらの必要機能に基づき現在、アプリケーション開発ツールの市場を構成している、主要なツール群を、製品の機能や、性格にあわせて評価ポイントを定義する。

2　システムニーズの分類

アプリケーション開発環境を考える場合の１つの切り口をつくるために、以下のような図を設定した。

『システムニーズによるエリアと実現すべき機能サポート』



2.1　縦軸の意味

Y軸には、アプリケーションの規模をとる。アプリケーションの規模とは、実際に開発しなくてはならないシステムにおける、画面数や、帳票数等から考えた規模とする。

一般的には、規模が大きくなるにつれて、開発に参加する人員が増加するとともに、可能な限り生産性を向上させる必要もあると考えれば良い。

2.1.1　縦軸の構成要素

縦軸を構成している要素には、以下のようなものがある。

□個人の生産性向上
□チームに対する開発支援
□メンテナンス能力の向上

2.2　横軸の意味

横軸には、そのアプリケーションで要求される複雑性をとる。

アプリケーションの複雑さとは、クライアント側、サーバー側の双方に要求されることになる。さらに、昨今の事情では、複数のデータソースへのアクセス、クライアント側での機能拡張等についても考慮する必要がある。

パフォーマンス、キャパシティ面で、大規模システムをサポートすることも非常に重要な要件であると考えられる。

2.2.1　横軸の構成要素

横軸を構成している要素には、以下のようなものがある。

□クライアント側での表現能力
□クライアント側アプリケーションパフォーマンス
□幅広いデータソースの利用
□データベースサーバーのパフォーマンス確保

3　システムの規模と、その中でのニーズ

前述したアプリケーションの定義領域図を元に、システムの規模と機能面から考えあわせた定義を考慮する。

3.1　【個人向けシステム】

【個人向けシステム】は、全社規模のアプリケーションではなく、部門や、個人での使用が前提となるアプリケーションである。

このエリアで重要なのは、使い勝手であり、生産性や、機能性にはあまり依存しない。

逆に言えば、なるべく簡単なツールで済ましてしまえば良い領域であり、値段や、簡単さなどが重要な要件になると考えれば良い。

3.2　中規模企業システム

ある程度の規模と、ある程度の複雑さを持つシステムを指す。
ここで問題になるのは、「ある程度」というどの程度のものを指すかである。

ここでは、その指標を考える。考えられる指標は以下のものであろう。

□データベースの規模
□クライアント側に転送が必要なデータ量
□画面上で必要になるオブジェクトの数
□実際のアプリケーションを稼働させた場合の各種バッファーの大きさ

いずれにしても、現在のクライアント・サーバー環境におけるクライアント側のインフラストラクチャーがWindows上に構築されている以上、それに依存しているものも大きい。

ここでは、それらの問題も含めて現在の境界線を以下のように設定してみることにする。

□データベースの規模
クライアント側から発行するＳＱＬ文のチューニングの必要性が発生する、10,000件程度が目安か

□クライアント側に転送が必要なデータ量バッファーサイズに関連していると考えて良い。データ長×レコード数の合計が、６４ＫＢ程度が限界か？

□各種のバッファーサイズ
６４ＫＢが目安となる。

やはり問題になるのは、【個人向けシステム】の構築基盤のディフォルトの機能では、対応出来なくなってしまうような機能が出始めてからが、この規模のシステムになると考えて良い。

3.3　特殊用途システム

通常Ｃ言語を中心に非常に複雑なシステムを構築する場合と考えて良い。
１つ１つのアプリケーションは複雑だが、用途が一般的ではないため、高度な生産性を要求されることはない。

一般的なシステムではないため、ここでは解説することを避ける。

3.4　大規模全社システム

非常に複雑でかつ大規模なシステム開発を行う領域である。
アプリケーションへの機能ニーズも高くまた、作らなくてはならないアプリケーションも多い。
すなわち、高い機能と生産性の確保が必要とされるシステムなのである。

このタイプのシステムは、現在のところでは、クライアント・サーバー環境が未だに挑戦している領域であり、これからの分野であると考えてもよいであろう。

以下では、大規模システムへの対応を若干述べることにする。

3.4.1 大規模システムへの対応を推進するには？

今、考えなくてはならない問題は、どのようにすれば、【大規模全社システム】を、クライアント／サーバー環境で構築することができるかである。

いずれにしても、【個人向けシステム】や、【特殊用途システム】を開発するためのインフラストラクチャーでは、最終的なニーズには、対応することが難しいと考えて良い。

決定的に機能が不足しているのであれば、求めるアプリケーションを作り上げることは不可能であるし、どれだけ複雑なアプリケーションが作り出せても生産性が伴わないのであれば、実際に開発することは困難だからである。

３年後にシステムができあがるような形態では、今の企業の情報システムへのニーズを満足することはできない。

高機能でかつ高生産性を実現するようなシステムの構築には、【中規模企業システム】の構築に大きな役割を果たしたツールや、ソフトウエアが重要な役割を果たすと考えて良い。

以降の章では、それらを念頭に置き、現在の開発ツール全体の状況を記述することにする。

4　現在のアプリケーション開発ツールの状況

ここでは、アプリケーション開発ツールを評価するための、評価項目を設定し、それらの定義を行うことにする。
それらの評価項目を以下に示す。

4.1　高機能アプリケーションの構築

□フロントエンドの構築能力
　●基本オブジェクトの利用可能範囲
　●カスタマイズ可能度
　●他アプリケーション利用度
□フロントエンドの限界能力の向上
　●アプリケーションコード減量化機構
　●インタプリタ高速化
□データベース利用機構
　●発行SQL文カスタマイズ機構
　●検索結果表示機構
　●データベースサーバー機能拡張利用
　●データベースサーバー拡張言語利用
　●マルチデータソース連携

4.2　高生産性の実現

高生産性を実現するためのいくつかの必要機能と評価項目を記述する。

□個人の生産性向上
　●基本ビジュアル開発
　●ソフトウエア部品再利用機構
　●外部ソフトウエア連携機構
□チーム開発支援
　●ソースコード管理
　●アプリケーション実行環境構築機構
　●各種のバージョン管理
□メンテナンス支援
　●影響度解析機構
　●自動テストツール
　●オフラインデバック機構

5　ツールのカテゴリー分類

前述した機能分類を利用し、いくつかのツールを検討し、そこで考えられるカテゴリーへの分類を実施した。
その場合の製品カテゴリーは、以下のように大別できると考えて良い。

5.1　パーソナルデータベースシステム構築ツール

個人が利用するデータベースシステムの構築ツールと考えれば良い。

［長所］

□特定のデータベースサーバーを必要としないケースが多く、すぐに開発を始めることができる。
□ロジック記述部分、用意されているオブジェクトが少ないために習得が容易である。
□ある程度の表現能力を持っているケースが多く、プロトタイプなどは効果的に作成することができる。

［短所］

□ランタイム版でもチャージが必要になるケースがあり、クライアントの台数によっては、非常に高額な場合になることもある。
□ツールのディフォルト機能以上のニーズがある場合、実現が困難なケースが多い。
□データベースの検索機構が画一的であり、中規模以上のシステムでは、対応が難しい。
□ソフトウエアを部品化して再利用出来る機能は提供されていない。
□複雑なアプリケーションを作成した場合、実行速度が遅いケースがある。
□全体的にチームでの開発支援機能は準備されていない。
□ソースコードの出力や影響度解析ツールの提供が無いケースが多く、他人の開発したアプリケーションのメンテナンスが難しい。

5.2　クライアント／サーバーアプリケーション開発システム

ここ１～２年間で、急速に整備が進んだソフトウエア群である。

［長所］

□データベースサーバーへのアクセス方法が多様で、データ件数の増大や、トランザクションの増大にも対応することができる。
□他のアプリケーションを取り込む機構が数多く搭載されているケースが多く、表現力の自由度が高い。
□チーム開発、アプリケーションの管理など、ビジュアルツール以外の機能が多く提供されている。
□オブジェクト指向技術、各種のソフトウエア部品が用意されている場合、個人及び、チームでの開発生産性の向上が期待できる。
□非常にプリミティブな画面オブジェクトと、高度のハンドリング機構が用意されているので、カスタマイズ能力が高い。

［短所］

□価格が高いものが多くまた、開発時にも比較的リソース（ＣＰＵ、メモリー）を消費するケースが多い。
□機能が多い分習得に時間がかかる。
□機能カスタマイズの自由度が高い分、あまりやりすぎるとWindows自体の持つ、限界を超えてしまうこともある。限界への考慮が難しい。
□アプリケーション全体での仕様、実装方法の標準化、コーディングルールの設定などを行わなければ、開発、メンテナンスでの労力が大きくなる。また、行う場合、開発管理面でのワークロードが発生する。

5.3　４ＧＬ

UNIXやDOS環境で、ある程度長期間提供されていたアプリケーション開発用ツール。

［長所］

□実績があるものが多く、稼働の信頼性が高い。
□ソフトウエア部品などが提供されているケースが多く、それらを利用することで、開発の効率化を図ることができる。
□比較的カスタマイズが可能なケースが多く、ある程度の処理を３ＧＬに比べれば簡単に行える。
□ＯＳファイルなどへのアクセス、ＯＳ上の資源制御が可能であり、データベースアプリケーション以外への対応能力が大きい。
□ある程度の生産性と、実行速度を確保できるため、個人で利用するユーティリティなどを開発する場合には、有効であるケースが多い。

［短所］

□変更できないオブジェクトとして機能が提供されているケースが多く、カスタマイズできる場合でも限界があるケースが多い。
□予め提供されている機能をカスタマイズする必要がある場合、提供されているオブジェクトがプリミティブではないため、カスタマイズ能力に限界がある。
□各製品とも特有のプログラミング技術の習得が強要されるケースが多く、習得時間が長い。
□データベースの相互運用性、マルチデータソースアプリケーションには対応していない場合が多い。
□ソースコード管理、チーム開発の支援機構が貧弱なケースが多い。
□ソースコードの出力や影響度解析ツールの提供が無いケースが多く、他人の開発したアプリケーションのメンテナンスが難しい。

5.4　３ＧＬ

既存の３ＧＬのことである。

［長所］

□考えられるだけの機能拡張性を持ち、あらゆるニーズに対応することができる。
□４ＧＬ等と比較した場合、高速に動作するケースが多い。
□データベースアクセスなどのについても、それぞれのデータベース毎に最大限のチューニングが可能である。
□リソースの消費量が最も少ない。

［短所］

□メモリー管理、変数内のデータ管理など複雑な処理を行う必要があるケースがあり、本来のアプリケーション開発以外が必要なケースが多い。
□ソースコードを大量に記述するので、物理的なメンテナンスでのワークロードが膨大である。
□各アプリケーションが非常にデリケートな状態となるケースが多く、メンテナンスが難しい。
□チーム開発や、ソースコード分析等の他の機能は、別途ツールを購入して行う必要がある。
□ユーザーの目に見えるようなプロトタイプをすぐに作り出すのは非常に困難である。
□習得に時間がかかる。

6　おわりに

クライアント／サーバー環境でのアプリケーションの開発は、ここ数年間で最も注目を集めているシステム構築方法である。

その一方で、それまで３ＧＬと、データベースベンダーの提供する４ＧＬのような開発ツールのみの状況から、開発環境を専門に扱うソフトウエアベンダーの台頭などにより、非常に混沌とした状況になっていることも事実である。

1994年から95年にかけての開発ツールの状況は、UNIXサーバー、PCサーバー上のRDBMSに対する評価が確立していなかった、４年前の状況と同様の状態、すなわち、不明確な評価基準と、機能評価への偏重といった状況にあると考えてもよいであろう。

いずれにしても、１つのツールで全てのシステムの構築を行うのは現実には不可能な状況であり、システムとツールの適切な組み合わせを選択することこそが、開発プロジェクトの初期の段階で最も重要な要件の内の１つであることは疑問の余地が無いことである。

開発環境を構築するためのソフトウエアが果たす役割は、益々大きくなると考えて良いのではないだろうか？