J2EE と Microsoft .NET の比較

オラクル・ホワイト・ペーパー
2002 年 4 月

ORACLE

J2EE と Microsoft .NET の比較

背景

1990 年代から広がりを見せたインターネットは、業務アプリケーションに対する新しい要件を生み出しました。ブラウザベースのアプリケーション配布によるコスト削減およびビジネス・チャンスの拡大は実証済みです。今日では、パートナは、バックオフィス・システムへの直接アクセスや統合を見込んでいます。従業員は、社内外からの業務アプリケーションへのアクセスを必要としています。顧客は、製品とサービスが 1 日 24 時間、いつでも入手できるものと考えています。

「今日の企業アプリケーション開発市場では、Java 2 Platform Enterprise Edition（J2EE）**とマイクロソフトのプラットフォームという 2 大アプリケーション・プラットフォームが主流となっています。**

どちらのプラットフォームを使用するかはビジネス上の意思決定であるが、技術的な要因がビジネスに重要な影響を与える可能性があります。」

Giga, September 2001

新しい技術の登場により、企業が現在必要としているタイプのアプリケーションを記述できるようになりました。ポータルは、すべての業務アプリケーションへの新しいエントリ・ポイントです。コンテンツ管理システムは、Web サイトの作成を自動化します。個々のユーザーに合わせてアプリケーションをパーソナライズするために、ビジネス・インテリジェンスが使用されています。また Web サービスは、ブラウザ、音声、および無線などのアプリケーションを配布するための最も新しいチャネルです。

技術ベンダーと企業は、この新しい世界で機能するよう、独自のインフラストラクチャを適用させる努力をしてきました。業務要件のこのような変化と平行して、2 つの主なアプリケーション開発モデルが登場しました。Java 2 Enterprise Edition（J2EE）と.NET です。J2EE は、Java アプリケーションのサーバー・サイド・アプリケーション・モデルです。.NET は、マイクロソフトのサーバー・サイド・アプリケーション・モデルから発展した最新のモデルです。

この 2 つのアプリケーション・モデルは、インターネットによる新しい機会を活用するよう設計されています。ただし、これらのモデルは類似のアーキテクチャ原則に基づくにもかかわらず、考え方の背景はそれぞれ異なります。J2EE はオープンな仕様を基準とした広範な導入を目標としているのに対し、.NET は単一のオペレーティング・システム、つまり Windows の導入を基準とした広範な導入を目標としています。このため技術者の間では、J2EE と.NET のどちらを採用するかが継続的な議論の的です。

そして、経営者がこの議論に介入してきました。インターネットベースのアプリケーションは、コスト削減および競争力強化に優れた効果を発揮することは実証済みです。先見の明のある経営者は、情報技術への投資を効果的に行うためには、この問題を自分自身で理解する必要があると認識しています。

はじめに

J2EE と.NET どちらに対しても先入観があり、意思決定する場合の理解と評価を困難にしています。たとえば、製品対仕様共存はできるか、また互換性はあるかどちらがより完成度が高く実績があるか。どちらがより多くの機能を提供するか、などです。

「マイクロソフトの無計画ともとれる用語の使用は、.NET に大きな混乱を招いています。
このため、企業の経営者が.NET を検討する場合は、最初にこの用語が意味する可能性をすべて考慮し、適切なコンテキストで質問に対し回答することが重要です。」

Gartner, April 2001

これらの質問は、それぞれの企業環境のコンテキストに照らし合せて考える必要があります。同種または異種どちらのインフラストラクチャ内で意思決定を行うか。テクニカル・リソースのスキルはどの程度必要か。アプリケーションは自社開発か、外部購入か。新しい技術の採用に対する企業の熱意はどの程度か。どの程度の期間、システムが稼働できるか。

この 2 つのアーキテクチャのどちらを選択するかは、簡単に決めることはできません。短期の戦術的な目標については、どちらのアプリケーション・アーキテクチャでも十分でしょう。長期の戦略的なアーキテクチャについては、より慎重に選択する必要があります。2 つのうちどちらを選択するかで、将来の事業コストと利益は大きく変わります。J2EE か.NET かを決定する前に、これらについてよく理解しておく必要があります。

準備

この 2 つのアーキテクチャを正当に比較するために、次の 2 つの分野で共通の基盤を確立する必要があります。

- **製品対仕様**
  J2EE が複数の製品実装を持つ仕様であるのに対し、.NET では製品と仕様が混在しています。J2EE と.NET の分析では、製品と仕様それぞれを適切に比較する必要があります。
- **アーキテクチャ**
  J2EE と.NET はどちらも、サーバー・サイドのランタイム環境と一連のランタイム・サービスをアプリケーションに提供します。これらは高レベルでは同様に見えますが、詳細に分析すると重要な点で異なります。

製品対仕様

この議論において、まず何を比較するかを理解しておく必要があります。J2EE 自体は、仕様です。複数のベンダーが J2EE 仕様をアプリケーション・サーバーに実装し、この仕様を実装した製品を生産しています。たとえば Oracle、IBM、および BEA は、高い競争力を備える J2EE 認定アプリケーション・サーバーを提供しています。

J2EE は、Java 2 Standard Edition（J2SE）、Java 2 Enterprise Edition（J2EE）、および Java 2 Micro Edition（J2ME）という 3 つのエディションで構成された広範な Java プラットフォームの一部です。これらの Java 仕様は、マイクロソフトを除く多くの主流技術ベンダーが属する Java Community Process という標準化プロセスから発展しました。

.NET とは、広範な提供物を指すマイクロソフトの用語です。プログラミング・レベルでは、J2SE、J2EE、および J2ME と類似した特徴をもつサーバー・サイド・

アプリケーション・モデルを指します。J2EE との比較をわかりにくくしているのは、このプログラミング・フレームワークに基づくすべてのマイクロソフト製品が包括的に Microsoft .NET と呼ばれていることです。

プラットフォームの比較

「J2EE と.NET の共通点は、XML との相互運用性、負荷分散、およびトランザクションなどの Web サービスを構築する際の基礎をあらかじめ備えていることです。

開発者は基礎を各自で記述する必要はなく、これらの複雑なサービスを提供するコンテナの内部で動作するアプリケーションを記述できます。」

The Middleware Company, June 2001

より正当に比較するには、比較の範囲を個々の仕様に限定せず、プラットフォーム全体に拡大する必要があります。つまり、表 1 に示すとおり、アプリケーション・モデル、サーバー実装、製品、およびオンライン・サービスという 4 つのレベルで比較する必要があります。

アプリケーション・モデルは、J2EE と.NET が公開するアプリケーション・プログラミング・インタフェースのセットです。サーバー実装は通常、J2EE サイドではアプリケーション・サーバー、.NET サイドでは Windows オペレーティング・システムです。製品は、各ベンダーまたはサード・パーティにより提供される製品セットです。オンライン・サービスは、ホスティングによる製品提供です。

比較レベル	Java	マイクロソフト
アプリケーション・モデル	J2EE 仕様	.NET フレームワーク
サーバー実装	複数ベンダー	Microsoft Windows
製品	複数ベンダー	複数ベンダー
オンライン・サービス	複数ベンダー	複数ベンダー

表 1: **プラットフォームの比較**

このような高レベルの比較からでも十分に明らかなように、J2EE は、文書化された仕様を基に、複数のベンダーがサーバー実装を行っています。.NET は、唯一のベンダーであるマイクロソフトが単一のベンダー・フレームワークを基に.NET サーバー実装を提供しています。

またこの表には示されていませんが、J2EE 仕様はオープンな業界団体 Java Community Process が開発および管理していることも重要です。.NET は、唯一のベンダー、マイクロソフトがコアとなる.NET 仕様を管理しています。

ここで問題になるのは選択肢です。.NET を選択すると、自動的にマイクロソフトを選択することになります。他に選択肢はありません。J2EE を選択すると、初期および持続的なインフラストラクチャ事業を獲得しようと競合する広範なベンダー群から選択することになります。J2EE 市場のベンダーは、コスト、パフォーマンス、製品層の厚みなど、様々な特徴について開発努力を行っています。.NET を選択すると、比較の対象となる持続的な競合がありません。

アーキテクチャ

J2EE と.NET は、どちらも 3 層アーキテクチャのコンセプトに基づいています。このアーキテクチャの目標は、ユーザー・インタフェースの作成と配布をビジネス・ロジックの作成から分離し、さらに、アクセス対象のインフラストラクチャであるバックエンド・データからも分離することです。

このタイプのアーキテクチャでは、拡張性、信頼性、および可用性を実現する多数のサービスが中間層コンテナに依存して提供されます。実装によって異なりま

すが、このコンテナは、一般にセキュリティ、トランザクション、および接続性に関する豊富なサービスも提供します。J2EE と.NET アーキテクチャは、どちらもこのコア・インフラストラクチャを開発者に提供しているため、開発者が各自で作成する必要はありません。図 1 は、Java アーキテクチャ全体における J2EE の位置付けを表しています。このアーキテクチャは 4 つの大きな層に分かれています。

「標準はベンダーのものではありません。エンドユーザーと購入者のものです。

技術を使用、購入、および管理する立場の人物が、競争力のある代替品に乗り換えることを躊躇しているなら、ベンダーが謳う標準準拠の利点をまず考えてみてください。」

Tech Update, January 2002

1. **プレゼンテーションとアクセス**
   J2EE は、シン・クライアント・アプリケーション用に、タグ指向の動的な HTML ページを生成する Java ServerPages（JSP）とプログラマティック HTML ページを生成するサーブレットを備えています。高性能で複雑なクライアントには、Java Foundation Classses（JFC）を使用します。プログラマティック・アクセスには Web サービスを使用します。

2. **ビジネス・ロジック**
   J2EE は、業務トランザクション用の Session Enterprise JavaBeans や永続性のあるトランザクション・データ用の Entity Enterprise JavaBeans など、拡張ビジネス・コンポーネント・モデルを提供します。非同期の業務プロセスには Message Driven Beans を使用します。

3. **接続性**
   J2EE では、標準のデータベース・プロトコル Java Database Connectivity（JDBC）を使用してデータベースにアクセスできます。ホストベースのアプリケーション・アクセスには、Java Connector Architecture（JCA）を使用します。非同期の業務プロセスには Java Messaging Service（JMS）を使用します。Java と XML プロトコル（SOAP など）をマッピングするために、拡張 Java API for XML が提供されています。

4. **ランタイム**
   すべての Java アプリケーションでは、共通のランタイム Java Runtime Engine（JRE）を使用して業務アプリケーションを実行します。Java コードは JRE を搭載するすべてのプラットフォームで Java バイトコードにコンパイルされ、実行されます。

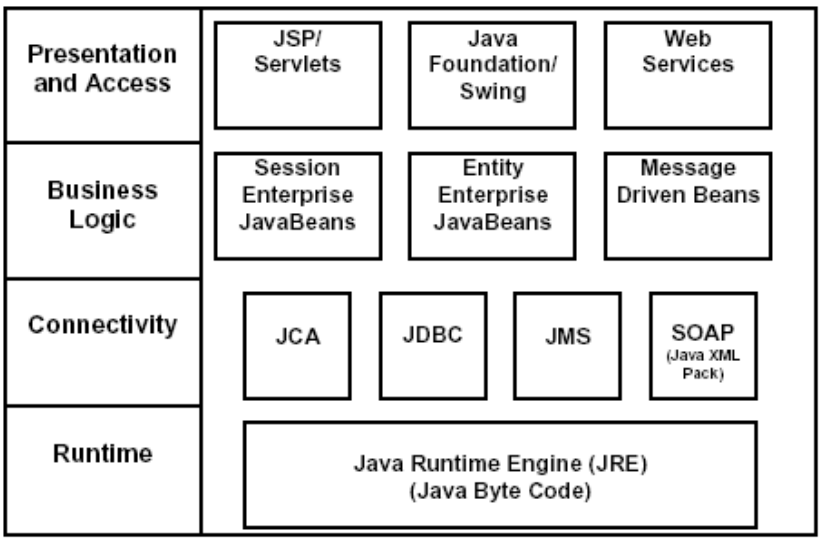

図 1: Java **アーキテクチャの概念**

「マイクロソフトの IL ランタイムに優れた目標があるとすれば、それは、フレームワークに参入する際のプログラミング言語による障壁を排除することです。

一方、Java はプラットフォームによる障壁を排除することです。」

Jim Farley, Oreilly, 2001

図 2 は、.NET アーキテクチャの概念を示しています。このアーキテクチャも、大きく 4 つの層にわかれています。

1. **プレゼンテーションとアクセス**
   .NET では、シン・クライアント・アプリケーションについては、ASP. NET を使用してタグ指向の動的な HTML ページを実装します。高性能で複雑なクライアントには、Windows フォームを使用します。プログラマティック・アクセスには Web サービスを使用します。
2. **ビジネス・ロジック**
   .NET は、同期的な業務トランザクション用の.NET Managed Components と非同期的な業務トランザクション用の COM Queued Components という 2 大ビジネス・コンポーネントを備えています。
3. **接続性**
   データベース・アクセスには、ADO .NET が使用されます。.NET コンポーネントと XML プロトコル(SOAP など)をマッピングするために、XML API が提供されています。
4. **ランタイム**
   すべての.NET アプリケーションでは、独自のランタイム・エンジン Common Language Runtime（CLR）を使用して業務アプリケーションを実行します。アプリケーションは複数の言語で記述して Intermediate Language バイトコードにコンパイルし、CLR で実行できます。CLR をサポートしている唯一のプラットフォームは Windows です。

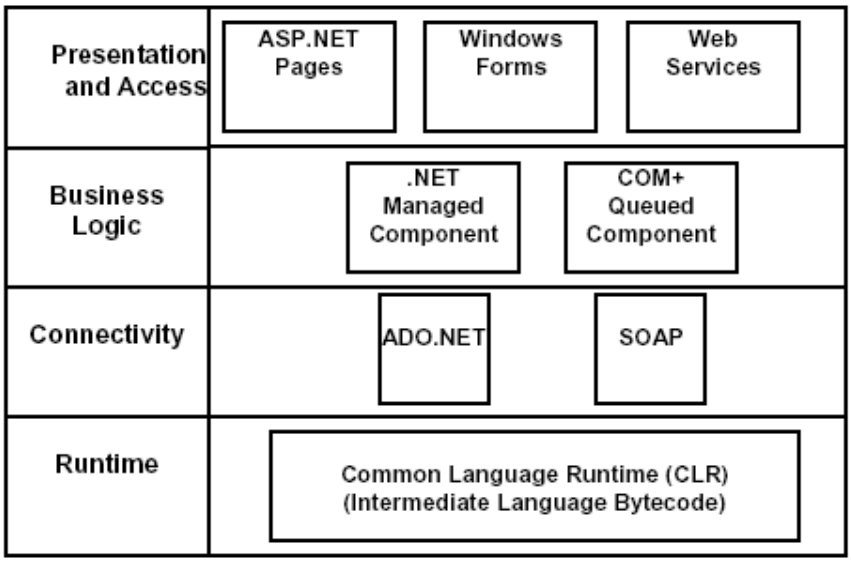

図 2: .NET **アーキテクチャの概念**

アーキテクチャの比較

高レベルでは、2 つのアーキテクチャは類似しています。どちらも 1980 年代と 1990 年代の研究から複数層アーキテクチャに発展したもので、この概念に高度な業務価値を与え実装したものです。

ただし詳細に調べた場合、アーキテクチャの仕様、言語中心性、アーキテクチャの深度、および Web サービスの 4 つの分野で大きな相違点が見えてきます。

アーキテクチャの仕様

「ECMA に提出された.NET フレームワークの一部は、C#言語のコンパイラとそれをサポートするランタイムの作成には十分です。

しかし残念ながら、実際的なソリューションとしては魅力に乏しいという点が問題です。なぜなら、プラットフォーム全体での配置が潜在的に見込まれる、堅固で移植可能な.NET アプリケーションを作成するにはまだ不十分であるためです。」

Mark Driver, .NET Magazine, April 2002

アーキテクチャを分析すると、J2EE と.NET の重要な違いが明らかになります。J2EE アーキテクチャの各アスペクトは、文書化された仕様に定義されています。

開発者は Java Community Process の Web サイト（http://www.jcp.org/ ）で、アーキテクチャの各コンポーネントの詳細な仕様を参照できます。これらの仕様により、開発者はアーキテクチャを理解し、ベンダーはアーキテクチャを製品に実装できます。

仕様の他に、各仕様のリファレンス実装があることも Java Community Process の重要な特徴です。このため、たとえば、J2EE 仕様に付属するスタンドアロンでベンダー非依存のリファレンス実装を使用して、開発者はベンダーの製品が仕様に適合することを確認できます。

これに対し、.NET アーキテクチャには仕様が存在しません。.NET アーキテクチャは、様々なマイクロソフト製品の範囲内で提供されています。Microsoft .NET 製品自体がリファレンス実装となっています。

このような独占的な印象を除去するため、マイクロソフトは、.NET フレームワークの一部を規格団体に提出しました。このうちプログラミング言語 C#は最近ヨーロッパ電子計算機工業会（ECMA）により承認されました。Java Community Process とは異なり、マイクロソフトには、プラットフォーム全体にまで標準化を推進する意図は見えません。今回提出された部分は、.NET フレームワーク 1 の 10%にも満たないと評価する業界アナリストもいます。[1]

言語中心性

言語中心性も、2 つのアーキテクチャの大きな違いです。マイクロソフトは、.NET が 20 言語以上をサポートしていると説明しています。J2EE 環境では、Java がプログラミング言語であり、その他の言語は、Java Native Interface API、または Java Connector Architecture などの接続 API を介して利用できます。

Microsoft .NET 環境では、Visual Basic .NET と C#という 2 大言語が業務アプリケーションに奨励され、使用されているというのが現状です。Visual Basic .NET は、マイクロソフトが同社の最も一般的な言語である Visual Basic を.NET に合わせて大幅にリライトしたものです。また C#も、Visual Basic .NET よりも低レベルで、C++ほど複雑でないプログラミング言語を開発者に提供するために、同社が.NET プラットフォーム用に特別に作成したものです。この結果、マイクロソフト・コミュニティではどの言語を使用するかについて大きな混乱が発生しました。

[1] Giga Application Platform and Development Strategies 2002,Florida, March 12, 2002.

「…Giga Information Group（本社: マサチューセッツ州ケンブリッジ）では、グループ内の 78%が、Web サービスを構築および配置するための最も効率的なプラットフォームは J2EE（Java 2 Enterprise Edition）サーバー・ソフトウェアであると判断していることが判明しました。

一方、Windows サーバー・オペレーティング・システム用の Web サービスを構築できるマイクロソフトの.NET を支持したのは、22%でした。」

InfoWorld, December 2001

Java 開発者コミュニティは、今日、プログラミング言語の中で最大の団体であると認識されています。たとえば IDC では、Java 開発者数が 2004 年までに 400 万人を突破すると予測しています。[2]Java 人気の原因は、J2EE アーキテクチャを採用しているうえに、世界中の技術教育システムで入門および上級どちらのプログラミング・クラスでも Java が一般的に使用されていることです。

さらに低レベルでは、.NET の多言語サポートの基盤である Common Language Runtime エンジンが議論の的となっています。.NET は確かに優れた技術ですが、大規模な配置ではまだ実績がありません。製作者は CLR について、マネージコード対アンマネージコード、および全体のパフォーマンス、信頼性、および拡張性という 2 つの分野について憂慮しています。

非常に多くの共通のランタイムを言語に提供するために、マイクロソフトは個々の言語のネイティブな特長やデータタイプの多くを犠牲にする必要がありました。マイクロソフトは、この問題を回避するためにアンマネージコードのコンセプトを提供しています。アンマネージコードを使用することで、ネイティブの言語機能へのアクセスが可能になりますが、.NET との互換性は開発者に委ねられます。

さらに、パフォーマンス、信頼性、および拡張性に関する問題が重要です。CLR は新しい技術であるため、.NET の大規模な配置で使用された実績がほとんどありません。CLR の大規模な配置に向けた豊富な経験とベスト・プラクティスが蓄積されるまでにはまだ数年かかるでしょう。

それに対し、Java Runtime Engine は、多数のプラットフォームを持つ企業レベルで 7 年にわたり配置されてきた実績を持ち、今後も引き続き実際の経験をもとにしたアーキテクチャの強化が見込まれます。また、Java Native Interface を使用した多言語サポートや、Java Connector Architecture および Java Messaging Service など相互運用性標準に対する実証済みの技術が存在します。パフォーマンスの問題への取組みは、高度な拡張性、信頼性、およびパフォーマンス指向の環境で JRE を管理するため十分に理解されたベスト・プラクティスを通じて実施される他、JRE 提供者の熾烈な競争市場によっても行われています。

アーキテクチャの深度

この 2 つのアーキテクチャについて製作者が注目する主なポイントは、J2EE では細粒度のサービス・セットがビジネス・コンポーネント・レベルおよび接続性または相互運用性レベルで定義されていることです。

J2EE は、業務トランザクション用の Session Enterprise JavaBeans と永続性のあるトランザクション・データ用の Entity Enterprise JavaBeans という 2 つのオプションを、ビジネス・コンポーネント・レベルで提供します。Session Enterprise JavaBeans では、ステートレスなビジネス・オブジェクトとステートフルなビジネス・オブジェクトの両方をサポートします。一方、.NET ではステートフルなビジネス・オブジェクトはサポートせず、永続性のあるビジネス・データ用のコンポーネント・モデルも提供しません。

[2] IDC, June 2001

「… あなたが ISV またはコンサルティング会社に勤務していると仮定して、どのプラットフォームを選択するかを考えてみてください。

恐らく契約のチャンスを失わないよう、顧客にそれぞれのプラットフォームをそのまま使用させるため、J2EE の背後にあるアーキテクチャを採用するでしょう。」

The Middleware Company, June 2001

マイクロソフトは、これらのサービスを提供しない理由について、これらの機能は多くのアプリケーションで必要とされないためであるとしています。これに対し J2EE 側では、これらのアプローチが適している一般的なビジネス・シナリオやテクニカル・シナリオは多く存在するため、開発者は、自身でプログラミング・パラダイムを創出するのではなくアーキテクチャに組み込まれているものを使用できることが必要だ、と反論しています。

接続性および相互運用性レベルでは、J2EE 仕様は、Java Database Connectivity（データベース用）、Java Messaging Service（同期接続用）、および Java Connector Architecture（企業情報システム用）など多数の API を提供しています。マイクロソフトは、Microsoft Message Queuing（MSMQ）などの技術と Microsoft Biztalk などの製品が混在したものを提供しています。

ベンダーが 1 社の Microsoft .NET 環境での相互運用性実装と異なり、Java プラットフォームでは標準相互運用性 API の存在によりベンダー間の市場競争が生まれ、これらの仕様に準拠するとともに、差別化された実装および付加価値のある製品が提供されています。また、Oracle、SAP、および People Soft などのパッケージ・アプリケーション・ベンダーがコア・アプリケーション・アーキテクチャに Java を採用したことも、この市場の活性化に寄与しています。

製作者にとって、これらの違いは、マイクロソフト・プラットフォームよりも高度に記述、文書化、および実装される傾向にある機能が J2EE アーキテクチャにあることを示しています。このように幅広い接続性および相互運用性の機能があるということは、ある程度、マイクロソフト製品開発プロセスの参加者よりも多彩なベンダーが存在することを表しています。

Web サービス

Web サービスは、多くの企業からの注目を集め、投資の対象となっています。企業は、XML ベースのプログラマティック・アクセスをバックエンドの業務アプリケーションに提供することで、新しく費用効果の高い方法でこれまでの技術投資を再利用できます。

J2EE と.NET はどちらも拡張 Web サービス・インフラストラクチャを提供します。J2EE の Web サービスは、すでに実証済みのアーキテクチャを Java XML Pack とも呼ばれる Java Application Programming Interfaces for XML という標準セットを使用して、単純に無理なく拡張したものです。.NET の Web サービス・サポートは、ベースとなるマイクロソフト・インフラストラクチャを大幅にリライトした結果得られたものです。

このため、マイクロソフトが.NET Web サービス・インフラストラクチャを独自の方法で拡張し、ユーザーをマイクロソフトのみの環境に封じ込めようとするのではないか、と懸念されています。この分野での独自の傾向に対抗するため、Web サービス実装間の相互運用性を保証するベンダー中立団体 Web Services Interoperability Organization（WS-I）が創設されました。WS-I は、この分野での標準準拠を保証するために、Oracle、IBM、およびマイクロソフトなどのベンダーによって共同で設立されました。

意思決定の理解

これまでの説明では、J2EE と.NET を技術的に比較するための準備に焦点を当ててきました。このような技術的な分析に加え、数あるアーキテクチャ中から特定のアーキテクチャを選択する意思決定について、ビジネス・リスクおよびテクニカル・リスクの観点から説明します。

「… Java の使用は単に拡大を続けているだけでなく、E-ビジネスにおける IT 事業の主流を成すプラットフォームとなっています。

回答者の 80%は、アプリケーション開発（AD）企業で Java テクノロジを使用していると報告しています…」

Gartner Summit Survey, February 2001

ビジネス・リスク

企業の環境

各企業には既存の環境があり、新しい、全社的なアーキテクチャとの統合が必要です。過去の経験、既存の投資、技術へのアプローチおよびその他の要因が意思決定の決め手となります。J2EE と.NET それぞれの特徴によって、最も適合する企業環境が決まります。

同種の技術環境を持つ企業では一般に、1 つのテクノロジを採用する傾向にあります。企業のインフラストラクチャ全体が Windows ベースである場合には、.NET にアップグレードするのが合理的な選択です。逆に、Unix の同種環境または複数のオペレーティング・システムおよびハードウェア・プラットフォームが混在する異種環境を持つ企業では、どのようなプラットフォームとオペレーティング・システムでも動作する Java アプリケーションの機能を理由に、通常 J2EE を選択します。

興味深いことに、Windows 中心の環境にとっても J2EE は魅力ある選択肢です。各 J2EE ベンダーは、Intel アーキテクチャが費用効果の高いハードウェア・プラットフォームと見られているため、Windows および Linux でのアプリケーション・サーバー実装を提供しています。J2EE をこのコンテキストで使用することで、企業は J2EE の柔軟性を得られるとともに、低コストな Intel ハードウェアの利点も享受できます。さらに、必要に応じて別のハードウェア・プラットフォームに移行することもできます。

ベンダーの選択

ベンダーの選択および変更が可能であることは、アーキテクチャにとってビジネス上の価値の高い特徴です。競合製品がなければ、企業は特定の技術供給者に固定され、アーキテクチャ全体の再構築を余儀なくされる可能性があります。

この領域で、J2EE の移植可能性は、ベンダーを選択できるという点で大変魅力的です。2002 年度には、37 を超えるベンダーが Java ライセンスを取得し、17 社が Java 互換のアプリケーション・サーバーを発売しました。これは、JBoss など無償のオープン・ソース・アプリケーションから Oracle9*i* Application Server など企業クラスのアプリケーション・サーバーまで様々です。これにより、熾烈な競争を伴う Java 開発ツール市場も出現しました。

この選択により、企業は特定の価格ポイントで最良の機能を提供するベンダーを選択できるようになりました。さらに、1 つのベンダーが期待に応えられなかった場合には、競合ベンダーの実装に変更できるという交渉手段も使えます。

またベンダーの選択は、アプリケーションの開発および配置環境全体への投資においても役立っています。Microsoft .NET では、開発および配置に使用できるオペ

レーティング・システムは Microsoft Windows のみです。この結果、マイクロソフトの価格ポイントからのみ選択が可能となっています。

J2EE では、低価格の Intel ベース機で開発およびテストを行い、多様なハードウェアおよびオペレーティング・システムにシームレスに配置するというのが一般的なシナリオです。企業は、多様な価格ポイントから選択でき、これまでの投資を無理なく活用し、インフラストラクチャへの投資を要件に合わせて直接調整できます。

プラットフォームの完全性

コア・レベルでは、J2EE と.NET はどちらも業務アプリケーションを配置できるインフラストラクチャです。多くの場合、インフラストラクチャ（J2EE または.NET）を提供する特定のベンダーが選択され、そのベンダーがそのインフラストラクチャの製品も提供します。

表 2 は、Oracle9*i* Application Server をベースにした J2EE インフラストラクチャの配置と、Microsoft .NET の配置の例を示しています。この比較で注目する重要な点は、次の 2 つです。

1. J2EE と.NET アーキテクチャの比較は、製品レベルの比較にまで拡大します。Oracle などのベンダーは、インフラストラクチャ・レベルにかぎらず、機能面でもマイクロソフト製品と競合しています。
2. J2EE 領域のプラットフォーム・ベンダーは、ベストブリードの機能レベルでも他の J2EE 互換製品の広範な市場と競合します。J2EE 市場ではこのレベルの競合があるため、Oracle などのベンダーは継続的に革新を続けますが、Microsoft .NET などの単一ベンダーにより制御されたアーキテクチャにはこのような競争に対する努力はそれほど顕著ではありません。

機能	Java	マイクロソフト
コンポーネント・モデル	J2EE	.NET
アプリケーション・サーバー	Oracle9*i* Application Server	Windows
データベース	Oracle9*i* Database	SQL Server
開発ツール	Oracle9*i* Developer Suite	Visual Studio.NET
ポータル	Oracle9*i*AS Portal	Sharepoint Portal
クライアント管理	Oracle9*i* Internet File System	Content Manager
ビジネス・インテリジェンス	Oracle9*i*AS Business Intelligence	**提供なし**
キャッシング	Oracle9*i*AS Cache	ISA Server
モバイル	Oracle9*i*AS Wireless Edition	Mobile Information Server

表 2: Oracle9*i* Application Server **と** Microsoft .NET

リソースとスキルの可用性

J2EE 対.NET の議論は、基本的に、アーキテクチャへの長期投資および戦略的な業務アプリケーションの実装についての議論です。ただし、アーキテクチャを開発、配置、および管理する熟練リソースがいなければアーキテクチャは不十分になります。この分野について J2EE と.NET は、どちらも十分なリソースを有しています。ただし、それぞれに考慮事項があります。

「.NET への移行は、スキル、アーキテクチャ、およびツールの刷新により深刻な中断を引き起こし、2004 年度中は開発者に問題を投げかけることになるでしょう。」

Gartner, July 2001

J2EE アーキテクチャは、1997 年以来使用されており、多数の開発者および製作者を抱えています。最近の Gartner による調査では、80%以上の IT 企業がアプリケーション開発部門で Java を使用していることが明らかになりました。[3]このように長期にわたって幅広く採用されてきたことから、開発企業は熟練した Java リソースを簡単に見つけることができます。

マイクロソフトも伝統のある巨大な開発者コミュニティを有しています。ただし、マイクロソフトの大きな問題は、.NET 開発環境の最新リリース Visual Studio .NET では、従来の製品のアーキテクチャが大幅に見直されていることです。この結果、マイクロソフト最大の開発者コミュニティである熟練した Visual Basic 開発者のようには、経験を積んだ熟練.NET 開発者を見つけにくくなる可能性があります。

熟練したサード・パーティに訓練や実装の支援を依頼して、J2EE および.NET 開発者に対する社内の熟練開発者の不足を補うというのも 1 つの方法です。J2EE および.NET 両方のアーキテクチャを専門とするコンサルティング会社も存在します。どのコンサルティング会社に対しても、特定のアーキテクチャについてコミットできるかどうかを質問する必要があります。J2EE は Windows を含む多数のプラットフォームに幅広く採用できるため、コンサルティング会社は市場機会が充実している事実および柔軟性の高さを考慮して、J2EE を選択する可能性があります。

リソースの可用性について考慮する必要のある最後の分野は、選択したプラットフォームでのパートナが存在するかどうかです。パートナは、独立系ソフトウェア・ベンダー、インフラストラクチャ・パートナ、システム・インテグレータ、およびサービス・プロバイダという 4 カテゴリに分類できます。これらの分野に厚みのないプラットフォームを J2EE または.NET で提供しても、技術実装の質にかかわらず成功は望めません。たとえば、Oracle9*i* Application Server を選択することで、900 を超えるパートナから J2EE 製品の提供を受けられます。

テクニカル・リスク

アーキテクチャ・リスク

業務アプリケーション配置に関する意思決定を行う前に、J2EE または.NET アーキテクチャから得られるサービスの品質を慎重に分析する必要があります。長年にわたり配置成功の実証記録を積み上げてきた J2EE と、基盤とするプラットフォームを再構築した.NET はどちらもこの分野を強みとしています。

.NET 最大のアーキテクチャ・リスクは、唯一のベンダーであるマイクロソフトへの依存性です。信頼性、拡張性、可用性、およびその他のサービス特性の品質が.NET プラットフォームに追加された場合は、このような機能を必要とするマイクロソフト・カスタマには有益です。ただし、これらの機能が追加されるかどうかはマ

[3] Gartner, February 2001

イクロソフト次第です。

「マイクロソフトの.NET プラットフォームは、PC ベースの Windows からインターネットベースの Web サービスの世界へのシフトを表しています。ただし、.NET 単体では企業のインターネット戦略の基盤とはなり得ません。」

Gartner, November 2001

一方 J2EE 市場では、サービスの品質はベンダーによって大きく異なります。たとえば、Oracle9*i* Application Server Release 2 はキャッシュ・レベル、Web サーバー・レベル、Web コンポーネント・レベル、および Enterprise JavaBean レベルでのクラスタリング機能を用意しています。これとは対照的に、ユーティリティ・アプリケーション・サーバーが提供されていても、Oracle9*i* Application Server など実証済みの環境への配置が見込まれている J2EE 開発環境もあります。

プラットフォームの完成度

アーキテクチャの完成度は、プロジェクトが成功するかどうかの重要な鍵となります。完成度には、ベスト・プラクティス、高度な熟練リソース、およびアプリケーションの開発および配置の一般的な問題に対する理解が伴います。

この分野は、J2EE が最も得意とするところです。5 年にわたる企業での配置と 37 以上の各種 J2EE ライセンスにより、完成度の高いプラットフォームとなっています。また、この市場での実績を立証するため、J2EE 利用者が一連の J2EE ベスト・プラクティスを公開し始めました。これは J2EE Design Patterns とも呼ばれ、一般的な設計上の問題を J2EE の観点から解決する方法が説明されています。

これに対して Microsoft .NET はまた開発途中のプラットフォームです。2002 年初頭まで.NET の中で本稼働にこぎつけたのは Visual Studio .NET 開発環境だけでした。マイクロソフトでは、対応する Windows .NET サーバーを今後数年にわたって発表していく長期計画を立てています。このため、Microsoft .NET は技術的には優れていますが、未完成のプラットフォームであることは否めません。

開発

マイクロソフトはこれまで開発ツールの優れたベンダーとして知られています。.NET とともに、それらのツールを再構築し、Visual Studio .NET を完成させました。このツールは、マイクロソフト・プラットフォームのアーキテクチャ上の変更が完全に活かされるよう設計されています。この分野でのマイクロソフトの強みは、シームレスに統合された開発環境を開発者に提供してきたことです。

Java 市場での開発者の大幅な増加は、J2EE ツール市場に熾烈な競合をもたらしました。Visual Studio .NET のみのマイクロソフト市場と異なり、J2EE 市場には、オープン・ソース・フレームワークから全ライフサイクル統合型の開発環境まで、多様な開発ツールが存在します。

「マイクロソフトの新しい Visual Studio.NET のリリースをもって、.NET の正式な発売開始となりました。この製品は、複数の言語、多彩なクライアント、および Web サービスをサポートします。

ただし、.NET は学習曲線の傾斜が急であるため、企業は、かわりに Java 2 プラットフォームへの切替えを考慮する必要があります。」

Forrester, February 2002

Oracle9*i* JDeveloper を供給する Oracle などの大規模ベンダーでは、統合型でエンドツーエンドの J2EE 機能に加え、オープン・ソース、サード・パーティ・ツール、および J2EE フレームワークがシームレスに協働する拡張可能な開発プラットフォームも市場に提供しています。

ツール市場の競合がこのように激化した結果、J2EE と.NET のどちらが優位に立つかわからなくなりました。サーバー・サイドの議論では、開発ツールで最も重要な争点は、J2EE プラットフォームでより多くの選択肢と価格ポイントを利用するか、.NET プラットフォームでより少数の選択肢を利用にするかに絞られてきているようです。

共存

多くの企業では、長期の共存方針または短期の移行環境として、Microsoft .NET と J2EE 環境の両方を同時にサポートするよう迫られるでしょう。このシナリオではプラットフォームの統合によるシナジーの利点が完全に活かされませんが、Oracle などの J2EE ベンダーは、この方針を成功させるために多くの投資を行っています。

たとえば、Oracle9*i* は、J2EE の利点を完全に活かすことを目的としたアーキテクチャを備えており、Microsoft .NET 環境にもシームレスに統合できるよう設計されています。多くの顧客が Oracle9*i* を使用して、Microsoft .NET 環境のコスト削減や信頼性、可用性、および拡張性の向上を実現しています。Oracle9*i* が備える共存および相互運用性の機能の一部を次に示します。

- ADO .NET、OLE DB .NET、および ODBC .NET のネイティブ実装によるデータベース統合
- Microsoft Transaction Server による、Microsoft .NET の移行環境での統合
- Microsoft Internet Information Server を Oracle9*i* Application Server の Apache Web Server にプロキシする機能
- ASP .NET コンテンツを Oracle9*i* Application Portal に含める機能
- Oracle9*i* Application Server Cache を使用してマイクロソフト.NETWeb コンテンツをキャッシュする機能
- Oracle9*i* JDeveloper を使用した開発期間と Oracle9*i* Application Server Web Services を使用した本稼働時の両方で、Microsoft .NET Web サービスと相互運用する機能
- Java Connector Architecture、Java Messaging Service など標準の J2EE API を介した相互運用性

Web サービス

J2EE と Microsoft .NET を検討する際、Web サービスを使用することで、いずれか一方を選択する必要はなくなるという考え方もあります。Web サービスが相互運用性に主眼を置いていることから、J2EE の Web サービスが.NET の Web サービスと簡単に通信できると考えられます。

Web Services Interoperability Organization のような著名な団体があることから、Web サービスの相互運用性はベンダーの説明以上に高いと思われます。たとえば Oracle は、.NET Web サービスとの相互運用性をテストして、Oracle9*i* Application Server の顧客が Microsoft .NET への投資と Microsoft .NET Web サービスを活用できることを保証しています。

ただし、業務アプリケーションは Web サービスとして公開されているかどうかにかかわらず、QoS、トランザクション、セキュリティ、およびその他の企業クラスのサービスが必要とされており、これらは Web サービス自体ではなく J2EE や.NET など基盤アーキテクチャの特徴です。戦略レベルでは、Web サービスは単に J2EE 対.NET の意思決定の上に位置するもう 1 つの層であり、それを 2 つのアーキテクチャを選択する場合の決定要因とするのではなく、実際的な利点を評価する必要があります。

Oracle の戦略指針

Oracle では常に、オープンな業界標準を使用してベストブリードの製品を実装することで、高い競争力をもつ差別化を図ってきました。1979 年以来、Oracle は、複数のプラットフォームおよび複数のオペレーティング・システムで動作する製品を提供しており、顧客は、唯一の選択肢に縛られることなく最良の価格ポイントおよび機能を選択できます。ただし、オープンな標準のみではベンダー選択の決定的な要因になりません。Oracle は、これらの標準を使用して、高いパフォーマンス、信頼性、可用性および拡張性のあるソフトウェアも生産し、業界をリードしてきました。

JavaJ2EE を選択する決め手となったのは、Oracle の従来の製品がクロス・プラットフォーム、複数オペレーティング・システムおよびベストブリード製品、高パフォーマンスである点でした。Oracle は 1995 年、Java の支持を表明し、1997 年にインフラストラクチャ全体への J2EE 仕様の実装を開始しました。

Oracle は、Oracle9*i* Application Server、Oracle9*i* Developer Suite および Oracle9*i* Database という J2EE アプリケーションの開発、配置、および管理環境を提供する 3 つの製品を提供しています。さらに、業務アプリケーションの拡張セット Oracle E-Business Suite を提供し、同じ J2EE インフラストラクチャを基盤とする Enterprise Resource Planning および Customer Relationship Management の両方をカバーしています。

Oracle9*i* Application Server

Oracle9*i* Application Server は、完全に標準に準拠したアプリケーション・サーバーで、J2EE および Web サービス・アプリケーションが動作するための完全な統合型プラットフォームを提供します。J2EE 1.3 仕様に完全に準拠しているため、開発者は Oracle9*i* Application Server を使用して、複数のハードウェアおよびオペレーティング・システム・プラットフォームにおける J2EE アーキテクチャをフルに活用できます。

Oracle9*i* Application Server で使用される J2EE コンテナは、軽量で高いパフォーマンスを提供します。図 3 のベンチマークに示すとおり、Oracle9*i* Application Server では、互換 J2EE アプリケーション・サーバーの 2 倍の速度でアプリケーションが動作します。同様に、図 4 の Microsoft .NET との比較ベンチマークに示すとおり、同等のアプリケーションに格段に優れたパフォーマンスを提供します。

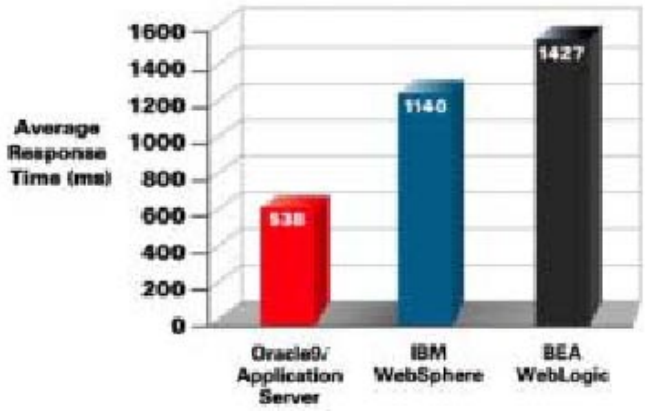

図 3: Oracle9*i* Application Server J2EE **ベンチマーク**

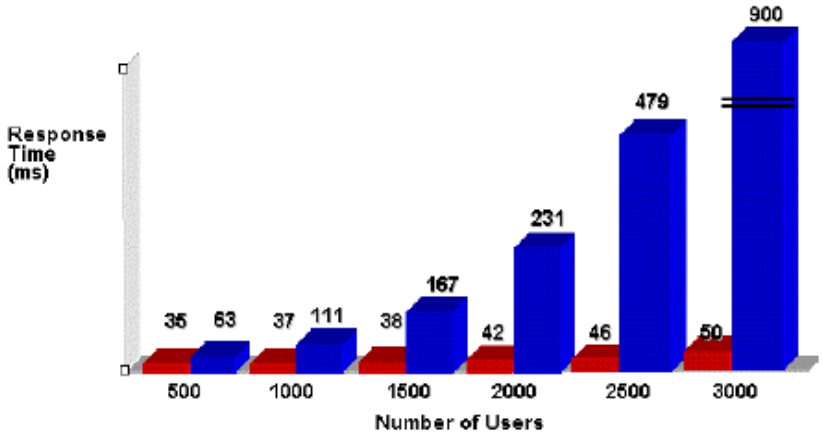

図 4: Oracle9*i* Application Server **と**.NET **との比較ベンチマーク**

また J2EE 市場でも、Oracle9*i* Application Server は信頼性、可用性、および拡張性の点で優れています。高度なクラスタリング・テクニックおよび組込みのキャッシング・テクノロジを使用して、Oracle9*i* Application Server は市販のアプリケーション・サーバーに業界で最高品質のサービスを提供します。

Oracle9*i* Application Server の J2EE インフラストラクチャは、完全な統合型プラットフォーム製品の基盤となります。J2EE を使用してポータル、無線、ビジネス・インテリジェンス、管理、セキュリティ、および統合機能を構築することで、顧客は Oracle9*i* Application Server の統合性および管理性を最大限に活用できます。図 5 は、さらに広い視野から見た Oracle9*i* Application Server の概要を示しています。

図 5: Oracle9*i* Application Server **の機能**

Oracle9*i* Developer Suite

「最初から最後まで Java **で記述することで、**Oracle **は、高度な** UML **モデリング、プロジェクト管理、リモート･デバッギング、実行およびメモリーのプロファイリングの統合、さらには組込みの** Oracle9*i*AS **アプリケーション･サーバーへのワンクリック配置に必要な機能まで、すべての統合を実現しています。**

Oacle9*i* JDeveloper **は、これまでにない最も応答性に優れ、完全で、最良の統合** Java **開発環境です。」**

Rick Ross, Java Developer Journal

Oracle9*i* Developer Suite は、J2EE 業務アプリケーションおよびサービスの短期間での開発を可能にする、業界最高の完全な統合型開発ツール･スイートです。Oracle9*i* Developer Suite は、Java、XML および SQL という 3 つの業界標準が備える機能を組み合せることで、業務アプリケーションおよびサービス指向のアーキテクチャの完全なソリューションとなっています。Oracle9*i* Developer Suite の主なコンポーネントは、Oracle9*i* JDeveloper です。

Oracle9*i* JDeveloper は､最新の J2EE 標準をサポートする統合型 Java 開発環境です。racle9*i* JDeveloper による J2EE アプリケーション開発は､共通 Java プログラミング･タスク、UML モデリング、チーム･サポート、コード･インサイトのための多数のウィザードを使用して簡単に行えます。開発者はこれらのウィザードを使用して、Java の記述、J2EE デプロイメント･ディスクリプタの宣言編集、および独自のデバッギングおよびプロファイリングが可能です。

また Oracle9*i* JDeveloper では､J2EE 開発を加速するためのビジュアル･ワークベンチを備える J2EE Design Patterns フレームワークを提供しています。開発者はこのフレームワークを使用して、業界最高のベスト･プラクティス･アプリケーションを開発できます。このフレームワークは、オブジェクト･リレーショナル･マッピング、ユーザー･インタフェース･プレゼンテーション･バインディング、およびビジネス･ロジック開発を自動化します。

Web サービスについては、J2EE コンポーネントを基に Web サービスを開発する機能を備えています。Java クラス、サーブレット、Enterprise JavaBeans およびストアド･プロシージャはすべて、SOAP、（Simple Object Access Protocol）、WSDL（Web Services Definition Language）および UDDI（Universal Description, Discovery and Integration）を使用して Web サービスとして公開されます。Oracle9*i* JDeveloper で開発した Web サービス･クライアントは、.NETWeb サービスと完全に相互運用します。

Oracle9*i* データベース

Oracle9*i* Database Server は、多数の断片化された異種のデスクトップおよび旧データ管理システムのデータまたはインターネット・コンテンツを統合および管理する総合的なデータ管理インフラストラクチャを提供します。この製品は、テラバイト級のデータ・ストレージに拡張が可能で、実績があり高いパフォーマンスを誇るデータベースです。

J2EE と.NET の視点からは、このデータベースは、業務トランザクションの永続メカニズムです。Oracle9*i* Database は、J2EE、.NET いずれのアーキテクチャ上でも動作するよう最適化されています。

J2EE 開発者に対しては、Oracle9*i* は、高パフォーマンスの JDBC と SQLJ データベース・アクセス・ドライバを提供します。また、Java 開発者は、Oracle9*i* Database の組込み Java エンジンを使用して、ストアド・プロシージャでビジネス・ロジックを作成できます。

結論

J2EE と.NET のどちらを選択するかは、プラットフォームに関する企業の戦略的な意思決定であり、テクノロジに関する戦術的な意思決定ではありません。技術的な問題の中には、分析を要する明らかに重要なものもありますが、最終的には、ビジネス上の問題を理解したうえで意思決定を行う必要があります。選択するアーキテクチャによる将来のコスト、柔軟性およびリスクへの影響は大きく、場合によっては初期投資よりも重要です。

Oracle は 1995 年、Java プラットフォームから得られる明確なビジネス上の利点から Java の採用を決定しました。業界をリードする Oracle9*i* Application Server、Oracle9*i* Developer Suite および Oracle9*i* Database 全体に Java および J2EE を深く統合できたことは、投資の成果です。

近日、Microsoft .NET アーキテクチャの登場により、この意思決定の正当性が再確認された形となりました。独自の単一ベンダー・アプローチを取る Microsoft .NET よりも、オープンな標準、選択、移植性、および相互運用性の基礎となるビジネスおよび技術的な価値を備えるという点で、J2EE が圧倒的に有利です。

J2EE と Microsoft .NET の比較 Oracle Corporation 発行の「J2EE and Microsoft .NET」の翻訳版です
2002 年 4 月
著者: Mike Lehmann

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

海外からのお問合せ窓口:
電話: +1.650.506.7000
ファックス: +1.650.506.7200
www.oracle.com