ミドルウェアに期待される役割と特性 ——MIDMOST® for .NET からみた考案

Expected Roles and Characteristic of Middleware ——Consideration seen from "MIDMOST® for .NET"

菅谷俊郎，白木和彦

要　約　オープンシステムのミドルウェアに求められる役割として，「基盤要素技術の隠蔽」「汎用的に利用できる共通機能の提供」がある．これに加え昨今は「技術・制約・要件の変化に対して迅速にシステムを対応させることが可能な柔軟性」という特性が求められる．日本ユニシスはミッションクリティカルな業務システムを .NET Framework 上で構築・運用することを支援するミドルウェア MIDMOST® for .NET を提供している．当製品は，IT システムがライフサイクル全般において，高いサービスレベル（高信頼性・高可用性・高管理性）を満たすことを可能にするための共通機能を，簡単かつ柔軟に利用できる形で提供している．

Abstract　There are" Concealment of core technologies "and" Offering of common functions for general used " as the role required for the middleware of the open system. Recently in addition to above, the characteristic " Flexibility for the system to be able to respond to changes in the technology, the restrictions, and the requirement promptly "is required. Nihon Unisys provides the middleware" MIDMOST® for .NET "that supports the building and operations of the mission-critical business system across the .NET Framework. This product is also provides the easy and flexible use of common functions that allows the IT system to meet the high level of services（high reliability, high availability, and high manageability）over the entire life cycle of system.

1. はじめに

日本ユニシスのシステム開発の歴史は 40 年前のメインフレーム時代にまで遡る．当時の技術革新の流れは緩やかで，ホストコンピュータの OS や基盤要素技術は基本的に変わることがなく，機能追加などのバージョンアップが発生する際にも，上位アプリケーションに極力インパクトを与えないよう配慮してきた．ホストコンピュータは非常に高価で貴重な存在だったため，限られたハードウェアリソースを最大限に活用するための工夫や知恵が，OS や基盤要素技術の機能として提供されていた．このような状況においてミドルウェアは，開発者に負担をかけることなくハードウェアリソースのポテンシャルを引き出し，高度な信頼性，可用性，保守性が要求されるミッションクリティカル業務を実現するための必要不可欠なソフトウェアとして開発され，長期間にわたり利用されてきた．

1990 年以降，Unix や Windows などのオープンプラットフォームの台頭，オブジェクト指向技術の浸透，開発ツールの洗練により，システム開発は変革の時代に入った．新しいソリューションが次々に登場し，拡張性や応用性に乏しいものは淘汰されていった．そのような中，日本ユニシスは，メインフレームに求められる高いサービスレベルを維持しつつ，オープンプラットフォームの柔軟性や俊敏性を兼ね備えたシステムを構築するための開発方法やソリュー

ションを模索してきた．

また 2000 年以降，Oracle や SQL Server のデファクトスタンダード化，ハイエンドサーバ ES 7000 の登場，マイクロソフト社の .NET 構想発表などが相まって，オープンプラットフォームでミッションクリティカル業務を実現するための土壌が整ってきた．

本稿では，日本ユニシスのシステム開発においてミドルウェアがどのように位置付けられてきたか，最近の開発環境や実行環境の飛躍的進化に際してミドルウェアがどうあるべきかを考察した後，日本ユニシスのミドルウェア製品である MIDMOST® for .NET のコンセプトや機能について紹介する．

なお，本稿では近年のシステム開発について，Windows プラットフォームと .NET Framework にのみ言及しており，Java 環境の考察は次の機会に譲る．

2. ミドルウェアの歴史と位置づけ

本章では，日本ユニシスにおけるミドルウェアの歴史を紹介するとともに，メインフレーム時代のミドルウェアの位置付け，そしてメインフレームからオープンへのプラットフォームの変遷にともなうミドルウェアへの影響について述べる．

2.1 日本ユニシスにおけるミドルウェアの歴史

日本ユニシスはメインフレームの時代から業界や業種を問わず様々な企業システムを構築してきた．当時の企業システムの中心である，端末からの要求をホストコンピュータで処理するオンラインシステムを開発するには，ホストコンピュータや周辺装置に固有の専門知識が必要であった．しかしオープンプラットフォームと比べると技術資料の入手が簡単ではなく，短期間に専門知識を習得することは困難であった．また，開発プロジェクトごとに同じような機能を一から開発するのは非効率的であり，共有化することで生産性向上を図る必要もあった．

このため日本ユニシスでは，開発者が少ない負担で高度なオンラインシステムを構築できることを目的として，データ通信制御，データベース制御，リカバリなどのオンライン中枢機能を統合したミドルウェアを開発してきた．

メインフレーム黎明期の BOSS-11 や全盛期の AIS 1100 では，オンライン処理で必要となる遅延更新方式によるデータベース回復やトランザクション管理，実行優先度制御などを極力ブラックボックス化することで開発者の負担を軽減することに成功し，メインフレーム時代のシステム開発で長期にわたり利用され続けてきた．また AIS 1100 は OSI や SNA などのプロトコルをサポートする通信制御機能をオプションで提供し，異機種間接続のアプリケーションを容易に構築することを可能とした．

1980 年代後半の金融業界第 3 次オンライン化などに伴い，サービスの多様化やシステムの機能拡張に対応するため，日本ユニシスは XTPA[*1] というアーキテクチャを用いてホストコンピュータの並列処理による無停止運転システムを実現した．AIS 1100 を機能拡張した XIS は，このような無停止運転システムを支援するためのミドルウェアとして登場し，金融機関向け勘定系システム TRITON[*2] を始めとしたミッションクリティカル業務を支えてきた．

2.2 メインフレーム時代のミドルウェアの位置付け

ミドルウェアの広義の定義は，「OS とアプリケーションの中間に位置し，OS やハードウェ

アの相違を吸収してアプリケーションへの影響を回避するためのソフトウェア」である．一般的には TP モニタやデータベース管理システムはミドルウェアに位置付けられる．

これに対して狭義の定義は，「OS，通信システム，データベース管理システムと言った基盤要素技術とアプリケーションの中間に位置し，基盤要素技術が異なる環境においてもアプリケーションの実装に影響を与えないことを目的とするソフトウェア」である．

日本ユニシスにおけるミドルウェアの概念は狭義の定義として捉えることが多い．メインフレーム時代のプロダクトは図 1 のように配置できる．コンピュータシステムの領域にはオペレーティングシステムの OS 1100 やトランザクション制御を行う TIP 1100 が対応する．通信システムの領域には端末や外部接続システムとの通信制御を行う CMS 2200 や MCB 1100 が対応する．データベース領域には UDS 1100 が対応し，ミドルウェアの領域には AIS 1100 や XIS が対応する．TRITON は業務システムの機能を提供すると同時に，システム基盤として XIS を包含しつつ業務運用を具現化した高いサービスレベルを満たすミドルウェアの機能も提供し，無停止運転システムを実現した．

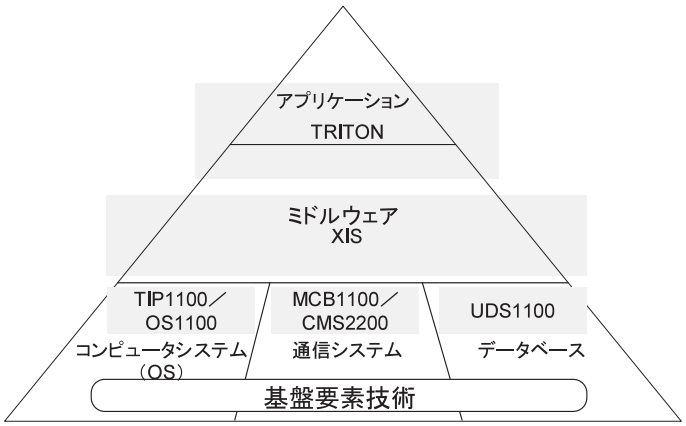


図 1　ミドルウェアの位置付けとメインフレーム時代のマッピング例

2.3　プラットフォームの変遷とミドルウェア

IT システムがメインフレームからオープンプラットフォームに変遷しても，ミドルウェアに求められる役割や目的は変わらない．しかし，ミドルウェアのあるべき姿は変わって行くことになる．

たとえば，オープンプラットフォーム用のミドルウェアを考えるとき，メインフレーム時代の経験をもとに，AIS 1100 や XIS の機能をそのまま実現することは効果的ではない．アプリケーション開発を取り巻く環境が全く異なるため，機能ベースの比較に意味をもたないためである．AIS 1100 や XIS が提供している機能の中で，オープンプラットフォームでの開発の傾向として不要になると考えられるものを表 1 に挙げた．

これ以外にもメインフレーム時代の開発はプロセス中心アプローチでバッチ処理が多く，中間ファイルが山積する傾向にある．これらバッチ処理をよく分析してみると，近年のプラットフォーム上では，SQL 文 1 つの実行で解決できるものや，データベースの設計を見直すことでバッチ処理が不要となるものもある．

一方，セキュリティ対策という点では，オープンプラットフォームのほうが重要性を増して

表 1　AIS 1100/XIS の機能と最近のオープンプラットフォームでの傾向

	AIS1100／XIS の機能	オープンプラットフォームの傾向
クライアント管理	端末単位で管理．事前に実行環境へ登録が必要．	ユーザ単位の管理．物理的な PC を意識しない．
アプリケーションの起動方法	端末機種毎に異なるルール．スケジューリング機能を提供．	クライアント機種に依存しない．
データアクセス	簡易な問い合わせのみ可能な UDS1100 のラッピング機能を提供．	SQL 文を直接実行できる．豊富な GUI で実装が簡単．
実行環境マルチ化	１台のホストコンピュータ内に複数の論理的な実行環境を構築可能．	安価なサーバを購入して複数の環境を構築する．

いる．特にインターネット向け Web アプリケーションの場合は，常に外部からの攻撃に注意しておく必要があり，基盤要素技術のセキュリティ脆弱性に関するパッチが提供された場合は速やかに適用することになる．アプリケーションレベルでも，予防のためのセキュリティ確保機能，監視のための操作記録採取や分析機能が必要になる．

日本ユニシスは，オープンプラットフォームにおけるミドルウェアを開発するにあたり，対象とするプラットフォームのテクノロジや開発環境，各ソリューション，開発スタイルを考え，メインフレーム時代のミドルウェアが提供していた機能をそのまま提供するのではなく，それぞれの機能の必要性について検討してきた．

3.　ミドルウェアの役割と .NET のプラットフォームにおけるあるべき姿

本章では，ミドルウェアの役割を明確にした上で，テクノロジの変化が著しいオープンプラットフォーム，特に .NET のプラットフォームにおける，ミドルウェアのあるべき姿について考察する．

3.1　ミドルウェアの役割

メインフレーム時代から現在に至るまで，プラットフォームの違いはあってもミドルウェアとしての役割については以下のように捉えることが出来る．

・基盤要素技術の隠蔽（ソースコードからの分離）
・汎用的に利用できる共通機能の提供

3.1.1　基盤要素技術の隠蔽（ソースコードからの分離）

企業システムのように広範囲で複雑な業務の場合，開発者が担当範囲の業務知識だけで実装できる状況が好ましい．業務に関する事柄以外はソースコードから分離させるという意味での隠蔽は従来の開発でも存在する技術である．ハードウェアの物理配置を隠蔽する COM＋サービスのサロゲート機能や DB サーバの接続文字列をソースコードから分離できるコンストラクタ文字列は隠蔽技術と捉えることができる．ミドルウェアとして提供される共通機能も，機能とインタフェースのみを開発者に周知させることで，ソースコード上に業務と無関係な記述をさせないことを目的としている．つまり，ハードウェアや基盤要素技術が変わってもアプリケ

ーションに影響を与えないことが理想的である．しかし，オープンプラットフォームのテクノロジは数年単位で激変しており，ミドルウェアが基盤要素技術を隠蔽するだけでは，アプリケーションへの影響を回避することは難しい．より大きなテクノロジの変化（進化）が，ミドルウェアが隠蔽している機能を概念から変えてしまうことがあるからである．

つまり，近年のオープンプラットフォームにおいては，ミドルウェアによる基盤要素技術の隠蔽は生産性向上や品質均一化に有効であるものの，基盤要素技術の変化に伴うアプリケーションへの影響は完全には吸収できないと考えるべきである．

3.1.2　汎用的に利用できる共通機能の提供

特定の業種や業務を前提としたドメインフレームワークは，機能要件と非機能要件の両方を満たすための機能が混在しているために，汎用性が低く適用範囲が限られてしまう．これに対して非機能要件を満たすためのプログラムは，共通機能を部品化すれば業種や業務に依存することなく汎用的に使用できるため，生産性向上や品質均一化に有効である．ミドルウェアを使わない開発プロジェクトにおいても，非機能要件を満たすことを目的として，認証機能，セッション管理，ログ採取，画面遷移，メッセージ管理などの共通部品を，各アプリケーションの開発者が個別に作成してきた．また，個別に作成したものを汎用的に使用できるよう部品化し，再利用することで生産性向上を図り一定の成果を上げてきた．図 2 は，Windows のプラットフォームにおける共通部品の位置づけについて示している．

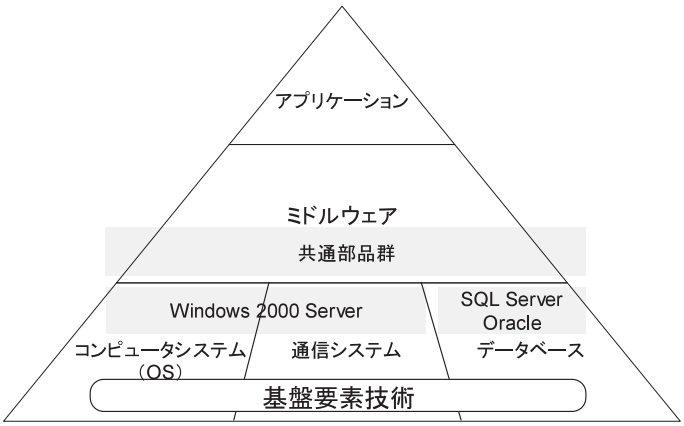


図 2　WindowsDNA 時代のマッピング例

通常のアプリケーション開発の場合はそれで問題ないが，ミッションクリティカルな業務システムの開発では加えて高いサービスレベル（高信頼性・高可用性・高管理性）という非機能要件が発生する．例えば，予想以上に処理要求が集中した場合においてもシステム停止を引き起こさないための流量制御や優先度制御が必要である．また，アプリケーションレベルでの障害復旧や障害の原因追跡，およびアプリケーションの稼働状況のモニタリングなども必要になってくる．そして，これらの非機能要件を満たすためのプログラムは，単なる共通部品の域を超え，一貫性のあるコンセプトによって体系的に整理され，システムのミドル（中間）に位置する信頼性および品質の高いソフトウェアとして提供されるべきである．

3.2 .NET プラットフォームでの開発

アプリケーション開発における共通機能の部品化によるメリットは実証されてきたが, .NET プラットフォームでのアプリケーション開発においては必ずしも共通部品を作成することが効果的ではない.

.NET Framework は .NET プラットフォームで実行される全てのプログラムの実行環境および豊富な共通ライブラリを提供しており，共通ライブラリを極力そのまま利用することが最もシンプルかつ効率的な開発であると考えることができる．つまり .NET Framework 自体が基盤要素技術を隠蔽し，アプリケーション開発に必要となる共通機能を提供するミドルウェアだと解釈することができる（図 3）．さらに Visual Studio .NET は，システム開発全般にわたり開発者に対して統合的かつ一貫した開発環境をもたらし，開発者は .NET Framework の共通ライブラリを効果的に利用することができる．また GUI 操作によるソースコードの自動生成機能を提供し，技術的なしきいを下げることで，開発者に負担をかけない優れた開発環境を提供している.

しかしながらミッションクリティカル業務の開発においては, 基盤要素技術や .NET Framework の機能だけでは満たせない高いサービスレベル（高信頼性・高可用性・高管理性）が要求されるため，本章で述べたようなミドルウェアが必要になる.

日本ユニシスは，これまでに培ったミドルウェアのノウハウを集約し，.NET プラットフォーム上でミッションクリティカル業務を構築するためのミドルウェアとして MIDMOST® for .NET を開発した.

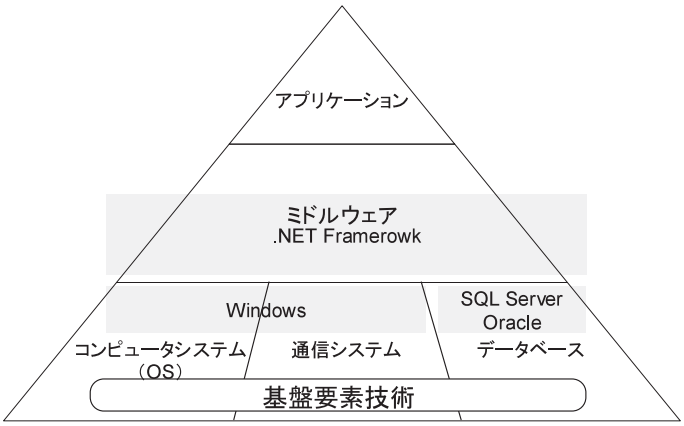


図 3 .NET プラットフォームでのマッピング例

4. MIDMOST® for .NET

MIDMOST® for .NET はミドルウェアであるが，基盤要素技術の違いを吸収する役割は担っていない（図 4）．本章では，MIDMOST® for .NET のミドルウェアとしての役割，開発の背景と必然性，そこから生まれてきた製品のコンセプトについて紹介する.

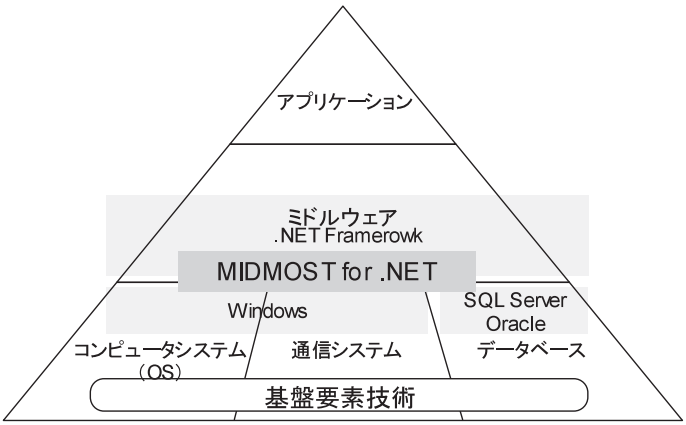


図 4　MIDMOST® for .NET のマッピング

4.1　MIDMOST® for .NET 開発の背景と製品のコンセプト

日本ユニシスでは 2002 年頃から .NET のテクノロジや製品を使った企業システムの開発業務を数多く手がけてきた．実際に .NET でのシステム開発プロジェクトにおいて，「.NET Framework や，開発ツールである Visual Studio を使うことで開発の生産性が向上すること」また「.NET を前提にした開発方法である LUCINA® for .NET を適用することにより，より品質の高いシステムの開発が可能であること」を実証し，ノウハウとして蓄積してきた．

しかしながら，これらの開発の経験を通して，.NET での開発プロジェクトで共通的に対峙する悩み・課題が浮き彫りになっていた．具体的には以下ような項目がある．

- ミッションクリティカルな業務システムを .NET ベースで開発する際には，.NET Framework やマイクロソフト標準の基盤要素技術（Windows オペレーティングシステム，SQL Server など）だけでは不足する共通機能（高い可用性，信頼性，運用容易性などの非機能要件を満たすための機能群）の開発が必要となる．この作業には非常に高い技術力と大きなコストが必要である．他社 ISV 製品を使うことで解決できる部分もあるが，この場合には製品の検証やその製品を使ったプログラムのプロトタイピングなどの作業が必要となる．
- 開発した共通機能を利用するための作業（学習・コーディング作業・テスト作業）の負荷が大きいと，設計・開発要員の作業誤りや情報伝達不足などの理由により，開発するプログラムにおける共通機能の利用方式・形態，適用範囲などの取り決めごとがプログラムの設計作業や実装作業の中で正しく適用されず，結果としてプログラムの品質が低下することがある．また共通機能を利用する際の制約[*3]が大きい場合には，システム開発の生産性を高めるための業界標準の技術（既存のアプリケーションアーキテクチャパターン，開発ツールなど）が利用できないなどの理由によりシステムの開発生産性が低下する．
- 大規模システムの場合，システム開発の期間が長くなるため，その開発期間中に非機能要件（効率性やセキュリティ，運用容易性など）の変更が発生するケースが多い．そのような変化に対応するためには，開発した共通機能の仕様の変更と，共通機能を利用するプログラム群の変更，それに伴うテスト・検証の作業が必要となる．この作業にかかる労力・コストはしばしば膨大なものとなり，また開発スケジュールの遅延の原因になる場合もある．

上記のような課題への対処はプロジェクトにとって非常に大きな負担となっていた．そこで2003年末，.NETベースでのシステム開発において前述の問題点を軽減・解消し，生産性が高く質の高いシステム構築サービスを提供するためのミドルウェアの開発・提供を行うに至った．

上記の課題に対する解決策は，「業務ロジックへの透過性」と「既存テクノロジとの親和性と補完性」という2つのコンセプトに沿ったミドルウェア機能[*4]の提供である．当コンセプトに沿った機能の提供により，具体的にどのようなメリットがあるかについては次章で説明する．

5. MIDMOST® for. NETが提供する機能のコンセプトとそこから生まれるメリット

4章で説明した通り，オープン時代のシステム構築サービスに求められるより高い生産性と，構築するシステムに求められる高い柔軟性とを実現するために，「業務ロジックへの透過性」と「業界標準テクノロジとの親和性と補完性」という，2つのコンセプトに沿った機能群を開発した．本章ではこの2つのコンセプトと，コンセプトに沿った機能の提供によりエンドユーザが受けるメリットを，具体的なMIDMOST® for .NETの機能の紹介を通して説明する．

5.1 コンセプト1：業務ロジックへの透過性

予測を超える業務処理量の増加，法律改正による遵守するべきセキュリティ要件の変化など，近年のミッションクリティカルシステムでは，世の中の突然かつ想定外の変化に対応できる柔軟性や俊敏性そのものが，新たな非機能要件となっている．

.NETを利用した開発により，システムの柔軟性や俊敏性を高くすることができた．加えて，MIDMOST® for .NETが提供する共通機能についても業務ロジックへの透過性を提供することができれば，前述のような想定外の要件の変化に対して，開発者や運用管理者はより小さいコストでより柔軟に対処することが可能となる．

ここで言う業務ロジックへの透過性とは，業務プログラムをアスペクト指向プログラミングで開発することによりミドルウェアを利用できる形態・特性を指す．

アスペクト指向プログラミング（以降AOPと略称する）により，ミドルウェアを利用する業務プログラムは，ビジネスロジックを記述する部分のプログラム（ビジネスロジック部と呼ぶこととする）と，非業務要件を意識した処理，つまりミドルウェアの呼び出し箇所や呼出しパラメータを記述するプログラム（ポイントカット部と呼ぶこととする）とが分離される．

ロギング処理など，システムの広範囲から同一の共通機能を呼び出すケースでは，一般的にAOPを利用すると，AOPでない場合に比べて，プログラムコードの量が減る．また，共通機能を呼び出す箇所の変更や，呼び出しに指定するパラメータの変更などの変更要求に対する更新作業の量も少なくなる（図5参照）．

また，ポイントカット部のプログラムを動的に変更することが可能となれば，変化に迅速に対応することが可能となる．

具体的には，MIDMOST® for .NETのログ機能であれば，出力する情報の種類やタイミング（ログ情報の出力のオン・オフ，プログラムのどこで出力するか）や出力形態（出力データ形式や圧縮方式・保存場所）などのポイントカット部分を業務プログラムとは分離してプログラミングできる．また，このポイントカット部分を，システム運行中に動的に変更できる．MIDMOST® for .NETで提供している機能の多く[*5]はアプリケーションプログラムからこのアス

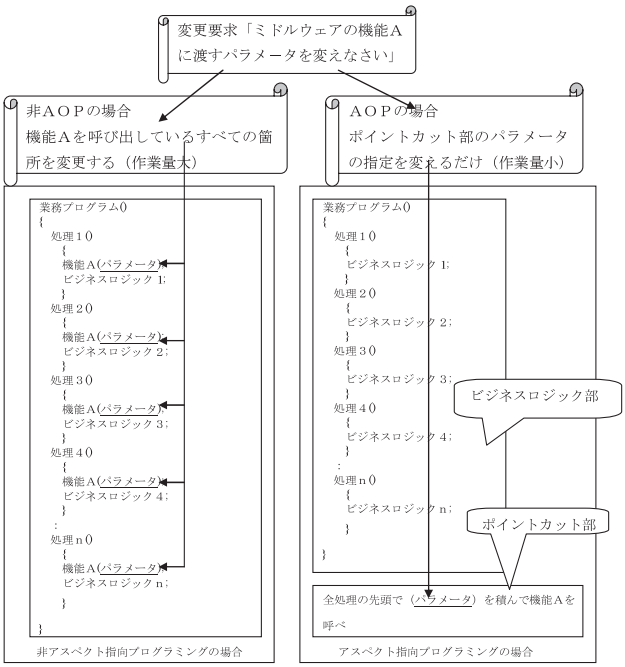


図5　非 AOP と AOP との変更要求に対する作業量の比較

ペクト指向プログラミングで呼び出しできる．

加えて，この透過性によってシステム開発者は開発時に，ビジネスロジックの開発に専念できるため，システムの品質や保守性の向上，開発期間の短縮の効果も期待できる．

MIDMOST® for .NET がどのようなテクノロジを利用して透過性を提供しているか，また透過性を利用できる各種機能（ログ，閉塞など）の詳細については，技報 85 号「ミッションクリティカルシステム構築・運用支援ミドルウェア MIDMOST® for .NET」[1]を参照していただきたい．

5.2　コンセプト 2：業界標準テクノロジとの親和性と補完性

業界標準テクノロジとの親和性とは，MIDMOST® for .NET を利用したシステムやアプリケーションの開発作業の中で，業界標準で最新の基盤技術を制約なく利用・適用できる特性をさしている．MIDMOST® for .NET は，業務プログラムがアプリケーションサーバとして IIS*6 を，トランザクション制御機能として MS-DTC*7 を利用することを前提に，IIS や MS-DTC などのマイクロソフト社が提供する標準的な機能に不足している機能やサービスレベルを補完する機能を提供する．「業界標準テクノロジとの補完性」とはこのような特性を指している．

一般的なミドルウェアの機能は，対象とする開発・実行プラットフォームの既存製品やテクノロジの機能を代替する形態で提供されるケースが多い．そのような形態だと，ミドルウェアを利用するプログラムを開発するために，特殊なツールや開発方法，アーキテクチャ，その他諸々の制約を強いることがある．基本的に，この制約を受け入れるためには何らかのオーバーヘッド，つまり生産性の低下が発生する．

開発するシステムの要件・形態によって，ミドルウェアを利用するための制約に対するオーバーヘッドの量は異なる．しかしながらこのオーバーヘッドが大きくなると，ミドルウェアの

共通機能を利用することにより得られるはずの生産性向上のメリットを減らしてしまうことになる．

これに対して，MIDMOST® for .NET では，ミドルウェアに依存した，アドホックな開発環境や実行環境という制約はなく，ミドルウェアの機能を利用するプログラマに行うべき教育のための追加コストも少ない．開発ツールや運用監視ツールも業界で広く普及しているもの[*8]を利用できるため，プログラムの開発作業に強いる制約も非常に小さい．また，開発方法・開発プロセスは LUCINA® for .NET を適用することが可能であり，親和性も高い．

以下では，この親和性と補完性が具体的に MIDMOST® for .NET の機能としてどのように提供されているかについて解説する．

5.2.1 アプリケーションサーバ（IIS）との親和性と補完性

MIDMOST® for .NET は IIS の機能はそのままに，IIS 上で稼働する業務アプリケーションの信頼性や可用性，運用性を向上させるための補完機能として主に以下の 3 つの機能を提供している[*9]．

1） 流量制御

業務処理の要求量が想定した範囲を超えて増加するような局面において，アプリケーションサーバやデータベースサーバのリソースの枯渇によるシステム障害を防ぐための機能である．

業務アプリケーションは，IIS の仮想ディレクトリ毎に同時受付可能な要求数の上限値を定義できる（この上限値を超えた要求を IIS にて受け付けた場合，要求元に対してエラーを返す）．

実際に当機能を使う場面では，ある仮想ディレクトリに配置したプログラムが実行されることにより消費されるリソース量（AP サーバや DB サーバの消費 CPU 時間，メモリ量）を見積もり，それを同時にどれだけの数実行させることを許容するかを決めて流量値を算出するというサイジングの作業が必要である．同時実行数の上限値はシステム稼働中に動的に変更することが可能である．

2） 実行優先度制御

業務処理の種類に応じて実行の優先度をつけることができる機能である．サーバの負荷が高い局面において，緊急性が高くより優先的に実行させたい処理がある場合に有効な機能である．

ユーザは，IIS 上の特定のアプリケーションプール[*10]に対して処理の優先度を 5 段階（Below, Below Normal, Normal, Above Normal, Above）で設定できる．

MIDMOST® for .NET の実行優先度制御は，このユーザが指定した優先度の値を，指定したアプリケーションプールに対応する IIS の実行プロセス（w 3 wp.exe）の CPU 実行優先度の値に割り当てる．結果として，より優先度の高い処理が優先的に CPU に割り当てられる．しかしながら，より優先度の高い処理であっても，I/O 待ちや Sleep などで待ちになっている状況においては，CPU に割り当て可能なより低い優先度のプログラムの処理が実行されることになる．

あるアプリケーションプールに対する実行優先度の値は，業務稼働中に動的に変更できる．優先度を変更した時点で実行中・仕掛り中の処理は強制的に停止され，アプリケーシ

ョンプールの再起動が行われる．

3）　AP 配布

業務アプリケーションの改修に伴うファイル群のアップデートを行う場合には，メンテナンス時間と称してシステム・業務全体を停止させることが多い．連続稼働が求められる業務においては，このメンテナンス時間をより短く，ゼロに近づけることが望まれる．

IIS の環境では，業務プログラムの関連ファイルの更新を処理実行中に行うことが可能だが，実際には IIS の停止により業務の全面停止を行い，ファイルの更新を行う．これは，ファイル群の更新作業中に受け付けてしまった要求に対する処理の整合性を保つことができないためである．

また，拡張性・可用性の確保のために AP サーバを複数台数束ねてサーバファームの構成で運用する場合，関連ファイル群をすべての AP サーバに対して誤りなく配置（デプロイ）するためには非常に手間がかかり，誤操作などのリスクも大きくなる．

MIDMOST® for .NET では Web/AP サーバの IIS 上で稼働している業務アプリケーションのプログラムファイルや構成ファイル類をサーバファーム単位で入れ替える機能として AP 配布機能を提供している．当機能を理解するために，当機能を利用した業務プログラムの更新業務の流れについて説明する．

① システムの運用管理者は，更新後の業務アプリケーションのファイル群（プログラムファイル・構成ファイル・データファイル）を配置準備場所（ステージャと呼ぶサーバマシン上の特定の IIS 仮想ディレクトリ）に置く．この操作のことをステージングと呼ぶ．この時点で，運用管理者はステージングされたアプリケーションを試験的に稼働させてみて，プログラムや業務の挙動に問題が発生しないかどうかの確認テストを行うことができる．

② システムの運用管理者は，MIDMOST® for .NET のパッケージ化コマンドを起動し，ステージングされたファイル群を 1 つのファイルにパッケージ化する．

③ システムの運用管理者が MIDMOST® for .NET の配布コマンドを実行すると，以下の④から⑥までの処理が自動的に実行される．

④ 配布対象のすべての Web/AP サーバの配布先のアプリケーションプールに対する処理要求が保留状態となる（IIS での http のリクエストの受付は行われるが処理は保留とされ実行されない）．この時点で当アプリケーションプール上で実行途中であった処理が存在する場合には処理が完了するまで待つ（一定の時間が経過しても完了しない場合には強制的に終了させる）．

⑤ パッケージ化された配布ファイル群を配布対象の Web/AP サーバ上に転送し，パッケージを解除し，配布先フォルダ上のファイルを上書き更新する．

⑥ ④の受付・保留を解除し，処理要求に対する処理を続行させる．

5.2.2　MS-DTC との親和性と補完性

.NET ベースでのトランザクション処理の業務プログラムを開発する場合には，プログラムの生産性・保守性などの観点から一般的に自動トランザクション方式[*11] を利用する．しかしながら，自動トランザクション方式を利用する際にはいくつかの課題があった．

1）　プログラムの実行環境の構築・維持のコストが非常に大きい（トランザクションプログ

ラムのライブラリに厳密名をつけた上で COM＋のカタログに登録する必要がある．バージョンの不一致などがあった場合の障害解析には非常に手間と時間がかかる）．

2） トランザクションの処理に関する，より詳細な情報（たとえば，どの PC から，誰が要求した処理に対するトランザクションなのか，トランザクションを処理したスレッドの番号はなにか，トランザクション完了までにかかった時間はどのくらいか？など）を見ることができない．アプリケーションの不具合などにより障害が発生した際の解析や，セキュリティ監査やパフォーマンス監視の観点での証跡の情報としては不充分である

MIDMOST® for .NET では自動トランザクション方式と同じプログラミングモデルを採用しながらも，上記の欠点を補う機能を提供している．MIDMOST® for .NET を利用するトランザクション処理と MIDMOST® for .NET を利用しないトランザクション処理の連結・連携も可能である．また，MIDMOST® for .NET の機能を使ったトランザクション処理も内部的には MS-DTC を利用しているため，トランザクションの実行状況を閲覧するための管理ツールも，マイクロソフト社が提供しているものと MIDMOST® for .NET で提供しているものとの同時利用が可能である．MIDMOST® for .NET が提供しているトランザクション機能のメリットと強化される機能をまとめると，

1） トランザクション処理プログラム（COM コンポーネント）の厳密名付与および COM への登録作業の軽減

MIDMOST® for .NET 用にビルドされたトランザクション処理プログラムファイル（DLL ファイル）を実行用 AP サーバにコピーするだけでよい．

2） RDBMS のデッドロックを自動的に回避することが可能（デッドロックリトライ機能）

トランザクション処理中に SQL Server や Oracle などの RDBMS からデッドロック状態が発生したことを知らされた場合に，トランザクションを一旦ロールバックし，一定時間が経過した後自動的に再実行させることによってデッドロック状態を回避することができる[*12]．

3） より詳細なトランザクションのログ情報を採取可能（トランザクションログ機能）

トランザクションの要求元の端末名やアカウントを特定するための識別子，トランザクションの処理受付・完了時刻，トランザクションの実行ステータス（コミット・アボート・ロールバック・ロールフォワード），実行時のスレッドの識別子（スレッド ID）などの情報をログとして出力できる．

4） 仕掛中のトランザクションの強制停止が可能（トランザクション強制停止機能）

非常に長い時間がかかるトランザクション処理を実行している最中に，その処理を強制的に停止させることができる．これは，プログラムの不具合による無限ループなどの障害が発生した場合や，長時間かかるトランザクションを開始した後に運用上のイベントが発生したために処理を停止させたいなどの局面で有効である．

5.2.3 運用管理ツール類との親和性

MIDMOST® for .NET は，業界標準の運用管理ツールや製品と連携して利用される局面を意識し，管理ツールとの親和性についても考慮している．例えば，「トランザクションログ機能」「入出力ログ機能」「AP ログトレース機能」のログ出力フォーマットは標準の XML フォーマット以外にも TSV[*13] フォーマットでの出力をサポートする．これにより，MIDMOST® for

.NET のログ出力情報を JP 1 などの業界で広く使われている運用管理ツール製品の統合ログコンソール上から閲覧することが容易に行える（JP 1[*14] では XML フォーマットのログをサポートしていない）．

また MIDMOST® for .NET は，あらかじめ定義した論理的な業務単位[*15] 毎に，システム運用に役立つデータをパフォーマンスカウンタの形で出力できる．したがって，Windows 標準の運用管理ツールである「システムモニタ」やパフォーマンスカウンタ値を閲覧する ISV 製の GUI ツールで閲覧したり，運用管理ジョブの起動判定用のデータとして活用することが可能となっている．表 2 に MIDMOST® for .NET が提供しているパフォーマンスカウンタ値を示す．

表 2　MIDMOST® for .NET が提供しているパフォーマンスカウンタ

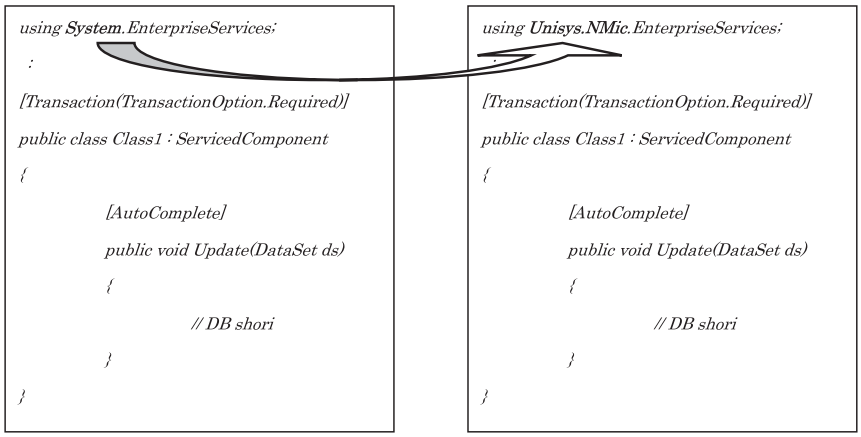
パフォーマンスカウンタの名称	説明
エラー発生回数（累計）	報告されたエラーの累計
エラー発生回数（チェックポイント間隔）	一定時間の中で報告されたエラー回数
アクティブ・サービスアクション数	現時点で実行中の要求の数.
アクティブ・サービスアクション数（最大）	同時実行数の最大値
サービスアクション[*16]処理総数（累計）	処理した要求数の累計
サービスアクション処理総数（毎秒）	秒あたりに受け付けた要求数（平均値）
サービスアクション処理成功数（累計）	処理結果が成功した要求数の累計
サービスアクション処理成功数（毎秒）	秒あたりに処理結果が成功した要求数（平均値）
サービスアクション処理失敗数（累計）	処理結果が失敗した要求数の累計
サービスアクション処理失敗数（毎秒）	秒あたりに処理結果が失敗した要求数（平均値）
サービスアクション応答時間（最大）	要求に対して処理にかかった時間の最大値（ms）
サービスアクション応答時間（最小）	要求に対して処理にかかった時間の最小値（ms）
サービスアクション応答時間（累計）	要求に対して処理にかかった時間の累計値（ms）
サービスアクション応答時間（平均）	要求に対して処理にかかった時間の平均値（ms）
アクティブ・トランザクション数	現在実行中の要求の数
アクティブ・トランザクション数（最大）	処理実行中の要求数の最大値
トランザクション数（累計）	処理したトランザクション数の累計値
トランザクション数（毎秒）	秒あたりに処理したトランザクション数の平均
トランザクション・ロールバック数（累計）	ロールバックしたトランザクションの累計数
トランザクション・ロールバック数（毎秒）	秒あたりにロールバックしたトランザクションの平均値
トランザクション・コミット数（累計）	コミットしたトランザクション数の累計
トランザクション・コミット数（毎秒）	秒あたりにコミットしたトランザクション数の平均値
トランザクション・デッドロック数（累計）	デッドロックを検出した回数の累計
トランザクション・デッドロック数（毎秒）	秒あたりに検出したデッドロックの回数の平均値
トランザクション応答時間（最大）	トランザクション処理にかかった時間の最大値(ms)
トランザクション応答時間（最小）	トランザクション処理にかかった時間の最小値(ms)
トランザクション応答時間（累計）	トランザクション処理にかかった時間の累計時間(ms)
トランザクション応答時間（平均）	秒あたりにトランザクション処理にかかった時間の平均値(ms)

5.2.4　LUCINA® for .NET との親和性

MIDMOST® for .NET は，LUCINA® for .NET をつかったシステム構築作業の中で利用しやすいような考慮をしている．実際に MIDMOST® for .NET が提供する機能を LUCINA® Web Foundation for .NET（以降 LWF と略称する）の業務サンプルプログラムの中に織り込む作業は非常に簡単である．

LWF で採用している自動トランザクションのプログラミングモデルを使ったアプリケーションプログラムの中で，MIDMOST® for .NET のトランザクション機能や入出力ログ機能を使うように変更したければ，図 6 のように自動トランザクション方式によるトランザクションを定義している部分（ServicedComponent クラスを継承したクラス）の基底クラスの名前空

間を“System.EnterpriseServices”から“Unisys.NMic.EnterpriseServices”に変更するだけでよい．



―MIDMOST for .NET を利用しないプログラム―　　―MIDMOST for .NET の機能を利用するプログラム―

図 6　自動トランザクション方式のプログラムへの MIDMOST® for .NET 機能の適用例

5.2.5　その他の Windows 関連テクノロジとの親和性

MIDMOST® for .NET は以下のような Windows の実行基盤の構成に対する親和性を持っている．

- ハードウェアによるネットワーク負荷分散処理装置や，マイクロソフト社が提供している Network Load Balancing の環境においても MIDMOST® for .NET の機能がサーバファーム単位で利用できるようにするために，Web/AP サーバファーム環境での単一なアプリケーションの管理環境およびツール類を提供している．
- Web/AP/DB サーバ群が同一の筐体の上で稼働する物理 1 階層の構成にも対応できる．また MSCS クラスタのマシン上に MIDMOST® for .NET のアプリケーションを稼働させることが可能である．

5.2.6　MIDMOST® for .NET が提供する機能

MIDMOST® for .NET が提供する 18 個の機能について，本章で紹介したものも含め，表 3 にまとめた．当製品の機能の選定にあたっては，開発プロジェクトの中から，業種・業態・業務内容に依存せず，かつ .NET の特にミッションクリティカルな案件にて汎用的にあてはまる要件を収集・選択した．また，2 章で解説したトランザクション業務向けのミドルウエア製品群（AIS 1100/XIS など）の機能で対応している要件も一部取り入れた．

6.　お　わ　り　に

MIDMOST® for .NET は Windows や .NET をベースとしたミッションクリティカル業務を支えるシステムに求められる信頼性・可用性・保守運用容易性のサービスレベルを満たすための共通機能を体系的に提供するミドルウェアである．

メインフレーム時代から，ミドルウェアが IT システムの構築の局面で果たしてきた役割は大きい．ミドルウェアは従来から，共通機能を提供することによる開発コストの軽減と，基盤

表 3　MIDMOST® for .NET の機能一覧

機能分類	機能名	機能概要
トランザクション制御	トランザクション処理	高速かつ管理コストの低いトランザクション処理環境を提供する
	デッドロック時リトライ	RDBMS のデッドロックを自動回避する
	トランザクションログ出力	監査のための，詳細なトランザクションの実行ログ採取
	トランザクション強制終了	ロングトランザクションに割り込み，手動で停止させる..
リクエスト制御	リクエスト流量制御	Web/AP/DB サーバの過負荷状態を防ぎシステムを保護する
	リクエスト優先度制御	ビジネスロジックに実行の優先度をつける
	要求差し止め	処理要求を一定時間保留として，短時間のサーバメンテナンスにおいても業務停止を防ぐ
	入出力ログ出力	業務ロジックの実行履歴の情報を自動的に採取する.
運行制御支援	実行状況監視	トランザクションの業務処理の実行状況を閲覧することができる.
	閉塞	全体停止せずに，業務のなかの一部を停止させる.
障害対応支援	AP ログ・トレース出力	業務エラーやトレース情報を出力する
	障害時自動アクション	業務ロジックで障害（業務障害）がしきい値の回数を超えたときに，事前に定義されたリカバリのためのロジック（アクション）を実行させることができる
	操作ログ出力	監査のために，MIDMOST for .NET に関する操作実績のログを採取する
配布更新支援	AP 配布	業務アプリケーションを業務稼働中に入れ替えることができる.
運用管理支援	各種管理ツール	トランザクション監視コンソール，サービスブロックマネージャなど
	統合運用コンソール連携	MIDMOST for .NET が出力するログ情報やパフォーマンスのデータが，Windows の標準的な管理ツール類で閲覧可能となるような考慮
開発支援	開発テンプレート	Visual Studio で MIDMOST for .NET のアプリケーションの初期コードを自動生成させるためのテンプレートファイル.
	開発用リファレンス	MIDMOST for .NET を利用した開発を行うために必要なリファレンス情報のドキュメント. ヘルプファイル形式で提供される

要素技術の違いの吸収という 2 つの役割があった．しかしながら，コンピューティング環境がオープンになるとともにミドルウェアに期待される役割や必要な特性も大きく変貌してきている．

近年の企業向けのオープンシステムでは，世の中の想定外の変化に柔軟にかつ迅速に対応できることが非常に重要になってきている．つまり，システムが高い俊敏性，高い柔軟性，高い保守性[*17] を持つことが求められるということである．故に，ミドルウェアについても，それを利用したシステムがこの性質（俊敏性，柔軟性，保守性）を維持できるような特性が求められる．

MIDMOST® for .NET は，この特性を持つミドルウェアであることを第一の目標として開発された．したがって，MIDMOST® for .NET は基盤要素技術の違いを吸収するということはあえて担わず，「業務ロジックへの透過性」つまり業務プログラムをアスペクト指向プログラミングすることによりミドルウェアの機能を利用できる特性と，「既存テクノロジとの親和

性と補完性」つまり業界標準のテクノロジを制約なく協調利用できるという特性，の2つを合わせもつことを目指した．これにより，同製品を利用することで .NET プラットフォーム上のミッションクリティカルな業務システムの開発をより効率的に行うことが可能になり，また高品質なシステムの開発・運用が容易になると考えられる．

＊1　eXtended Transaction Processing Architecture：複数ホストコンピュータ間でのデータベース共有機能を提供することにより，信頼性，処理能力，可用性および拡張性に優れたシステムの構築を可能とする拡張処理アーキテクチャ

＊2　TRIune Taskforce for new ONline banking system：日本ユニシス/紀陽銀行/百五銀行にて共同開発した総合オンライン勘定系システム．集計レス・伝票レスによる営業店事務省力化，ホストコンピュータの並列処理による無停止システムの実現，24時間365日対応，自由化商品への迅速な対応を可能とする仕組みを持つ．

＊3　制約として，共通機能を利用するために特有のアプリケーションアーキテクチャを適用しなければいけない，特有の開発ツールを利用して開発しなければならないなどの例が考えられる．

＊4　MIDMOST for .NET が提供している機能に何が含まれているかについては「5. 2. 6項 MIDMOST for .NET が提供する機能」を参照のこと

＊5　トランザクション制御，トランザクションログ，入出力ログ，閉塞，障害時自動アクション

＊6　IIS：Internet Information Service：Windows オペレーティングシステムに同梱されているAP サーバの機能．Windows Server 2003 では IIS Version 6.0.

＊7　MS-DTC：Microsoft Distributed Transaction Coordinator, 2相コミットの機能を提供するWindows システムにおける標準的な TP モニタ．Windows オペレーティングシステムに同梱．

＊8　開発ツール：Microsoft Visual Studio .NET，運用監視ツール：日立 JP 1 や Windows オペレーティングシステムに同梱されているツール類など

＊9　IIS の補完機能はこの3つ以外にも存在する（IIS 設定変更ツールなど）が当稿では割愛している

＊10　IIS の中の実行環境単位．1つのアプリケーションプールは，システム上でひとつのプロセスに対応する．1つのアプリケーションプールに対して複数業務を割り当てることも，業務ごとにアプリケーションプールを個別に割り当てることもできる．効率性を求める場合にはより少ないアプリケーションプールに多数の業務を入れ，信頼性・安全性を求める場合にはアプリケーションプールを複数業務毎に用意する場合が多い．

＊11　自動トランザクション方式：LUCINA for .NET でも推奨している .NET の標準的なトランザクションプログラミング方式．.NET Framework で，System. EnterpriseServices 名前空間の ServicedComponent クラスを継承したクラスを作成し，Transaction カスタム属性を付与する．トランザクションの開始点はメソッドの入り口となる．AutoComplete 属性を付与すれば明示的にコミットやロールバックを書く必要はない［技報85号より引用］

＊12　トランザクション処理プログラムの設計上の不具合に起因したデッドロックを回避するための機能ではない．RDBMS の実装に起因し，ユーザが予測困難でかつ非常にまれに発生するデッドロック（SQL Server のロックエスカレーションなど）を回避するための機能である．

＊13　TSV：Tab Separated Values

＊14　JP 1 Version 7 i for Windows

＊15　MIDMOST® for .NET では業務単位の範囲をサービスブロックとして定義しておくことができる．

＊16　サービスアクション：MIDMOST® for .NET が提供するクラス（Unisys. Nmic. Enterprise-Service 名前空間の ServicedComponent）を継承したクラス内で実行される範囲をサービスアクションと呼ぶ．サービスアクションの中でトランザクションを生成するものとしないものがある．

＊17　高い俊敏性：要件が変更された場合に短時間に対応できること．高い柔軟性：カスタマイズなどによる対応が可能な要件の幅が大きいこと．高い保守性：運用コストが少ないこと，保守コストが少ないこと．

参考文献　［1］　溝上昌宏，ミッションクリティカルシステム構築・運用支援ミドルウェア「MIDMOST for .NET」，Unisys 技報，通巻85号，2005年5月

［2］　山岸重雄，.NET の本来の姿と現状，そして日本ユニシスの .NET，Unisys 技報，通巻85号，2005年5月

執筆者紹介　菅　谷　俊　郎（Toshiro Sugaya）

1989 年青山学院大学理工学部卒同年日本ユニシス(株)入社．UNIX オペレーティングシステムや Windows オペレーティングシステムのサポートに従事．現在は総合技術研究所 IT ソリューション部に所属し，Windows および .NET 上の大規模ミッションクリティカルシステム構築・運用業務用のミドルウェア製品である MIDMOST® for .NET や ACAB の開発およびサポートに従事．

執筆者紹介　白　木　和　彦（Kazuhiko Shiraki）

1991 年日本ユニシス(株)入社．　XIS や TRITON の開発を経て，Windows 上での金融機関向けシステム開発に従事．現在，日本ユニシス・ソリューション(株)共通技術開発本部コンピテンスセンタ .NET 基盤技術室に所属し，.NET システム構築支援に従事．