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」

Mission-critical System Implementation and Operation Support Middleware "MIDMOST for .NET"

溝 上 昌 宏

要 約 .NET Framework は，アプリケーションの開発生産性を向上させる技術として開発の現場に確実に浸透してきたが，ミッションクリティカルな基幹業務システムを構築するためには，まだ多くの機能が提供されていない．多くの現場では信頼性や可用性といった非業務要件を実現するために，機能を独自に開発することを余儀なくされている．

こうした背景から，日本ユニシスは，.NET Framework ベースのミッションクリティカルシステムの構築・運用を支援するミドルウェア MIDMOST for .NET を開発した．MIDMOST for .NET のコンセプトは，横断的関心事の分離，.NET 標準開発スタイルとの親和性確保，そして，既存テクノロジーとの融合である．本稿では，これらのコンセプトを軸に，.NET Framework だけでは解決されない課題を MIDMOST for .NET が如何にして解決するのかを論じる．また，コンセプトを具現化するアーキテクチャの特徴を紹介する．

Abstract .NET Framework has surely spread into the development site as the technology that improves the productivity of the application development. However, to implement a mission-critical system, a lot of functions have not been provided. To achieve non-business requirement such as the reliability and availability, etc., many site are forced to develop this kinds of functions independently.

Nihon Unisys has developed the middleware MIDMOST for .NET that supports implementation and operation of a mission-critical system based on .NET Framework from such a background. The concepts of MIDMOST for .NET are the separation of cross-cutting concerns, ensuring of compatibility with the .NET standard development style, and uniting with existing technologies. This paper discusses how MIDMOST for .NET solves the problems that cannot be solved only by .NET Framework, leading with these concepts. Moreover, it introduces the feature of architecture that makes the concept embody.

1. はじめに

.NET Framework は登場から 3 年が経過し，アプリケーションの開発生産性を向上させる技術として開発の現場に確実に浸透してきた．しかしながら，ミッションクリティカルな基幹業務システムを構築するためには，まだ多くの機能が提供されていない．多くの現場では信頼性や可用性といった非業務要件を実現するために，こうした機能を独自に開発することを余儀なくされている．

こうした背景から，日本ユニシス（以下，当社）は，.NET Framework ベースのミッションクリティカルシステムの構築・運用を支援するミドルウェア MIDMOST for .NET を開発し，2005 年 2 月に初期バージョンをリリースした．本稿では MIDMOST for .NET のコンセプトを中心に，MIDMOST for .NET の機能とアーキテクチャを紹介する．まず始めに，開発の背景について述べ，MIDMOST for .NET の概要を紹介する．次に，コンセプトについて述

べ，.NET Framework だけでは解決されない課題を MIDMOST for .NET が如何にして解決するのかを論じる．最後に，コンセプトを具現化するアーキテクチャの特徴を紹介する．

2. MIDMOST for .NET 開発の背景

ミッションクリティカルな基幹業務システムの構築といえば，メインフレームや UNIX サーバが主流の時代が長く続いてきたが，2000 年に登場した Windows 2000 Server や，ハードウェア技術の革新により，Windows 上でも安定した信頼性と高可用性を実現するシステムの構築が可能となった．当社において，そのエポックメイキングなシステムが，2001 年に本番稼働を開始した某都市銀行勘定系システムの対外接続システムであり，以来，当社の IA サーバ Unisys Enterprise Server ES 7000 とマイクロソフト社の Windows Server System 製品群の組み合わせによって基幹業務システム構築の実績を積み重ねてきた．これらの実績を支えているのが当社が開発・提供するオープンミドルウェア MIDMOST と HA システム構築支援ソフトウェア ACAB[*1] である．

一方，Windows 上でのアプリケーション開発という領域に大きな転換をもたらしたのが，2002 年にマイクロソフト社がリリースした.NET Framework である．.NET Framework とその開発環境である Visual Studio .NET は，アプリケーション開発の生産性を飛躍的に向上させる技術であるということができる．それらは，クライアントからデータベースに至るシステム全体の開発に対して，統合的かつ一貫した開発基盤を提供する．また，実行基盤である共通言語ランタイムはシステムに高い信頼性と安定性をもたらす．そうした一方で，.NET Framework は非常に高機能かつ拡張性の高い開発フレームワークを提供するため，設計や実装における自由度が高く，システムの品質や生産性にばらつきが生じやすいという側面も否めない．当社においては，こうした側面をカバーするため，.NET Framework を前提とする開発方法とテンプレートから構成される LUCINA for .NET[*2] を提供している．

Windows 上での業務システム開発は，従来の VB や C，C++を使用した Win 32 API ベースの開発から，.NET Framework ベースの開発へと確実に移行してきている．しかし，.NET Framework がいかに高機能であるとはいえ，基幹業務システムの構築に必要な機能という観点で見た場合，過去のプラットフォームが提供してきた機能に比べると，まだ多くの機能が不足している．一例をあげると，システムを稼働させたまま一時的にサービスを停止させる閉塞処理や障害時の原因追及に必要十分な情報を確実に採取し，解析する機能，システムの安定性を保証するために処理要求の流量を制御する機能などである．これらはミッションクリティカルシステムに求められる信頼性や可用性といった非業務要件を実現するために必要な機能であり，従来のメインフレームであれば当たり前のように提供されていた機能である．先に述べた当社のミドルウェア MIDMOST は，まさにこうした機能を Windows 上で実現するミドルウェアであるが，Win 32 API ベースの開発が基本であり，.NET Framework ベースの業務アプリケーション開発はサポートしていない．.NET Framework ベースでのシステム開発においては，業務処理の実装と同時に，業務処理以外のいわば本質的でない機能を独自に作りこまなければならず，それが開発者にとって大きな負担となっていたのである．

こうした背景から誕生したミドルウェアが MIDMOST for .NET である．MIDMOST for .NET の目的は，.NET Framework ベースの基幹業務システム構築において共通的に必要な機能を提供し，開発者を非業務機能の開発から解放することである．それによって，開発者を本

来の業務処理の開発に専念させ，開発期間の短縮，開発コストの削減，品質の向上を図る．MIDMOST for .NET のアーキテクチャ設計においてもっとも重要視した点は，.NET Framework が本来持っている高生産性をもたらす開発フレームワークとしての長所を活かし，独自な開発スタイルを持ち込まないことである．そのためには，MIDMOST for .NET が提供する様々な機能は.NET アプリケーションから自然な形で利用できることが求められる．

3. MIDMOST for .NET の全体概要

3.1 MIDMOST for .NET とは

MIDMOST for .NET は，ミッションクリティカルシステムに求められる要件である，高信頼性，高可用性，高管理性，高拡張性を実現するための 7 種類の機能を提供するミドルウェアである（図 1）．

これらの機能は，.NET Framework ベースのシステムの開発・実行・運用に際して，Windows プラットフォームや.NET Framework の機能では不足する部分を拡充・強化するものであり，これらを有機的に結合することによって，ミッションクリティカルシステムの開発・実行・運用を支援する一つのソリューションを形成する．

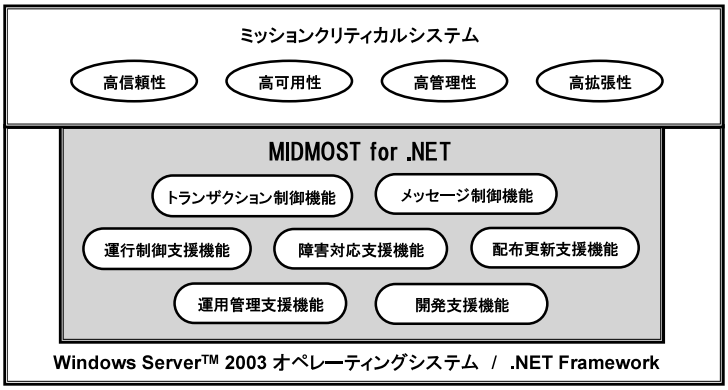

図 1 MIDMOST for .NET

3.2 MIDMOST for .NET の機能概要

次に，MIDMOST for .NET の各機能の概要を紹介する．

- トランザクション制御

  トランザクションを保証する制御機構は MS DTC（Microsoft Distributed Transaction Coordinator）を利用する．その上で，MS DTC では提供されていないトランザクションの発生履歴に関するログを採取する機能を提供し，障害追跡や効率分析等に活用可能とする．さらに，トランザクション障害の中でもユーザアプリケーションでの予測・対処が困難なリソース競合によるトランザクションのタイムアウトや滞留などの現象に対して，自動的なトランザクションのリトライや強制的なトランザクションの停止・ロールバックの機能を提供する．

- メッセージ制御

  クライアント/サーバ間の通信や業務コンポーネントに対する入出力の履歴を保存する機能を提供し，稼働状況分析や障害原因追及，セキュリティ監査等に活用可能とする．また，メッセージの紛失を可能な限り防ぎ，サーバ障害時に既に受け付けた処理要求や処理

に対する応答を保証する機能や，IIS との連係によってリクエストの流量制御や優先度制御を実現する機能を計画中である．

- 運行制御支援

  アプリケーションの実行状況をリアルタイムに監視する機能を提供する．また，障害検出時にプロセスのダンプ採取やアプリケーションプールのリサイクリングやシャットダウンなどのアクションを自動実行する機能を提供し，障害時の迅速なリカバリを可能とする．さらに，クライアントから受け付けた要求をサーバ内で保留または停止させて，クライアントに適切な応答を行う閉塞機能を提供し，オンライン処理の一部または全部を一定期間サービス停止させたり，短時間のデータベースのバックアップのために処理は受け付けたまま保留させたりすることを可能とする．

- 障害対応支援

  業務アプリケーションの実行履歴をアプリケーション固有のデータとともに保存する機能を提供する．また，MIDMOST for .NET によって出力された各種ログをデバッグや障害対応などの局面で利用しやすい形で閲覧・検索するツールを計画中である．

- 配布更新支援

  クライアントからの要求を差し止める機能を提供し，24 時間 365 日運転を続けるシステムにおいて業務アプリケーションの動的入れ替えを可能にする．また，JP 1 との連係によって各サーバの構成情報の書き換えを支援する機能を計画中である．

- 運用管理支援

  JP 1 等の ISV 製運用管理ツールによる統合運用管理を実現するための連係機能を計画中である．

- 開発支援

  Visual Studio .NET 用のテンプレートを提供し，MIDMOST for .NET を利用したプログラミングの生産性を高める．また，MIDMOST for .NET を活用した開発，運用のためのドキュメント類を提供する．

3.3 MIDMOST for .NET のシステム構成

MIDMOST for .NET のシステム構成の例を図 2 に示す．ここでは，Web サーバと AP サーバを同一筐体にした物理 2 階層の例を示している．Web サーバと AP サーバを分離して物理 3 階層とすることも可能である．MIDMOST for .NET の実行環境は Web/AP サーバに導入するのはもちろんのこと，クライアント/サーバ間を Web サービスまたは.NET Remoting で接続する場合は，クライアントにも導入できる．

4. MIDMOST for .NET のコンセプト

本章では，MIDMOST for .NET が機能を提供するにあたって，その設計の根底にある重要なコンセプトについて述べる．MIDMOST for .NET の設計指針ともいうべきコンセプトは次の 3 点にまとめられる．

- 横断的関心事の分離
- .NET 標準開発スタイルとの親和性確保
- 既存テクノロジーとの融合

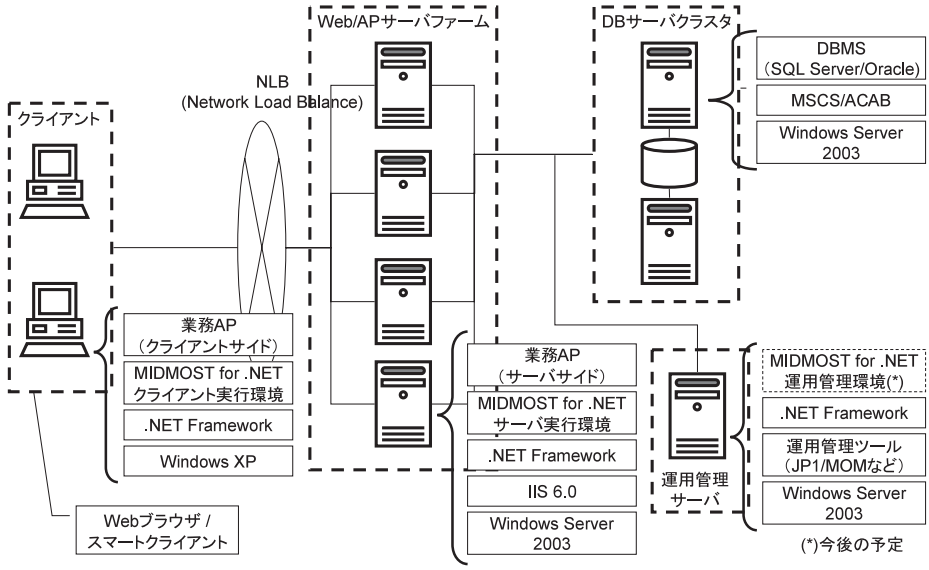

図 2　MIDMOST for .NET システム構成例

4.1　横断的関心事の分離

2 章で，MIDMOST for .NET が提供する様々な機能は.NET アプリケーションから自然な形で利用できることが求められると述べた．ここで自然な形とは，開発者が実装する業務プログラムから，MIDMOST for .NET が提供する機能，すなわち非業務機能の存在をできる限り意識させないということを意味する．

一般にソフトウェアはモジュールに分解して開発する．いくつかのモジュールで共通的に必要なコードは，別のモジュールにまとめることによって，同じようなコードを各所に実装することを避ける．それは，ソフトウェアのより本質的な流れの部分と，そこから外れる本質的でない部分とを分離することによって抽象度を高め，生産性や保守性を向上させるための努力である．

例えば，手続き的プログラミングでは，手続きと呼ばれる単位でモジュール分割が行われる．ただし，データという側面での分離は行われない．これに対して，オブジェクト指向プログラミングでは，データと手続きを一体化したオブジェクトの単位で分離され，より高いレベルでモジュール分割される．

しかしながら，こうした従来の枠組みでは，利用する側のモジュールが利用される側のモジュールを呼び出すというコードそのものはなくならない．本質的でない非業務機能を実装するコードは別モジュールとして分離できても，それを呼び出すコードは依然として各モジュールに散在してしまう（図 3）．アスペクト指向プログラミング（Aspect Oriented Programming）では，そうした各モジュールに横断的に散在するものを横断的関心事（cross-cutting concerns）と呼び，それさえもモジュール化して分離しようとする．

さて，やや議論が横に逸れたが，MIDMOST for .NET が提供する機能の存在をアプリケーションから意識させないためには，その機能の振る舞いだけではなく，その機能を呼び出すコード自体を業務プログラムから分離することが必要と考えた．幸いなことに，.NET Framework は，こうしたアスペクト指向の考え方を実践するための下地を持っていた．その一つが

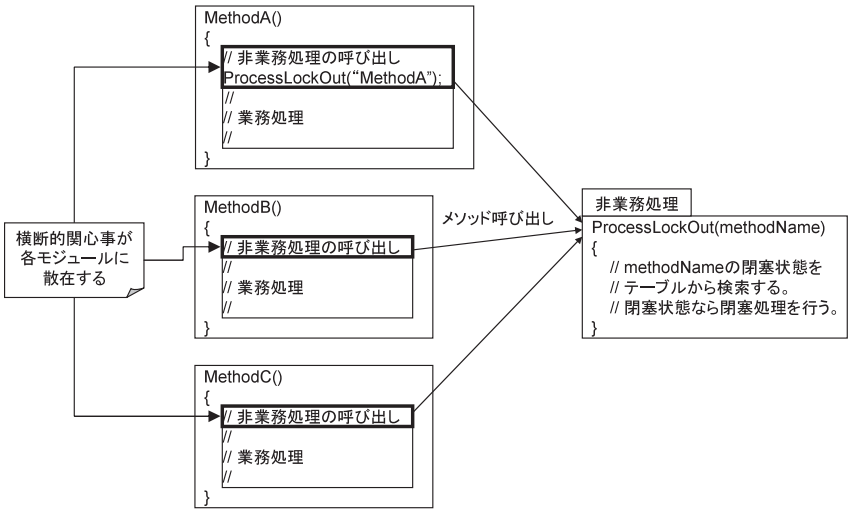

図 3　散在する横断的関心事

カスタム属性である．カスタム属性はメタデータに拡張性を与えるためのものであり，言語仕様には規定されていない情報をクラスやメソッドなどの言語要素に追加することを可能にする．

MIDMOST for .NET では，機能を業務プログラムに利用させるための“しるし”をカスタム属性という形で事前定義して提供する．アプリケーション開発者は，MIDMOST for .NET の機能を利用したいクラスやメソッドに対して，事前定義されたカスタム属性を付与するだけである．もはや，業務プログラムには MIDMOST for .NET の機能を呼び出すコードすら必要なく，開発者は本来の業務処理の実装だけに注力すればよい．MIDMOST for .NET の機能は業務プログラムから完全に分離され，開発者の知らないところで動作する．これが MIDMOST for .NET の目指すところである．

4.2　.NET 標準開発スタイルとの親和性確保

.NET Framework とその開発環境である Visual Studio .NET は，アプリケーションの生産性を飛躍的に向上させる技術として高い評価を得ている．.NET Framework プログラミングを学習するための参考書は巷に溢れている．また，アプリケーションアーキテクチャを設計する際にも，既にベストプラクティスと呼ばれる方法が存在する．マイクロソフトが出版している patterns & practices がその好例である．また，当社の LUCINA for .NET のような開発方法論もある．

しかし，ミドルウェアを利用するために，ミドルウェアが前提とする開発スタイルを強制されたり，特定のアプリケーションアーキテクチャを強制されたりするとしたら，開発者は新しい学習を余儀なくされ，これまで学習してきたものとのギャップの解消に神経をすり減らすこととなり，.NET Framework の魅力を損なうことにもなりかねない．

こうした点を踏まえ，MIDMOST for .NET では，開発者に対して独自の開発スタイルを強制せず，.NET Framework ならびに Visual Studio .NET による開発経験者であれば，違和感なく MIDMOST for .NET による開発を習得できることを目指した．データベースのアクセスは ADO.NET，サーバとの通信は Web サービスや.NET Remoting をそのまま利用可能とした．

また，サーバの配置に関しては，他社製品に多い Web サーバ/AP サーバ間の通信に.NET Remoting を用いた物理 3 階層に限定せず，より高いパフォーマンスを得られるように Web サーバ/AP サーバを同一筐体とする物理 2 階層にも対応し，アプリケーションアーキテクチャという観点でも標準的なアーキテクチャを採用できるようにした．

4.3 既存テクノロジーとの融合

.NET Framework ベースの基幹業務システム構築に必要な機能をソフトウェアとして打ち出していくには，.NET Framework を始めマイクロソフト製品群が提供する機能とその将来動向を考慮し，追随していくことが重要となる．また，マイクロソフト製品以外でも，Windows ベースのシステム構築・運用において業界標準といわれるソフトウェアが存在する．MIDMOST for .NET の提供機能を検討するにあたっては，.NET Framework を中心としたテクノロジーおよび実績のある製品群が提供する機能を基本とし，現状不足している機能を補完するというアプローチをとった．

アプリケーションサーバとしては Windows Server に標準装備される IIS の利用を前提とした．IIS は Windows Server 2003 からは IIS 6.0 となり，本格的にアプリケーションサーバとしての機能が強化され，信頼性，可用性，効率性において十分な性能を持つに至った．MIDMOST for .NET では，この IIS をアプリケーションサーバとして活用し，独自のアプリケーションサーバは提供しないこととした．

ミッションクリティカルシステムの要となるトランザクションの ACID 特性を実現するアーキテクチャは，.NET Framework ベースのシステム構築において実績があり，保証されているトランザクション処理モニタ（TP モニタ）およびデータベース管理システム（DBMS）を採用することとした．すなわち，TP モニタには Windows Server に標準装備されている MS DTC（Microsoft Distributed Transaction Coordinator）を採用し，DBMS にはマイクロソフト社の SQL Server あるいはオラクル社の Oracle を前提とした．

システム運用管理に関しては，日立製作所(株)の JP 1 や Microsoft Operations Manager（MOM）などとの連係を前提とした．

5. MIDMOST for .NET による課題解決

本章では，.NET Framework だけでは解決しない課題を MIDMOST for .NET が如何にして解決するのかを述べる．MIDMOST for .NET の提供機能は多岐にわたり，ミッションクリティカルシステムを構築・運用するためのソリューションを構成するが，残念ながら本稿ですべてを紹介することはできない．ここでは，主要な次の三つの機能に着目することにする．

- トランザクション管理
- 入出力ログ
- 閉塞

5.1 トランザクション管理

5.1.1 .NET Framework における現状と課題

.NET Framework で扱えるトランザクション方式は次の 3 種類がある．

- データベーストランザクション

ストアドプロシージャ内にSQL拡張言語を使用してトランザクション制御を埋め込んでおき，これを外部から呼び出す方式

- マニュアルトランザクション

  ADO.NETによって，データベース外部からトランザクションを制御する方式

- 自動トランザクション

  COM＋とMS DTCによって分散トランザクションを制御する方式

これらの方式のうち，自動トランザクションがプログラミングの容易性・生産性の観点において優れており，ほとんどのケースにおいて有効とされる．そのため，LUCINA for .NETでも自動トランザクションを推奨している．自動トランザクションを利用するには，COM＋コンポーネントを作成する必要があるが，.NET Frameworkでは，System.EnterpriseServices名前空間のServicedComponentクラスから継承したクラスを作成し，Transactionカスタム属性を付与するだけでよい．トランザクションの開始点はServicedComponentクラスのメソッドの入口であり，メソッドにAutoComplete属性を設定すれば，明示的にコミットやロールバックを書く必要がない．

このように，COM＋による自動トランザクションはプログラミングが容易だが，運用管理面では次のような課題が認識されている．

- 環境構築のコストが高い．

  作成したCOM＋コンポーネントを実行させるためには，アセンブリを厳密名で署名し，COM＋カタログ（レジストリ）への登録が必要となる．登録や設定などの作業が煩雑なため，開発環境の構築や運用環境への配置の際に，バージョンの不整合など登録に起因するトラブルが発生しやすく，解決に時間がかかることが多い．

- 稼働状況の把握や障害時の原因追求に利用できる情報が少ない．

  MS DTCのログは得られる情報が少なく，障害の原因追求には使用できない．また，トランザクションの稼働状況を把握するには，COM＋の管理画面で採取できる情報では不十分である．

5.1.2 MIDMOST for .NETのトランザクション管理機能

MIDMOST for .NETのトランザクション管理機能は，COM＋の自動トランザクションによるトランザクションプログラミングの容易性・生産性を維持しながら，前述の課題を解決する．

MIDMOST for .NETでの実装例を図4に示す．.NET Frameworkが提供していたクラスや

```
[C#]
using Unisys.NMic.EnterpriseServices;
    …
[Transaction(TransactionOption.Required)]
public class Class1 : ServicedComponent
{
    [AutoComplete]
    public void Update(DataSet ds)
    {
        // データベース更新処理
    }
}
```

図4　トランザクション機能を利用するクラスの実装例

属性は，MIDMOST for .NET でも名前空間のみが異なる同じ名前のクラスや属性として提供している．コーディングスタイルは変わらないため，開発コストは同じである．既存のソースコードを移行する場合でもわずかな修正で可能である．

.NET Framework の標準機能との大きな違いは，COM＋カタログへの登録が不要になったことである．通常のアセンブリと同じようにファイルのコピーで配置が完了するため，これまでの課題であった環境構築時の負荷がなくなる．トランザクションオプションの変更などCOM＋の管理画面で行っていた構成変更は，MIDMOST for .NET では構成ファイルで行えるようになっており，管理コストが少ない．

トランザクションログには，トランザクションの開始/コミット/ロールバックおよびデッドロックの検出時に次の情報が出力される．

- 発生日時
- 呼び出し ID
- 呼び出し元マシン名
- 呼び出し元ユーザ名
- ローカルマシン名
- アセンブリ名
- プロセス ID
- アプリケーションドメイン名
- スレッド ID
- クラス名
- メソッド名
- トランザクションの動作
- トランザクション識別子

これらの情報は後に述べる入出力ログやアプリケーショントレースログなど，他のログと付き合わせることで，詳細な分析が可能である．

また，ここでは詳しくは述べないが，MIDMOST for .NET の監視コンソールを使用して，トランザクションの稼働状況をリアルタイムに監視することが可能である．さらに，この監視コンソールから実行中のトランザクションを強制停止する機能も備えている．長時間かかるトランザクションを実行中だったが，メンテナンス等の理由により急遽トランザクションを中止する必要があっても，MS DTC では外部からトランザクションを取り消すことはできなかった．

5.2 入出力ログ

5.2.1 .NET Framework における現状と課題

稼働状況分析，障害原因追及，セキュリティ監査等を目的に業務システムに対する入出力メッセージをトレースする機会は多い．クライアント/サーバ双方の送受信記録をもとに，障害発生時にどこまでメッセージが到達していたのかを確認できる必要がある．

こうした要求を.NET Framework の標準機能だけで実現しようとすると，次のような現状や課題に直面することになる．

- 必要な情報を採取するために専用のロギングライブラリを独自に作成する必要がある．
- メソッドの入口や出口，例外検出箇所など，決まりきった箇所でも逐一そのライブラリを呼び出す処理を追加しなければならない．
- ロギング処理を呼び出す処理が複数個所に散在するためライブラリの仕様変更に弱い．
- 処理を追加すべき場所に追加し忘れたり，追加すべきでない場所に追加したりするなど，コーディングミスが発生しやすい．

5.2.2　MIDMOST for .NET の入出力ログ機能

前述のようなロギング処理は，業務システム開発において必要な処理ではあるが，本来開発者が注力しなければならない業務処理とは無関係な処理である．MIDMOST for .NET では，こうした定型的だが本質的ではないロギング処理を業務プログラムから完全に分離する．

入出力ログ機能の利用方法は，ローカルなメソッド呼び出しの場合とクライアント/サーバ間の通信の場合とで設定方法が異なるが，いずれの場合も業務プログラムから機能を呼び出すようなコードを記述する必要はない．

ローカルなメソッド呼び出しに対する入出力ログを採取するには，ロギングを行いたいクラスに ServicedComponent クラスを継承させ，TraceLog 属性を付与するだけである．この設定を行うだけで，当該クラスのメソッドが呼び出された際に，メソッドの入り口，出口および例外発生時に自動的にログを採取することができる．実装例を図 5 に示す．

```
[C#]
using Unisys.NMic.EnterpriseServices;
    …
[TraceLog(TraceLogLevel.Full)]
public class Class1 : ServicedComponent
{
    public void Update([DisableLog] DataSet ds)
    {
        // データベース更新処理
    }
}
```

図 5　入出力ログ機能を利用するクラスの実装例

ログの出力箇所を制御するログレベルは 3 種類があり，カスタム属性のプロパティで指定する．この設定は構成ファイルでも変更できる．ログに出力される内容は次のとおりである．なお，特定のメソッドやパラメータを出力対象から外したい場合は，図 5 に示すように DisableLog 属性を利用できる．

- 発生日時
- 呼び出し ID
- 呼び出し元マシン名
- 呼び出し元ユーザ名
- ローカルマシン名
- アセンブリ名
- プロセス ID
- アプリケーションドメイン名
- スレッド ID
- クラス名
- メソッド名
- 入力/出力/例外の区別
- パラメータの名前，型，値
- 戻り値の型，値
- 例外クラス名
- 例外詳細情報(メッセージ，スタックトレース等)

Web サービスまたは.NET　Remoting によるクライアント/サーバ間の通信において，クライアント/サーバの双方で入出力メッセージを採取できる．この場合は，ロギング対象となるクラスとログレベルの指定を構成ファイルで行う．プログラム自体は通常の Web サービスや.NET　Remoting のプログラミングを行うだけでよく，特別な考慮は必要としない．採取される情報はローカルなメソッド呼び出しの場合とほぼ同様だが，Web サービスの場合は SOAP メッセージの形式で出力される点が異なる．

MIDMOST for .NET のメッセージ制御機能の特徴として呼び出し元情報がある．呼び出し

元情報は，クライアントがサーバに要求を送信してから応答を受信するまでの一連の呼び出しの間，暗黙的に引き継がれる情報で．呼び出しを一意に特定する呼び出し ID，呼び出し元ユーザ名および呼び出し元マシン名が含まれる．呼び出し元情報は，MIDMOST for .NET が出力する各種ログに共通的に出力される．

クライアントおよびサーバの入出力ログメッセージを呼び出し ID でつき合わせることで，障害発生時にどこまでメッセージが到達していたかといった追跡が可能である．また，サーバ上のログに記録された呼び出し元ユーザ名や呼び出し元マシン名から，メッセージがどのユーザやクライアントから要求されたものかを監査することができる．

5.3 閉　　塞

5.3.1 .NET Framework における現状と課題

基幹業務システムでは，オンライン処理を全体または部分的に一定期間サービス停止させ，その間の新規要求に対し，エラーとせずに適切な対応が必要となる局面が存在する．例えば，障害の分析のために，ある特定のサービスだけを比較的長時間停めたい場合がある．この場合は，クライアントにはメンテナンス中の画面を表示するといった対応をとることになる．また，運用中のデータベースのバックアップのために，僅かな時間データベースへのアクセスを止めたいといった場合には，特にクライアントにはサービスが停止していることを意識させず，新たな要求は受け付けたまま保留するという動作をとりたい．

このような処理は閉塞と呼ばれる．閉塞処理を.NET　Framework の標準機能だけで実現するのは非常に難しい．例えば，次のような現状や課題に直面することになる．

- 閉塞情報をどのように保持すべきか，閉塞の単位はどうするか，閉塞指示はどのようにして行うかといった設計上の考慮事項が多く存在する．一から作り込むには設計の負担が大きく，開発コストが増大する．
- COM＋1.5 からは，閉塞に似た機能として特定のコンポーネントの動作を停止させる機能があるが，閉塞単位はコンポーネントのみで，単に例外を呼び出し元に返すという動作しかしない．

5.3.2 MIDMOST for .NET の閉塞機能

前述のように，閉塞処理はミッションクリティカルシステムでは必ず要求される処理であるが，安定した品質の機能を作り込むには非常にコストがかかる．MIDMSOT for .NET の閉塞機能はこうした課題に対する解決策である．MIDMOST for .NET の閉塞機能には，長期閉塞と短期閉塞の 2 種類がある．

- 長期閉塞

  新規要求を受け付けず，呼び出し元に即座に例外を返す．障害分析などのために一定期間特定のサービスをとめる場合に使用する．

- 短期閉塞

  一定期間要求を受け付けながら，受け付けた要求を保留する．保留された要求は閉塞が解除されると処理される．オンライン中のデータベースのバックアップ時などに使用する．

閉塞の単位は事前に定義した任意のメソッドの集合である．MIDMOST for .NET では，こ

の閉塞の単位となるメソッドの集合をサービスブロックと呼ぶ．サービスブロックはクライアントに対するサービスの単位を表す．また，MIDMOST for .NET の監視コンソールを利用して，サービスブロックの単位で実行状況を監視することもできる．

閉塞機能を利用するには，事前にサービスブロック定義ファイルでサービスブロックを定義する．また，サービスブロックに含まれるメソッドが所属するクラスは ServicedComponent クラスから継承して，LockOut 属性を付与しておく．閉塞や閉塞解除の指示は，監視コンソールまたはコマンドラインから特定のサービスブロックを指定して行う．

6. MIDMOST for .NET のアーキテクチャ

本章では，これまでに述べた機能を実現するための MIDMOST for .NET のアーキテクチャについて主に実行環境に着目して述べる．

図 6 に MIDMOST for .NET のアーキテクチャを示す．

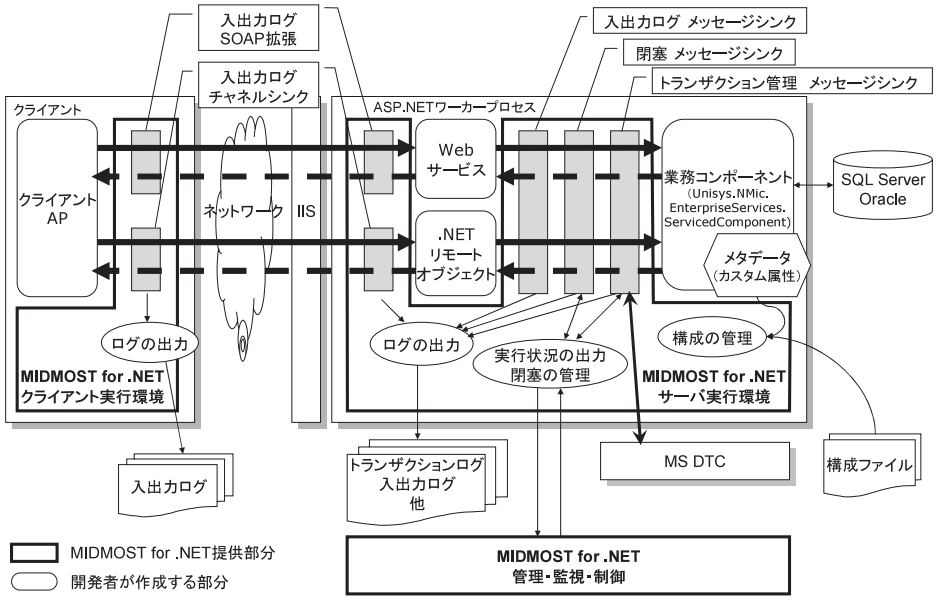

図 6 MIDMOST for .NET のアーキテクチャ

6.1 IIS の活用

MIDMOST for .NET の実行環境の実体は，実行時ライブラリ（DLL）である．業務アプリケーションをホストするサービスは IIS であり，MIDMOST for .NET は業務アプリケーションと同じ ASP.NET ワーカープロセス上で動作する．MIDMOST for .NET では，独自のアプリケーションサーバをたてるのではなく，IIS をアプリケーションサーバとして利用することで，IIS の機能を最大限に活かす設計となっている．

このように，アプリケーションは標準的な ASP.NET アプリケーションの形態であるため，アプリケーションアーキテクチャを設計する際も，ミドルウェアを意識することなく，.NET Framework の標準的な構成を採用できる．

6.2　透過的割り込みによる横断的関心事の分離

MIDMOST for .NET が提供するサービスは，一部の機能を除き，ユーザプログラムから直接呼び出されるのではなく，ユーザプログラムのコンポーネント間の呼び出しに透過的に割り込んで動作する．これによって，業務アプリケーションから横断的関心事の分離を実現する．

透過的な割り込みを実現するために使用したテクノロジーは次の 3 種類である．

- メッセージシンク

  コンポーネントに対するローカルなメソッド呼び出しに対しては，メッセージシンクによる割り込みを行う．MIDMOST for .NET の機能を利用する業務コンポーネント（オブジェクト）は MIDMOST for .NET が定義したコンテキスト内に束縛される．コンテキストとはオブジェクトが存在する環境を定義するプロパティの集合のことである．コンテキストはオブジェクトが生成されるときに作成され，オブジェクトに対するメソッド呼び出しが行われると，独自の処理を実装したメッセージシンクのチェインが構築される．トランザクション管理，入出力ログ，閉塞といった主要な機能はメッセージシンクによって実現している．

- SOAP 拡張機能

  Web サービス呼び出しに対しては，SOAP 拡張機能による割り込みを行う．クライアントから Web サービスを呼び出すと，クライアントからの要求送信，サーバでの要求受信，サーバからの応答送信，クライアントでの応答受信の各ステージにおいて，ASP.NET は SOAP メッセージへのシリアル化あるいは逆シリアル化を行うが，SOAP 拡張機能を利用すると，これらの処理の前後に割り込みを行うことができる．この機能を利用して，クライアント/サーバの双方で SOAP メッセージのロギングを実現している．

- チャネルシンク

  .NET　Remoting 呼び出しに対しては，チャネルシンクによる割り込みを行う．クライアント/サーバ双方のチャネルシンクチェインには，基本機能を提供するフォーマッタシンクやトランスポートシンクのほかに，独自の処理を実装したシンクを挿入することができる．この機能を利用して，クライアント/サーバの双方で.NET　Remoting 呼び出しメッセージのロギングを実現している．

6.3　MS DTC の直接操作によるトランザクション制御

.NET Framework の標準的な MS DTC による自動トランザクションの利用方法は，System.EnterpriseServices 名前空間の ServicedComponennt クラスを継承したクラスを利用する方式だが，この場合 MS DTC へのアクセスは COM＋コンポーネント経由となる．

これに対して，MIDMOST　for　.NET の自動トランザクションは，COM＋コンポーネントを経由することなく，MS　DTC を直接操作する．これによって，COM＋カタログ（レジストリ）への登録が不要となり，大きな問題であった管理性の課題が解決する．

.NET　Framework　1.1/COM＋1.5 から利用可能になった ServiceDomain クラスは，COM＋コンポーネントを作成することなく，COM＋サービスを利用するためのインタフェースを提供している．ServiceDomain クラスの Enter/Leave メソッドで囲まれたコードブロックは，独自の COM＋コンテキストで実行され，あたかもそのコンテキスト内で作成されたオブジェ

クトに対して呼び出されたメソッドであるかのように振る舞う．MIDMOST for .NET のトランザクション管理メッセージシンクでは，ServiceDomain クラスを利用して MS DTC を直接操作している．

7. おわりに

前章までに，.NET Framework ベースのミッションクリティカルシステムの構築・運用に関する現状と課題を MIDMOST for .NET が如何に解決するかを述べた．また，MIDMOST for .NET の設計の根底にあるコンセプトと，それを具現化するアーキテクチャについて述べた．

MIDMOST for .NET のコンセプトは，透過性と親和性という言葉でまとめることができる．MIDMOST for .NET が提供する非業務機能は，開発者が実装する業務処理から完全に分離し，透過的に扱えるようなアーキテクチャを追及した．そして，.NET Framework を始めとする，実績があり，かつ標準的なテクノロジーに対する親和性を追及した．

MIDMOST for .NET は本年 2 月に初期バージョンをリリースした．今回のバージョンでは，ミッションクリティカルシステムの開発に必要な機能が主体となっている．次のフェーズでは運用に必要な機能を強化する予定である．

---

*1 MSCS（Microsoft Cluster Service）の標準機能を補完・拡張し，Windows Server 上での HA（High Availability）システムの構築を支援する基盤ソフトウェアである．迅速な障害検知，的確なリカバリ，適切な情報採取のための機能を提供し，システムの可用性を向上させる．

*2 .NET Framework での企業システム開発を支援するテクニカルドキュメントとテンプレートの総称である．テクニカルドキュメントには設計方針や開発プロセス，コーディング規約などのドキュメントが含まれる．テンプレートには設計方針にしたがったアプリケーション構造を示したサンプルコードと共通部品が含まれる．

参考文献
［1］ 山岸重雄他，BANCS 接続システム，Unisys 技報，第 75 号
［2］ 馬場定行，宮野将彰，佐藤則之，オープン系ミッションクリティカル業務構築を支援する MIDMOST 技術基盤，Unisys 技報，第 77 号
［3］ Don Box,Chris Sells,Essential .NET―共通言語ランタイムの本質，日経 BP ソフトプレス，2003
［4］ Dharma Shukla, Simon Fell,Chris Sells，コードのカプセル化と再利用を推進するアスペクト指向プログラミング，MSDN Magazine 日本語版，March 2002 Vol.17 No.3
［5］ Aspect-Oriented Software Development Community & Conference, http：//aosd.NET/index.php
［6］ Microsoft patterns & practices, http：//www.microsoft.com/japan/resources/practices/default.asp

執筆者紹介 溝 上 昌 宏（Masahiro Mizogami）
1965 年生．1989 年名古屋大学工学部情報工学科卒業．同年日本ユニシス(株)入社．SYSTEMv,ACAB 等の製品開発，.NET 技術支援に従事．現在，日本ユニシス・ソリューション(株)AD CoE コンピテンスセンタに所属し，MIDMOST for .NET の開発に従事．