日本ユニシスにおける開発標準の策定と適用への取り組み

Approach to Design and Application of Development Standards within Nihon Unisys

新 井 敦

要 約 システム開発が大規模化・短期化・複雑化する中で，開発の標準化による生産性と品質の向上が求められている．大規模なシステム開発プロジェクトは，数多くの開発メンバーが同時並行で開発作業を進めるため，開発プロセスを定義し，成果物の作成ルールや処理方式のパターンを規定することで標準化を図り，開発メンバーの経験やスキルの差に起因する成果物品質のバラツキを抑止する必要がある．こうした標準化は，開発技術の選択と組み合わせによる開発プロセスの改善であり，効果を高めるためには，開発プロジェクトの特徴や制約に合わせてカスタマイズし，開発現場でも適用を支援する必要がある．

日本ユニシスグループでは，Java EEベースの開発にはMIDMOST® for Java EEを，.NET Frameworkの開発ではLUCINA® for .NETをそれぞれ提供している．標準化を進めることの期待効果は大きいが，技術の評価と適用判断，人材育成の課題もある．開発標準は，新しい開発技術を取り込んで，ノウハウや知識をフィードバックすることで継続的に改善し，利用価値を高めていく必要があり，組織的な取り組みが求められている．

本稿では，大規模なシステム開発プロジェクトでの設計とプログラミング，テストの工程における標準化の背景や目的，実践的で現実的なアプローチとその効果について概説し，日本ユニシスグループが進める二つの開発標準について紹介する．

Abstract Improvement of productivity and quality by the standardization of development is needed while the system development grows in stature, increases in complexity and its development period shortens increasingly. In a large-scale system development project in which huge number of people perform their development tasks in parallel, it is necessary to prevent the variation in the quality of deliverables resulting from different levels of experiences and skills of development people by standardization such as defining the development process, and prescribing the preparation rule of deliverables and the pattern of processing method. Such standardization is an improvement of the development process by selection and combination of development technologies, and should be customized according to the feature and restrictions of the development project to heighten an effect, and also supported to apply within the development site.

Nihon Unisys Group offers MIDMOST® for Java EE for use in the Java EE based development, LUCINA® for .NET for use in .NET Framework development environment respectively. The expectation effect of promoting the standardization is high, however, there are challenges including evaluation of the technology, the decision on application and the personnel training. The development standard should be continuously improved by taking in new development technologies, and feeding back know-how and knowledge to raise the utility value, and a systematic approach is required.

This paper overviews the background and objectives of standardization throughout the design, programming and testing processes within a large-scale system development project, as well as the practical and realistic approach and its effectiveness, and introduces two development standards that Nihon Unisys is going to promote.

1. はじめに

企業における情報システムは，顧客に対するサービス提供と価値創造の中核的なインフラとして，ビジネス活動における重要性を高めている．優れた情報システムをいかに早く現場に提供できるかが，企業の競争力や活動の成果を大きく左右するため，情報システムは，様々なシステム化要求が機能として盛り込まれて大規模化する一方で，システム開発にかける期間は短期化している．

一方，システム開発現場の環境も急速に変化している．データベースやWebサーバなどのミドルウェア製品や，開発者の作業を効率化するための開発支援ツールは，次々と新しい製品やバージョンが提供され，機能が複合化して高度になっている．また，近年では，オープンソースのミドルウェアや開発支援ツールも機能性と品質が向上しており，急速に普及しつつある．こうしたシステム開発に適用するミドルウェアやツールとその機能が増えるに従って，開発のアプローチやツールの活用方法も高度になり，システム開発のプロセスはますます複雑になっている．

さらに，システム開発が大規模化・短期化することで，設計とプログラミング，テストの工程においては，必要なタイミングで必要な開発メンバーを調達するプロジェクト制組織が導入されており，これらの工程を担う開発メンバーはプロジェクトごとに離合集散を繰り返し，一つの開発プロジェクトは，異なる経験と様々なスキルを持つ開発メンバーが混在する形で新しく構成される．

このように，システム開発プロジェクトは，開発技術とプロジェクトの構成メンバー，開発アプローチを選択して組み合わせ，継続的に改善していくものであり，それぞれのプロジェクトは「新しいメンバーによって，新しい開発技術を適用して，新しいものを，短期間に作り上げる」きわめてクリエイティブな作業であると言える．

こうした数多くの開発メンバーが参画する大規模な開発プロジェクトを成功に導くためには，これまでの開発プロジェクトでの進め方やノウハウを開発標準として集約し，開発メンバーによる経験やスキルの差に起因する成果物品質のバラツキを抑えると共に，共通的な機能を再利用することで，生産性や品質の向上と，技術的リスクの回避を図り，設計からテストまでの一連の作業プロセスの立ち上げをスムースにする開発の標準化が必要になる．

日本ユニシスグループでは，オープン系システム構築における設計とプログラミング，テストの工程を中心とした二つの開発標準の策定を進めており，一部は商品として開発プロジェクトに適用している．一つは，Java EEをベースとした大規模システム向けの開発標準「MIDMOST® for Java EE」であり，もう一つは，Microsoft Windowsの.NET系開発をターゲットとした「LUCINA® for .NET」である．

本稿では，大規模なシステム開発プロジェクトでの設計とプログラミング，テストの工程における標準化の背景や目的，実践的で現実的なアプローチとその効果について概説し，日本ユニシスグループが進める二つの開発標準について紹介する．

2. システム開発における標準化

本章では，システム開発プロジェクトでの設計とプログラミング，テストの工程における標準化の目的や対象，その期待効果について説明する．

2. 1　標準化の背景と目的

情報システムが対象とする業務範囲が拡大することで，開発する画面数や業務ロジックの量が増大する一方，ビジネスの展開や業務改革のスピードに合わせたシステムサービスの提供が求められている．開発の作業量が増える中で開発期間が短期化しているため，大規模な開発プロジェクトでは，設計とプログラミング，テストの工程において，プロジェクト制組織を導入しており，元請けのシステムインテグレータ会社からソフトウェア開発会社に二次発注する階層的な発注構造により，複数の会社で構成される開発プロジェクトが大半である．

また，モデリングツールや統合開発環境，構成管理や品質管理といった開発支援ツールにおいても，次々に新しいツールが登場し，バージョンを重ねることで機能が追加されて改良が進んでいる．ツール間でのデータ連携や実行の同期制御も可能となり，こうした新しいツールやその機能を駆使することを前提として開発プロセスを改善していくことが，生産性や品質を向上させる大きなポイントとなっている．大規模なシステム開発の大きな工程の流れは，設計から実装，テストへと順に工程を進めるウォーターフォール型のアプローチで進めるが，それぞれの工程を細かく見ると，パイロット実装による設計の見直しや，設計と実装を繰り返すことで仕様を確認しつつ技術リスクを回避するスパイラル型のアプローチを随所に適用している．こうした中で，新しい開発支援ツールとその機能を見極めて組み合わせ，適材適所に活用することは，作業の効率化や自動化を図る上で重要であり，それぞれの作業の手順や成果物との関係を示した開発プロセスの定義と継続的な改善が必須である．

このように，開発技術を選択し組み合わせて改善された開発プロセスにより，新たに構成された数多くの開発メンバーが同時並行で開発作業を進めるため，開発メンバーの経験やスキルに起因する設計内容や成果物品質のバラツキが問題となる．例えば，画面上の項目配置と画面間の遷移のパターンや，システム内部の実行制御の基盤上に実装する業務ロジックの処理方式とコーディングスタイル，ID やデータ項目名などのネーミングは，実装する数が多くなるためにバラツキが生じやすい．こうした中で，生産性や品質を一定以上に保つためには，開発のプロセスや成果物に対して一定のルールを設けてメンバーに周知し，これに従って開発作業を進める必要がある．

こうした開発メンバーの経験やスキルの差に起因する成果物のバラツキを抑えるための有効な手段が標準化である．一般に標準化というと，工業製品などにおいて形状や質，大きさについて業界全体で規格を設けて統一することを指す．これに対して，システム開発プロジェクトにおける標準化は，生産性や品質の向上を目的として，全体の開発プロセスを定義し，設計書のフォーマットや記述要領のルールを定め，開発する画面の基本的な構成や処理方式のパターン化，各種 ID のネーミング規約やコーディングルールなどを規定することで，システム開発における成果物の統一化を図り，また，実行制御の共通的な仕組みを確立して，システム全体の品質を向上させるための組織的な取り組みである．

2. 2　標準化すべき対象

システム開発での設計とプログラミング，テストの工程において，標準化が必要となる主な対象を以下に挙げる．

2. 2. 1　アーキテクチャと実行制御機能

一般的に Web ブラウザをクライアントとするような情報システムでは，クライアントからのリクエストを受け付け，画面上で入力された値を解析して業務処理を実行し，その結果として次の画面を表示する．業務処理のロジックでは必要に応じてデータベースを参照し更新する．この一連の処理をどのような実行構造で実現するか，システム内部の構成要素とその関係を定義したものがアーキテクチャである．

アーキテクチャの基本的な考え方として，ここ数年で MVC2[*1] モデルが定着してきている．図 1 に示すように，システム内部の役割を，業務ロジック（Model）と画面の入出力処理（View），システム全体の実行制御（Controller）とに機能分割することで，それぞれの機能の独立性を高めて依存関係を最小化することができ，大規模システムにおける分散・並行開発を可能にしている．

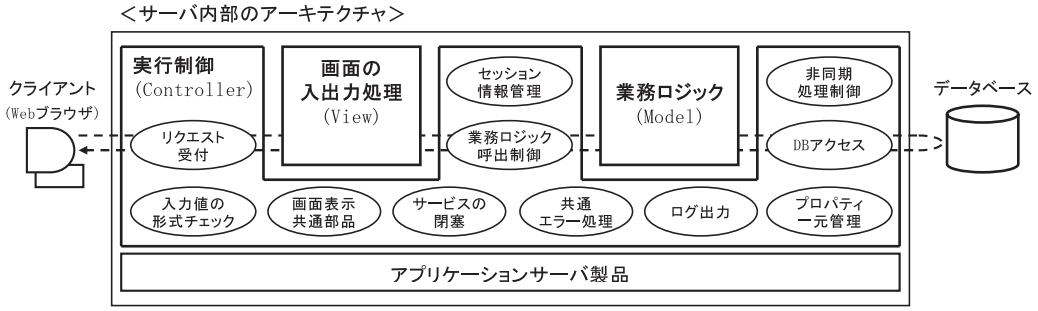

図 1　MVC2 モデルによるアーキテクチャ構造

この中で，データの計算や条件判断，データベースの更新を行う業務ロジック（Model），及び，画面上の入出力項目とそのチェックや次画面を表示する入出力処理（View）は，対象業務によって異なるために個別のシステムごとに開発する必要がある．これに対し，リクエストの受け付け，View や Model のモジュール間の呼び出し，ログ出力や DB アクセス，プロパティの一元管理といった実行制御（Controller）の機能部分は，システムが対象とする業務分野に関わらず同様の機能が求められるため，システムの実行基盤の標準として，複数のシステムで共通化することが可能である．

特に，近年の情報システムにおいては，マルチスレッド対応や分散サーバの実現，各種ミドルウェアやライブラリの組み合わせの多様化により，実行制御の仕組みが専門的かつ技術的に高度になっている．また，ビジネス活動の重要なデータや個人情報がシステム上に集積され，業務遂行上の基幹となっていることから，業務機能だけでなく，情報セキュリティやパフォーマンス，可用性に対する要求も厳しくなってきており，実行制御の機能部分にはシステムを安全に運用するための様々な機能が求められる．これらの非機能要件を担う実行制御の機能部分の開発には高度な技術力を要するため，標準化して複数のシステムで共有することのメリットは大きい．

これらの非機能要件の実装を共通化して一元化することで，OS やミドルウェア，各種ドライバソフトのインターフェイス部分の実装を画面入出力処理や業務ロジックから隠蔽することができ，実行環境に依存する実装を局所化することができる．これにより，サーバ環境の移行や，OS やミドルウェアのバージョンアップによるコード改修やテストの範囲を最小限に抑え

ることができ，システムの環境適応性を高めることができる．

このような実行制御の仕組みは，近年では，SI ベンダー各社もフレームワークとして商品提供しており，オープンソースでは Struts や Spring などが有名である．

2. 2. 2　処理方式パターン

Web ブラウザ上に表示される画面上の項目配置や画面間の遷移の操作は，ユーザの要求次第で様々な実装が可能である．例えば，データベース上のデータを画面から参照・更新するための，値の入力から入力内容の確認，処理結果を表示するまでのデータの見せ方や一連の画面遷移の手順には，いろいろなパターンや表現のスタイルがある．システムが対象とする業務範囲が拡大することで，前提とするユーザ数も増えており，操作方法の習熟や入力操作にかかる時間は業務効率にも大きく影響するため，システム全体で一貫した操作性を提供することで，操作を直感的に理解できるようにする必要がある．

また，システム内部の処理構造についても同様に，様々な実装のパターンがある．開発言語である Java や Microsoft C# の仕様，及び，各種ミドルウェア製品の機能は，開発者の様々なニーズやシステム化機能の要求を満たすために，多様なインターフェイスや実行の方法を提供している．しかし，多様なインターフェイスを駆使することで実現できる機能性は高まるものの，データ構造や実行ロジックが必要以上に複雑になったり，処理の手順や実装のバリエーションが増えたりすることで，データや条件判断の組み合わせの考慮不足や，テストケース漏れによる不具合も増えてくる．また，先駆的な技術の乱用は，実装コードの仕様確認や障害時のトレースを難しくし，システムを保守していく上での技術的なリスクも高まる．

このように，システムの画面遷移や内部の処理構造の実装パターンは，開発者の裁量による自由度が高く，多くの開発者が同時並行で作業を進める大規模開発では，開発者の技術力や過去の経験の差に起因するバラツキが大きくなりやすく，品質や保守性，ユーザの操作性の低下につながる．

こうしたバラツキを防ぐために，画面上の項目配置や遷移の手順，内部処理の実行ロジックをいくつかのパターンに絞り込んで統一し，ミドルウェアの機能を呼び出すための煩雑なデータ管理や，実行手順の制御を共通化することで，処理方式パターンの標準化を図る必要がある．それにより品質や保守性が向上し，業務ロジックがシンプルになって，技術的なリスクを低減することができる．

2. 2. 3　ネーミングとコーディングのルール

ソースコードは，担当する開発メンバーが作成してシステムに組み込むだけでなく，実装仕様や品質を確認するためのコードレビューを行い，障害が発生すれば該当する機能部分のソースコードをトレースしてデバッグする必要がある．この作業を効率良く行うためには，ソースコードの可読性（他人が読んだときの理解のしやすさ）を高めておく必要がある．例えば，同じデータ項目の値を格納する変数の名称は，異なるソースファイル間でも同じ変数名にすることがコードの意味を解釈する上で重要であり，このためにはネーミング規約が必要になる．ソースコードの記述形式も自由度が高く，システム全体でソースコード記述のルールを決めて形式を統一するための，コーディング規約を定める必要がある．

また，設計書やソースコードの中では ID が多用される．ID は，画面やソースファイル名，

エラーの種別やメッセージ文字列，データベース上のテーブル名やデータ項目名などを，システム全体で一意に識別するための文字列である．ID の重複を避けるためには，開発メンバーが自由に発番するのではなく，ネーミング規約として各 ID の文字列構成の体系を定義し，発番や管理の開発運用をプロジェクト全体で一元化する必要がある．

これらの ID は，単に成果物を識別するだけでなく，障害発生時のトレースや，仕様変更時の影響範囲の把握にも用いられ，ソースコードの可読性の高さと合わせて，システムの品質や耐障害性（障害からの復旧の早さ）や機能の拡張性に直結するため，システムのライフサイクル全体を見据えた標準化が求められる．

2. 2. 4　**開発プロセスと運用**

一般に，多くの人が同時並行で同じような作業を進める中で，その品質や効率を高めるためには，作業工程の順序や前後関係，完了基準を明確に定義しておく必要がある．システム開発においても同様であり，担当する機能部分によって内部の処理ロジックは異なるが，設計から実装，テストまでの一連の作業工程において，作成する成果物や作業の手順，完了基準は同じである．

また，それぞれの作業は独立したものではなく，一つの作業の結果として出力した成果物が次の作業の入力となり，成果物を介して全ての作業は前後関係があり，互いにつながっている．システム開発における最終的な成果物は，稼働するシステムに組み込まれるソースファイルや設定ファイルであり，そこまでに積み重ねてきた設計作業とその成果物は，最終成果物に反映される．このような作業の前後関係や成果物とのつながり，それぞれの作業の位置付けや目的を開発プロセスとして定義する必要がある．

さらに，それぞれの開発メンバーが開発プロセス全体を理解して作業することで，個々の作業において作成する成果物が最終的にシステムのどの部分に組み込まれていくのかを意識することができ，それぞれの成果物に記述すべき内容や詳細度を，開発メンバーやレビューアが自律的に判断することができる．

一般的なシステム開発プロジェクトでは，基本設計や詳細設計など，いくつかのフェーズに区切って開発を進めて成果物を納品するが，納品のためだけの成果物を作成するのではなく，最終的な成果物を見通して作業する必要がある．こうした意味でも，それぞれの作業の内容や成果物について目的や関係を詳細に整理し，それぞれの整合性を確保するように開発プロセス全体を標準化して，開発段階での運用を徹底することが重要である．

2. 3　**実践的な標準化のアプローチ**

本節では，実際のシステム開発プロジェクトの現場において，標準化を推進するための実践的なアプローチについて説明する．

2. 3. 1　**継続的な改善とカスタマイズ**

プロジェクトを構成する数多くの開発メンバーが実際に開発作業を進めるためには，標準化の方針や内容を開発規約として定義する必要があり，各開発フェーズの作業を始める前に必要な開発規約が提供されていなければならない．一方で，開発の対象となる業務内容や，前提とするミドルウェアやシステム外部とのデータ連携，開発期間や体制など，システム開発プロジ

ェクトには様々な特徴や制約がある．また，システム開発を発注する顧客側で開発標準が用意されていれば，それとの整合性を取る必要もある．さらに，開発技術やミドルウェア製品とその機能は日々進歩しており，ミドルウェア製品の適用や開発支援ツールの活用が，システム開発の生産性や品質に大きく影響する．

このように，標準化に必要な開発規約は，システム開発を進めるために事前に準備しておく必要があるが，開発するシステムやプロジェクトの特徴に合わせてカスタマイズできる柔軟性が求められる．さらに，新しい開発技術や，すでに検証された実績のある方法を取り込んで継続的に改善していくことが重要となる．開発現場の状況を無視した一方的な標準化や，行き過ぎた標準化は，逆に開発の足かせとなって，形式的で無意味なプロセスと成果物をつくりだし，生産性を落とすことになる．新しい開発技術を取り込んで継続的に改善し，システム開発プロジェクトの特徴や制約に合わせてカスタマイズする必要がある．

2. 3. 2　標準化の推進

標準化によりシステム開発の品質や生産性を高めるためには，それぞれの開発メンバーが，開発プロセス全体の中での位置付けや目的を理解した上で，担当する作業を進める必要がある．しかし，開発規約のドキュメントを作成して公開しただけでは全ての開発メンバーに周知徹底することは難しい．開発作業を進める中でも継続的に標準化の適用を支援して，標準化に従った成果物となっているかをチェックする必要がある．

このためには，プロジェクトのチーム体制の中で，標準化を推進する共通チームを設置することで，標準化方針の説明や理解を徹底し，設計内容に対する標準化のチェックや，技術的な課題解決を，組織的に進める必要がある．また，開発規約に準拠して実装したサンプルとなるアプリケーションを提供することで，開発メンバーが具体的な実装を理解できるようにすることも有効である．

2. 4　標準化の期待効果

システム開発において標準化を進めることで，開発メンバーの経験やスキルの差に起因するバラツキを抑えることができ，品質や生産性を向上させ，また，システムの耐障害性や機能拡張性などの保守性も高まることは，前述した通りである．

本節では，こうした標準化を進めることで波及する期待効果について言及する．

2. 4. 1　開発メンバーの融通と学習効果

標準化による品質と生産性の向上は，一つのプロジェクトに適用するだけでなく，複数のプロジェクトに適用することによって，より大きな効果を発揮することができる．開発プロジェクトによって標準化の方針や内容がそれぞれ異なると，新しくプロジェクトに参画した開発メンバーは，最初に標準化の内容を理解する必要があり，参画してから実際に成果物を作り始めるまでのオーバーヘッドの時間がかかる．全てのプロジェクトで標準化の内容が共通化できていれば，他のプロジェクトに移ってもすぐに実際の作業に取り掛かることができ，プロジェクト間での開発メンバーの融通がスムースになる．さらに，プロジェクトをまたがった開発メンバー間で技術情報を交換し，議論するための共通のリファレンスとして，コミュニケーションの基盤ともなる．

開発メンバーが離合集散を繰り返しているプロジェクト制組織の現状を考えると，プロジェクト間での標準化の方針や内容を共通化することで，他の開発プロジェクトでの経験を活かした即戦力として融通可能な開発メンバーが増え，全体の開発生産性を向上させることにつながる．

2. 4. 2　作業の自動化と品質向上

開発プロセスを定義し，成果物と作業との関係を整理して標準化していく過程で，開発メンバーの手作業をツールで自動化できる作業範囲が見えてくる．作業の工程と成果物には前後関係があり，成果物を介して全ての作業は互いにつながっている．それぞれの作業は，前工程で設計した内容を詳細化・具体化することで付加価値を高めていくことであり，作業そのものを完全に自動化することはできないが，前工程で定義した設計情報を入力として，次に作成する成果物に予め設計情報を埋め込んだ成果物の雛型を自動的に生成する半自動化は可能である．また，ソースコードを作成するために必要な設計情報が設計書に盛り込まれていれば，設計書からソースコードの雛型を自動生成することもできる．

このような作業の自動化を進めるために重要なポイントは，設計書の形式を決めてから自動生成の方法を考えるのではなく，自動生成ができるように設計書の形式や記述内容を決めることである．つまり，最終的な成果物であるソースファイルや設定ファイルの内容を前提に，全体の開発プロセスを定義し，そのために必要な設計書と，それぞれの設計書の記述項目や記述内容を決めていく必要がある．

また，作業の自動化は，作業工数の削減だけではなく，手作業によるコピー＆ペーストのミスや手順の漏れを防いで作業の品質を上げることでも大きなメリットがある．

3.　日本ユニシスグループでの取り組み

日本ユニシスグループにおけるシステム開発の設計とプログラミング，テストの工程を中心とした標準化の取り組みとして，次の二つの開発標準を紹介する．一つは，Java EE をベースとした大規模システム向けの開発標準「MIDMOST for Java EE」であり，もう一つは，Microsoft Windows の .NET 系開発をターゲットとした「LUCINA for .NET」である．

3. 1　MIDMOST for Java EE

MIDMOST for Java EE は，Java EE をベースとする大規模システム開発に適用する開発標準であり，日本ユニシスグループがこれまでの開発プロジェクトで蓄積してきた知識やノウハウを体系的に集大成した統合フレームワーク製品である．300 人月以上の大規模システム開発プロジェクトにおける設計とプログラミング，テストの工程を対象としている．提供先の顧客や開発するシステムの特徴，制約に合わせてカスタマイズすることを前提とした柔軟性の高いイージーオーダー型の開発標準として，図 2 に示すように，設計から実装の開発プロセスで必要となる以下のものを提供している．

- 開発者向けの開発ガイドラインや設計書作成ガイド，実装のサンプル
- 設計や開発運用を標準化するための設計書形式や規約のテンプレート
- システムの実行制御や共通ライブラリのソースコード
- 開発作業を効率化・自動化するための各種ツール

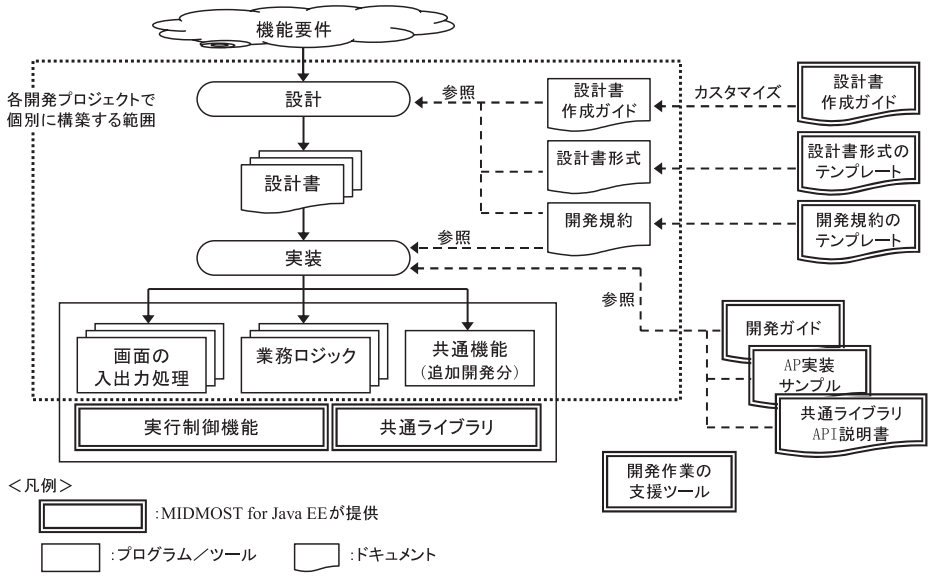

図2　MIDMOST for Java EE **を適用した開発**

この中で，実行制御機能や共通ライブラリは，実際の開発プロジェクトで既に適用実績のある LUCINA Web Foundation（LWF）を基幹として機能を拡充しており，安定した品質と実開発で検証された実行制御の基盤を提供している．一般的なフレームワークが備えている実行制御の機能に加え，エラー制御やログ出力機能の強化，DB アクセス機能部分の開発生産性やパフォーマンスに配慮し，大規模なミッションクリティカルシステムで必要とされる機能を充実させている．これらは，Java EE の仕様に準拠しており，業界標準のアプリケーションサーバである BEA 社の WebLogic Server の他に，オープンソースの JBoss にも対応している．

また，独自の開発支援ツールとして，設計情報のデータを一元的に管理するリポジトリツールを提供している．モジュール間で受け渡す情報の構造やデータベースの定義情報，各種 ID やコード値などの設計情報をデータベース化して一元管理し，各種設定ファイルやソースコード，データベース定義コードの自動生成を可能にしている．さらに，設計書をまたがるデータ項目の属性定義の整合性や一貫性をチェックする機能も備え，設計品質の向上を図ることができる．この他にもオープンソースの開発支援ツールを活用するノウハウを開発プロセスの中に盛り込み，大規模開発プロジェクトを前提とした生産性と品質の向上のノウハウを提供している．

さらに，適用に際しては MIDMOST for Java EE を熟知した専門の技術要員をプロジェクトに常駐させることで，継続的な改善による効果の拡大と，開発標準としての要件やノウハウのフィードバックを図っている．MIDMOST for Java EE を適用するプロジェクト側では，技術要員による支援のもと，網羅的に整備した各種ガイドや規約のテンプレートから，必要な部分をピックアップしてカスタマイズすることで開発標準を策定でき，開発の立ち上がりを短期化できる．また，実行制御機能のソースコードも提供しており，プロジェクト固有のシステム化要件に応じた共通機能の追加や変更，外部システムとの連携にも柔軟に対応できる．

こうした技術支援要員の常駐を通して，実際の開発プロジェクトで挙がった標準化や共通機能に対する新たな要件や，そこで得られた知識，ノウハウをフィードバックすることで，

MIDMOST for Java EE 自体の改善を継続的に進め，利用価値の向上を図っている．

3. 2　LUCINA for .NET

LUCINA for .NET は，日本ユニシスグループでの .NET をベースとする開発プロジェクトでの知識やノウハウを体系化したコンポーネント指向の開発方法である．Microsoft .NET Framework 上で稼働する Web アプリケーションを対象とし，実績のある開発プロセスと実行制御の基盤により，プロジェクトの迅速な立ち上げや並行開発による開発期間の短縮，技術的リスクの低減を図ることを目的としている．LUCINA for .NET は，ツールや .NET 要素技術の適用の指針や開発プロセスを解説した「LUICNA テクニカルドキュメント for .NET」と，設計方針に従った開発テンプレート集である「LUCINA Web Foundation for .NET」で構成されている．LUCINA for .NET の特徴と設計方針，適用による効果を以下に挙げる．

1）　コンポーネント指向開発方法

コンポーネントによって開発単位を最適化して段階的に組み上げ，インターフェイスと実装とを分離する疎結合モデルの指針として，システム内部の処理構造の推奨パターンやクラス分割の指針を提供することで，大規模システム開発において並行開発による開発期間の短期化と，ビジネス要求への柔軟な対応を可能としている．

2）　標準化された開発プロセスと設計方針

検証済みの開発プロセスや実行制御の基盤，開発テンプレートをベースに，フェーズ分割や体制の考え方，作業タスクの構成や成果物の記述内容を定義し，開発支援ツールの活用方法や技術的リスクへの対処を具体的に提示しており，プロジェクトを立ち上げる際の負担を軽減させることができる．

3）　リスク駆動型開発

事前に定義された開発プロセスや実行制御の基盤を再利用することで技術リスクを低減させるとともに，開発の早い段階で技術リスクを抽出するため，設計にかかる期間やコストの削減，アプリケーションの品質向上を図ることができる．

.NET Framework は，Web アプリケーションフレームワーク（ASP.NET）とデータベース接続フレームワーク（ADO.NET）を備え，基本的なセッション管理，入出力の形式チェック，ログ出力の機能も付随しており，LUCINA for .NET ではこれらに準拠した設計指針を提供している．また，開発環境の面でも Microsoft Visual Studio .NET を十分に活用するためのノウハウを提供している．事前に検証された技術的リスクの少ない開発プロセスやコンポーネントの設計指針，開発テンプレートを提供することで，開発者は本来のビジネスロジックの開発に専念することが可能となり，構築フェーズにおける品質と生産性の向上や，開発期間の短縮を図ることができる．

.NET Framework には開発者の視点で使いやすい機能が豊富に用意されており，比較的容易にアプリケーションを構築することが可能である反面，機能の選択や組み合わせの自由度が高いため，大規模なシステム開発では，処理方式パターンの多様化による保守性や品質の低下，

及び，先駆的な機能の乱用によるセキュリティ面での脆弱性が問題となるケースがある．このため，コンポーネントの粒度や構造の設計方針，コーディング規約だけでなく，サンプルコードを開発テンプレートとして開発者に提供することで，適用する機能の統一化を図っている．

LUCINA for .NET においても，実際の開発プロジェクトに適用する際には，プロジェクトに合わせたカスタマイズが必要となる．独自機能を実装した実行制御のライブラリや成果物テンプレートの提供，画面のデザインや遷移，処理方式パターンの標準化など，LUCINA for .NET が標準として提供しているものをベースに，個別の開発プロジェクトに合わせた適用を支援している．

こうした .NET をベースとする開発プロジェクトで蓄積されたノウハウや知識は社内で一元的に集約して管理し，最新技術も取り入れながら洗練してフィードバックすることで，.NET Framework のバージョンアップと合わせて，進化した LUCINA for .NET をリリースしている．

4. 標準化における課題

こうした開発標準の策定と適用，開発プロジェクトでの標準化の推進においては，以下に挙げるような課題もある．

1） 先駆的な開発技術の適用判断

先駆的な機能や高度な技術を駆使することで，ユーザが求める機能要件を高い水準で満たすことができるケースがある．しかし，システム開発プロジェクトの開発メンバーは必ずしも全てが高いスキルや豊富な経験を有しているわけではない．突出した高度な技術をもつ開発メンバーが担当する一部の機能については，先駆的な機能を駆使して高い機能性を実現できても，システム全体としてのバランスや操作性は悪くなってしまう．このため，標準化としてはシステム全体の統一性を優先し，成熟した開発技術を採用して処理方式をパターン化することで，先駆的な技術の適用を抑制する場合もある．つまり，高度な機能性の実現と，標準化の推進とは相反するケースがある．

プロジェクトの開発メンバーのスキルや開発規模，技術の必要性や成熟性を見極めて，どこまで先駆的な開発技術を適用するかを判断した上で，標準化を策定する必要がある．

2） 標準化を推進する開発技術者の育成と組織力の強化

標準化はマニュアル化して配布すれば推進できるものではない．新しい開発技術の適用を評価し，開発プロジェクトに合わせてカスタマイズし，開発を進める中で適用を支援していかなければならない．標準化を推進して支援するためには豊富な開発経験と高い視点，広範な技術知識が求められる．このため，標準化を推進する要員は慢性的に不足しているのが現状であり，標準化を推進できる技術要員を組織的に育成していく必要がある．

さらに，個々の技術者のスキルアップだけでなく，組織として標準化の策定と適用を推進する体制が求められており，組織力の強化も重要な課題である．

3） 開発メンバーの創造性の発揮

システム全体の操作性や機能性を標準化することは，全体としての成果物品質を底上げすることには大きな効果があるが，一方で，標準化は開発者の裁量の範囲や自由度を制限する

ことであり，開発者の創造性や技術スキルを抑えてしまうことにもなる．

システム開発プロジェクトにおいて開発メンバーが行う設計やプログラミング，テストの作業は，作業した時間数に比例した量の成果物が得られる労働集約型の作業ではなく，それぞれの開発者がスキルや経験，知識をフルに活用する知識集約型の作業である．

どこまでを標準化し，どこまでを開発メンバーの裁量の範囲とするかのバランスが重要であり，開発メンバーの裁量の範囲を適切にコントロールすることで，開発メンバーが創造性を発揮できるようにすべきである．

こうした課題を認識した上で技術や状況に合わせて判断し，継続的に改善していかなければ，効果的な標準化の策定と適用を推進していくことは難しい．

5. おわりに

本稿で述べてきた標準化は，大規模なシステム開発プロジェクトにおける設計とプログラミング，テストの工程を中心としたものであるが，情報システムとしての標準化は，顧客企業のIT 全般や業務プロセス，情報セキュリティまで含めた幅広い範囲での取り組みが必要である．統制可能な業務フローやマネジメントの仕組みの構築，正当性や網羅性を維持継続するための業務データ体系の整理，セキュリティ監視の運用管理体制の確立など，IT 戦略と企画，要件定義の段階から，様々な視点で標準化を図ることが重要である．

また，標準化のノウハウは複数の開発プロジェクトで共有すべきものであり，それぞれの開発プロジェクトでゼロから考えるよりも，既存の成果物を参照し，必要な事項や考えるべき要素をピックアップして再構成する方が，品質や生産性も高い．既存の成果物がそのままの形で使えなくても，考え方や事例としてだけでも十分に活用できる．何かを新しく考えて作ることは，コストとリスクを発生させることであり，ゼロから始めるよりは，検証済みで信頼できる既存の成果物を集めて取捨選択し，部分的に再利用してカスタマイズする方が，生産性も品質も高い．

日本ユニシスグループにおいては，Java EE ベースの開発には MIDMOST for Java EE を，.NET Farmework の開発では LUCINA for .NET をそれぞれ提供し，システム開発における標準化を進めている．それぞれの開発プロジェクトで生み出されたノウハウや新たな知識を，ベースとなる開発標準にフィードバックすることで，次々に新しいノウハウや知識が付け加わり，開発標準の利用価値を高めて進化させていくことができ，それを適用する次のプロジェクトでの開発標準はより良いものになる．こうした標準化を推進するための組織的な取り組みが求められている．

＊ 1　MVC2 は，Java ベースの Web アプリケーションを構築するための開発モデルである．Smalltalk-80（スモールトーク）で確立された MVC モデルを基本として Java EE の技術要素に対応させ，システム内部構造を Model-View-Controller に分けて実装する考え方を示している．

参考文献　［1］　ナンシー・M. ディクソン著，梅本勝博他訳，「ナレッジ・マネジメント 5 つの方法」，生産性出版，2003 年
［2］　野中郁次郎，竹内弘高著，梅本勝博訳，「知識創造企業」，東洋経済新報，1996 年

［3］ メアリー・ポッペンディーク，トム・ポッペンディーク著，平鍋健児，高嶋優子，佐野建樹訳，「Lean Software Development」，日経 BP 社，2004 年
［4］ 新井敦，「技術経営の発想による IT 企業での技術知識の活用」，ユニシス技報，日本ユニシス，Vol.24　No.4　通巻 84 号，2005 年 2 月

執筆者紹介 新　井　　敦（Atsushi Arai）
1992 年筑波大学卒業．同年日本ユニシス（株）入社．電力・ガス・通信の顧客システム構築を中心に，開発プロジェクトを手掛ける．情報処理技術者 アプリケーション・エンジニア．企業派遣により，早稲田大学大学院アジア太平洋研究科 MOT コース（国際経営学修士課程）を 2004 年に卒業．現在，総合技術研究所 OSS センターに所属し，社内プロダクトの開発と，開発プロジェクトへの技術支援を担当．