Vol.2012-GN-82 No.16
Vol.2012-CDS-3 No.16
2012/1/20

ＯＭＣＳのシステムモデルによるプラットフォーム構築

相澤正俊† 東健二††

オープン製品で基幹システムを構築する OMCS（Open Mission Critical System）において、それまでのメインフレーム同等以上のプラットフォームを構築することが最大の課題であった。そのために 3 層モデルを拡張したシステムモデルを開発して世界レベルの基幹システムを実現した。システムモデル開発の基本概念は、システムメトリックス導入による製品選定作業と Multi Unit Architecture によるシステムモデルの部品（ユニット）の再利用である。さらにシステムモデルによるシステム構築技法としてのスパイラル開発技法を開発し、システムモデルの発展形として、クラウド時代のプラットフォームとして Resource Pooling System を開発した。

System Model for OMCS platform development

MASATOSHI AIZAWA† KENJI HIGASHI††

The most important technical issue in OMCS was to guarantee Mission Critical capability equal to or higher than Main Frame system. The solution is System Model oriented Platform Development which is extension of 3-Tiers Model. The Basic ideas of System Model oriented Platform Development are an introduction of System Metrics and a reuse of a part of System by MUA(Multi Unit Architecture). And MUA enables the spiral development of system. This paper also describes Resource Pooling System as a extension of System Model.

1. はじめに

OMCS（Open Mission Critical Systems）とは、オープン製品やオープン技術を駆使して構築された大規模基幹システムと、それらを構築するための製品群および SI（System Integration）技術のメソドロジー（方法論）の総称である。OMCS の歴史的背景、発展の歴史の概説は参考文献[1]に詳しい。OMCS を支える中核テクノロジーは、Mission Critical な特性を持つ「システム部品」の組み合わせから成るシステムモデルとそれを用いたシステム構築技術、並びに“見える化”を徹底したプロジェクト管理技法である。これらは、現実のシステム開発の現場で直面した多くの問題を解決すべく実際に行った活動のアイディアを発展させてきたものの集大成である。本論文ではシステムモデルによるシステム構築を中心に述べる。

2. システムモデルによる構築技法の狙い

オープン製品を用いてメインフレームと同等以上の堅牢なプラットフォーム（PF）を構築することを最大のテーマとし、３層モデルを拡張したシステムモデルを開発してきたが、システムモデルによる構築技法の狙いは、以下の２つのリスクの排除ということができる。

- リスク１
  「Best of Breed[a]による多様性に起因するリスク」といえるもので、各社各様さまざまな製品が入り乱れ、過度に自由度の高いアーキテクチャや実装方式が混在している状態から生まれるリスクである。
- リスク２
  「システムに対する要求仕様の多様性に起因するリスク」とも言うべきもので、メインフレームと同等以上の性能や信頼性・拡張性はもとより、携帯電話に代表される急激なユーザ増や負荷変動への迅速な対応、メインフレームの数段上のスケーラビリティ等、これまでに無かった新たな要件がシステムに求められるようになったことに起因するリスクである。

3. 似て非なるシステムの乱立

システムをモデル化する最初の観点は“処理形態”である。アプリケーション視点で概観すると、世の中には多様なシステムが存在するように見えるが、視点を変えると、オンライン処理やバッチ処理、ゲートウェイ処理、大量イベント処理など代表的な幾つかの処理形態に分類可能であることが判る。

オンライン処理やバッチ処理は、勘定系、予約系、企業基幹系など、従来から存在する普遍的な処理形態であり、主に、MFR[b]領域で重要な処理形態である。ゲートウェイ処理は、いわゆる Hub としての処理や、ISP[c]に求められる処理形態であり、大量イ

† 国際社会経済研究所 理事長（前ＮＥＣ副社長） Institute for International Socio-Economic Studies
†† 日本電気株式会社 ＯＭＣＳ事業本部 事業本部長 NEC Corporation

a グローバルにデファクトスタンダードになっているハードウェア製品やプログラムプロダクトを活用すること
b MainFrame Replacement
c Internet Service Provider

Vol.2012-GN-82 No.16
Vol.2012-CDS-3 No.16
2012/1/20

ベント処理は、携帯電話の料金計算をリアルタイムで行うような数万件／秒以上の処理を必要とするようなものである。これについては[2]に詳しい。

この様に、代表的処理形態はそれほど多くはないが、一方で図１に示すように、多くの不確定要素から、似て非なるシステムが乱立するのである。

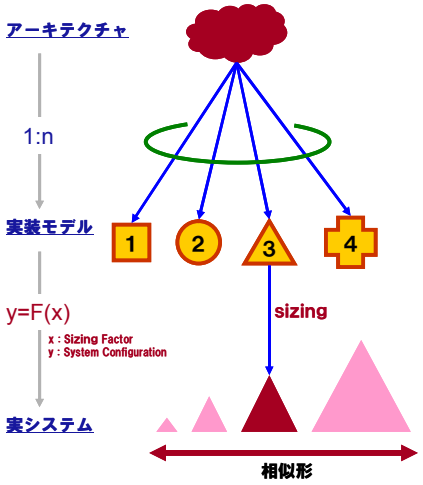


図１：似て非なるシステム構成の出現

すわなち、クライアントサーバ型や三層構造など、システムの構築様式から実システムに至る道は複数存在する。アーキテクチャを起点とし、実装方式の確定とプロダクトスタック、機能配置を確定する道も複数あり、通常、どれを選択するかは現場のシステム設計担当者に委ねられている。さらに、サイジング工程を経て、実システム構成が確定するが、サイジング要素も確定根拠も多様である。以上の結果として、単一アーキテクチャを起点としたシステム開発であるにも係わらず、似て非なるシステムが乱立するのである。実際のプロジェクトでの顧客へのプロポーザルにおいても、各社各様のシステム形態が提案されることも多々ある。

この様な乱立を回避し、均質なプラットフォーム（PF）構築を目指すのが、「OMCSのシステムモデルによるプラットフォーム構築」である。

4. 多様性リスクを排除すべき作業

多様性リスクを排除し、均質な PF 構築技法の確立のために必要な作業や“やり方”を確定するため、通常実施されている PF 構築手順の中から、多様性リスクによって失敗すると致命傷になる重要な作業を洗い出すと下記三点になる。

（１）　システム要件確定作業

性能や信頼性などの非機能要件について、客先キーマンとの合意が不十分な状態で構築が進み、不具合発生時、甚大なビジネスインパクトが発生し大問題になるケースが少なくない。

（２）　製品選定作業

利用実績も無く、実証評価もされてない製品（=初物）を機能的観点や製品の新規性のみから採用し、不具合発生時に甚大な対応コストが発生するケースが少なくない。

（３）　実装方式確定作業

個人の知識とノウハウ、経験に頼りきった PF 設計技法および PF 構築技法により、同じ処理形態、同じ規模感、同じ製品構成であるにも係わらず、バラバラのシステムが出来上がり、その出来具合（システム品質）もバラバラとなるケースが散見される。

5. システム要件確定作業と MC 性

システム要件確定作業から、極力、多様性リスクを排除するために、その定義と考え方を整理する。ただし、定義自体の完成度を高めるため、要件確定の曖昧さや浅さが、システムの不具合発生時、甚大なるビジネスインパクトに発展する可能性が高い重要なシステム（基幹系システム）を対象とし、シンプルなクライアントサーバモデル等は対象としない。

基幹システムが本来提供する機能以外に具備すべき特性を MC 性（mission critical capability）と呼ぶ。ここで、本来提供すべき機能とは、業務アプリケーションが提供する機能であり、すなわちプログラム（＝純然たるビジネスロジック＋データ）が動作することで提供される機能である。上記機能定義の補集合として残る部分、すなわち、ハードウェア、OS、ミドルウェア、アプリケーションフレームワーク（以降、AP-FW）などを総称してシステムプラットフォーム（PF）と改めて定義する。

基幹システム、特にオープン系大規模分散システムの構築実績におけるシステム要件を分析すると、以下の六つの特性に類型化できる。

- 高可用性

サービス継続性維持のための特性であり、システム運用で計画されたサービス停止以外に発生する予期せぬサービス停止を極小化し、その影響範囲を局所化するための特性である。予期せぬサービス停止の要因は、運用操作ミス、ハードウェアの経年劣化、ソフトバグ等様々で、どの程度の対策を講じるかが重要なシステム要件である。

なお、システムレベルの高可用性を実現するためには、システム構成要素の全てに考慮が必要であり、それにはアプリケーション自体も含まれる。

- 高性能性

性能劣化要因を排除した結果得られる特性だが、システム内一意通番取得のように

Vol.2012-GN-82 No.16
Vol.2012-CDS-3 No.16
2012/1/20

システムワイドで一貫性を保つことが必要な情報への更新系アクセスが全てのトランザクション区間内に存在する場合や複数のデータベース間の一貫性保障が必要な場合など性能劣化要因の排除が困難なケースも存在する。アプリケーションも含め、こういった隘路をどう克服するかが重要な課題である。

- 高運用性

多数ノードからなる分散システムを構成するハードウェア機器から、OS やミドルウェア、業務アプリケーションによって作り出される業務やサービスまでを一元運用可能とする機能である。運用は監視と操作に大別され、構成管理機能と密に連携する。

- 高拡張性

システムを取り巻く環境変化や社会的な環境変化により発生する通信量増加、事務量増加、情報量増加などに最適投資で追従できる特性である。最適投資を保証するためには、全サービスダウンを伴わない、小さな粒度の段階的拡張が必須である。

- 高機密性

インターネットの浸透により、旧来 MF による基幹システムで主流であった専用線の世界とは異なる観点であり、ネットワークセキュリティからシステムセキュリティ、業務セキュリティ、情報漏洩、内部統制と、その幅は広い。

- 高連携性

他システムとの高い連携性のことだが、単なるファイル転送からサービスを含めたトランザクショナルな連携まで幅広い特性である。

以下の表は、MC 性を実現する狙いや製品の組み合わせによるファシリティの例を示したものである。

表１：MC 性とそれを実現するファシリティの例

	MC性	MC性の狙い	ファシリティ
1	高可用性	企業の基幹,社会インフラシステムに,障害対応・保守・拡張を無停止で実現	プロセス監視,浮動予備 リモート監視,センタ縮退運用 Oracle,Tuxedo連携 流量制御・スロースタートによるシステム安定稼働
2	高性能性	通信業システム,大型ECサイトなど大量データを高速に処理	分散 DB設計,FCS・EMC技術 バッチ階層化,分散並列実行 超並列スレッドAP,流量制御方式
3	高運用性	大規模なデータセンターやマルチセンターを一元的に監視	ｼｮｱｸﾞﾙｰﾌﾟ単位の自動ﾘﾗﾝｽｹｼｭｰﾘﾝｸﾞ MoMによる一元監視,遠隔照会・監視 静止元帳作成,日替処理による連続運用
4	高拡張性	インターネット対応ビジネスなどの急激な成長に柔軟に対応	AP動的置換,スケールアウト方式, DB分割方式 独立拡張可能なサブシステム構成
5	高機密性	ECサイト,顧客情報DBなど,重要データを犯罪からガード	OS・LBによる攻撃回避,ツールによる診断 未知の攻撃パターン検出技術 ISO標準に独自要件を加えた設計・運用
6	高連携性	勘定系,情報系,企業間などシステム間を安全に接続	SOA(Service-Oriented Architecture) SOAP連携 ESB基盤(Enterprise Service Bus連携)

6. 課題に対する考え方と対策

4 節で述べた、多様性リスクを排除すべき作業フェーズと、５節で述べた MC 性を踏まえ次の対策を行った。

- 求められる MC 性からいかなる製品を選択すべきかの道筋を示すものとして、システムメトリックスを用いた製品選定作業
- 導き出された製品群を如何に再利用可能にするかを目指し、多次元的連携が表現可能で実装モデルへの対応を可能とする MUA（Multi Unit Architecture）の導入
- これらの考え方を統合した、３レイヤによるプラットフォーム構築技法

以下に各々を具体的に説明する。

6.1 システムメトリックスを導入した製品選定作業

ソフトウェア品質を計測する基準として Software Metrics（例えば品質会計等）があるように、システムが MC 性を保証できる計測基準としてシステムメトリックス（System Metrics）の考え方を新たに導入することとした。システム要件を MC 性で確定した後、システム構成要素である個々の製品が具備すべき機能に落とし込む“やり方”を規定している。

システムメトリックス体系は下記三層から成る。

（１） 要件カテゴリ　前述の MC 性を構成する六つの特性毎に要件を定義したもの
（２） システム要件　システム的観点から満足しなければならない要件。
（３） 構成要素要件　基盤構築要素（HW、OS、ミドルウェア）がシステム要件に関して満足しなければならない要件。

上記三層を具体例で示す。要件カテゴリのうち高拡張性を例に取り、下記のようなシステム要件が定義されたとする。

① HUB 層、AP 層、DB 層など、各層独立した拡張性を持つこと
② システム構成拡張時、業務アプリケーションの修正が不要なこと
③ サーバやディスク、ネットワーク機器など構成要素単位で拡張可能なこと
④ 拡張作業は、業務システム全体を止めずにできること
⑤ データベースは scale out 可能なこと
⑥ 業務サーバの動的追加が可能なこと
⑦ スケーラブルな性能向上が可能な構成であること
⑧ SW 諸元値の上限が存在しないこと（＝実装 HW に依存するのみ）

上記④に着目し、一例としてデータベースサーバにおけるデータオーバフロー対策を考えると、その構成要素に対する要件としては下記のような要件が導出される。

① データベース自体の動的拡張が可能なこと
② 上記を満足するために、論理ボリュームの動的追加が可能なこと
③ 上記を満足するために、物理ボリュームの動的追加が可能なこと

Vol.2012-GN-82 No.16
Vol.2012-CDS-3 No.16
2012/1/20

上記三要件を満足する個々の製品、すなわち、データベース製品、オペレーティングシステム（論理ボリューム管理）、ストレージ機器が連携することで初めて実現できるのである。メインフレーム時代は、上記三要件は一社に閉じた世界で担保されていたが、オープン時代になり整合された製品計画に基づいて開発されているわけではない複数ベンダーの製品組合せとなった。ここが、一社垂直統合とマルチベンダー水平分散の最大の違いである。

こういった一連の作業による製品選定作業を、システムメトリックスを用いた製品選定作業と呼ぶ。

6.2 実装方式確定作業と MUA

プラットフォーム構築の“考え方”として、マルチユニットアーキテクチャ（MUA）を導入する。MUA とは、自社製品とワールドワイドで認められた Best of Breed 製品やオープンソース（OSS）を用いて、さまざまな機能が複雑に連携する（協調する）大規模なシステムを“安全・確実・スピーディ”に構築するための“考え方”である。昨今、分散システムアーキテクチャとして、分散システム指向のミドルウェア製品や分散 AP 基盤の浸透により、三層アーキテクチャや多層アーキテクチャが浸透してきたが、いずれも最上位のマクロアーキテクチャレベルであり、即物的な実システム構成との隔たりは大きい。従い、アーキテクチャから実装まで１：１の落とし込みが難しく、単一アーキテクチャから、デザイナのスキルに依存した複数の実装モデルができてしまい、均質な横展開が難しい。

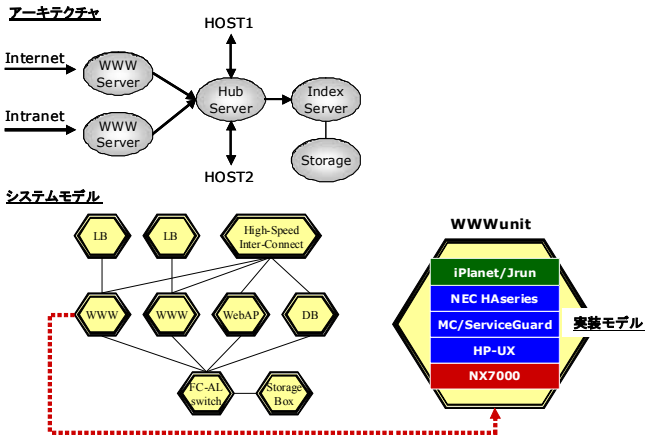


図２：MUA のイメージ（事例：企業統合に伴うシステム連携ＨＵＢ）

大規模分散システムとして、多種多様な機能が相互に協調動作するシステムを想定すると、ネットワークプロトコルのような、垂直方向に伸びる単なるレイヤのスタックでは適合せず、多次元的な連携が表現可能でかつ１：１で実装モデルにブレークダウン可能な考え方・メソドロジーが必要となる。そこでシステム基盤を、共通的な機能の塊に分離し、それぞれを“システム構築ユニット”として定義することにより、システムアーキテクチャレベルから実装モデルまでをシームレス（１対１）に繋ぐ役割を持たせる。このようにユニットを唯一のアブストラクション（抽象化）とし、その組み合わせでシステムモデルを表現する考え方をマルチユニットアーキテクチャ（MUA）と称する。

図２は、アーキテクチャから、システムモデルが設計され、システム構築ユニット（六角形の図）に分解され、各システム構築ユニット（例：WWW Unit）が製品により構築されているイメージを示したものである。このようにシステム構築ユニットとして定義される共通的な機能の例としては、トランザクション制御機能、データベース管理機能、バッチ制御機能、WebAP 制御機能などサーバ上のソフトウェアで具現化される機能や、L2 スイッチや L3 スイッチ、ロードバランサーなどネットワークアプライアンスとして提供される機能などである。

6.3 多様性リスク排除と抽象化

一方、システム設計とは、非常に高い抽象度で表現されたシステムコンセプトやアーキテクチャ、曖昧模糊としたシステム要件を、即物的な実システム構成に落とし込む作業のため、段階的な抽象度の遷移（高→低）を経て完結する。幾つかの設計事例を分析した結果を基に、PF 設計技法における抽象度のレベル分けを下記３レイヤで定義する。

- レベル１：概念モデル
  システムコンセプト（ターゲットシステムの狙いや効果等の明確化）
  システムアーキテクチャ（関連システムや人間系を含めた実現イメージ；青写真）
  システム要件（性能や可用性、拡張性、機密性などシステムとして具備すべき特性の定義）
- レベル２：論理モデル
  機能相関（システムを構成する機能間の関連定義）
  例：　三層アーキテクチャにおける画面制御機能－オンライン処理機能－RDB 管理機能など
  機能スタック（システムを構成する機能階層）
  例：ストレージ機能、サーバ機能、OS 機能、クラスタ制御機能、トランザクションモニタ機能、アプリケーションフレームワークなど
- レベル３：物理モデル
  静的モデルとして、プロセスモデル、拡張モデル、運用モデルなど。
  動的モデルとして、データフローモデル、性能モデル、コントロールフローモデ

Vol.2012-GN-82 No.16
Vol.2012-CDS-3 No.16
2012/1/20

ル、高可用モデルなどが存在する。

図３に３レイヤによるプラットフォーム構築アプローチの概念を示す。概念→論理→物理と段階的に落とし込んで行った抽象度を逆流することで即物的なシステム構成へと導く。すなわち、レベル 3 の物理モデルは、抽象化と具現化の要であり、実システム構成への起点となる。レベル 1 に位置付けられる モデルを“システムモデル”と呼び、サイジングファクタを確定することにより実システム構成が導出される。 レベル 2 にはシステムモデルを構成する唯一の要素である“システム構築ユニット”が位置づけられ、製品スタックが固定化された状態で存在する。

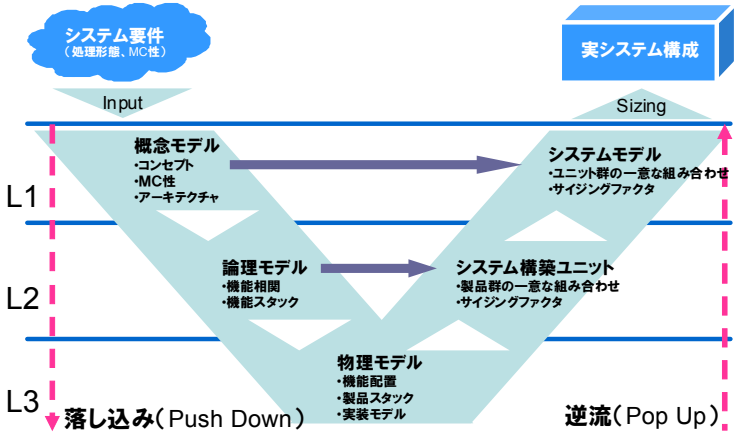


図３：３レイヤによるプラットフォーム構築アプローチ

6.4 ３レイヤによるプラットフォーム構築アプローチがもたらす効果

この３レイヤによるプラットフォーム構築アプローチの考え方により、プラットフォーム構築の要件整理が容易になり、実績のあるユニットを部品として使用することにより堅牢なプラットフォームを短期間にかつ安全に構築することができるようになる。図４は、プラットフォーム構築の期間短縮によるコスト削減効果のイメージを示したものである。各ユニットは、下記三タイプで実装される。

① 既存ユニット 100%流用のケース（＝システムモデル全面適用）
② 構成要素の変更に伴うユニットの改造が発生するケース
③ 新規ユニットを開発するケース（設計コストの他、MC 性検証で大きなコストが発生）

2010 年に構築した地銀向けバンキングシステムの事例では、2003 年の最初の地銀向けバンキングシステムの開発[1]で作成されたユニット（③のケース）を流用することにより（①のケース＋マイナー改造）、期間、コストともに 1/2 以下で実現することができた。

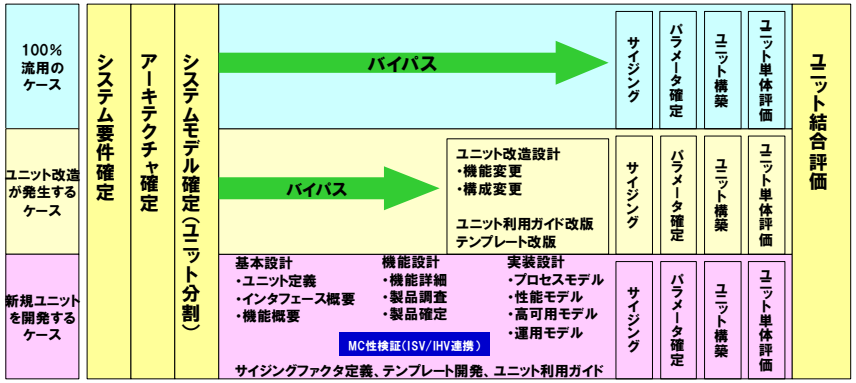


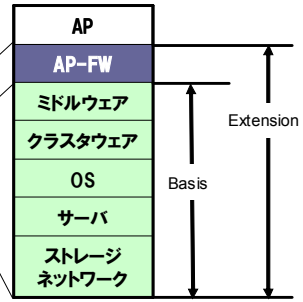
図４：期間短縮によるコスト削減のイメージ

7. システムモデルの拡張

システムモデルの構成要素である“システム構築ユニット”のうち、アプリケーションが実装される構築ユニットは、トランザクションシステムの中核となる OLTP ユニットや、Web 三層の中間層である WebAP ユニットである。当初、それらの機能スタックにおける規定範囲は、ハードウェアからミドルウェアまでであった。 しかし、システムモデルの適用事例が増加するのに伴い、同一システムモデルであってもシステム特性が全く異なるシステムが散見されるようになった（下表）。

表２：システムモデルの変遷

適用システム	無停止オンライン モデル	地銀向け 超高信頼性モデル	メガバンク向け 超高信頼性モデル
業務AP	–	–	–
AP-FW	OpenDIOSA	OpenDIOSA2*	DIOSA/OE**
構築ユニット	OLTPユニット	←	←
性能（tps）	500件規模	100件規模	2,000件規模
DB切替	10分程度	30秒	30秒
システム領域	ビリング	地銀勘定系	メガバンク勘定系



*OpenDIOSA2,**DIOSA/OE：いずれもNEC製ミドルウェアOpenDIOSAの進化形である

その主たる要因は、アプリケーションフレームワーク（AP-FW）である。

Vol.2012-GN-82 No.16
Vol.2012-CDS-3 No.16
2012/1/20

ミドルウェアまでの規定範囲を Basis、AP-FW までの規定範囲を Extension と呼ぶ。

オープン時代の三層構造（プレゼンテーション-アプリケーション-DB）を起点に、高信頼性オンラインシステムに向けて HUB 層を付加したセンター内三層（HUB-AP-DB）や、超並列バッチ処理基盤に向けた「収集-加工-蓄積」からなる三層構造など(派生三層)、新たな処理形態に追従するためのモデルが新規創出されてきた。当初は、ハードウェアからミドルウェアまでを規定範囲にしてきたシステムプラットフォームも、新たなアプリケーション実行基盤（アプリケーションフレームワーク（AP-FW）含む）を取り込むよう拡張してきた。当初のアプリケーション（AP）は、独自の AP-FW 上でその都度開発することが主体(スクラッチ開発型)であったが、Java の浸透と伴に、世間で流通している AP-FW を採用する傾向が強まる。昨今は、パッケージ AP の利用が増え、NEC でも 2 年かけて社内基幹システム（G1：Global One）を、SAP で構築し、ERP 領域のシステムモデルをパッケージ型へ拡張した。

8. 先進的システムモデル

OMCS により世界最先端の大規模システムが数多く構築されてきた[1]が、それらの中で特に以下の３つのシステムモデルは重要である。

- 超高信頼性システムモデル[1]
  従来メインフレームで行われていた高信頼性を要求される業務をオープン系システムに移行することを目標としたシステムモデル
- サービスプラットフォーム（SPF）システムモデルの超並列システムモデル[1]
  通信キャリアが提供するメール、インターネットアクセス等を実現する、巨大 ISP（Internet Service Provider）システムをモデル化したもの
- サービスプラットフォーム（SPF）システムモデルの PSA[d]システムモデル[1][2]
  IT と NW が融合するユビキタス社会における、次世代の大量イベント収集/加工業務システム（数万件/秒以上）でのスーパーOLTP としてのリアルタイム処理システム基盤

9. システムモデルを活用したスパイラル開発技法

今まで述べてきたシステムをユニットに分割する考え方は、アプリケーションを含めたシステム全体の開発にも、高い効果をもたらした。それがスパイラル開発技法である。スパイラル開発技法はコンカレントエンジニアリングの１つの実現法といえる。以下、それを説明する。

9.1 スパイラル開発技法の狙いと手法

大規模なオープン・システムの開発においては、システム構成の自由度がないメインフレーム時代と異なり、自由度があるが故、アプリケーション群を開発するプロセスと、アプリケーション群を動作させるシステムプラットフォーム構築のプロセスの両面が必要となり、以下の様な種々な問題が発生した。

- アプリケーションの評価環境の構築が遅れ、アプリケーションの開発が遅延。
- アプリケーション群とシステムプラットフォームの整合がとれず、誤動作・性能問題の発覚が遅れ、納期に致命的な影響をおよぼし、1990 年代前半当時、米国においても、サーバを数百台使用する大規模分散システムでは、実質的に開発停止や大幅遅延に追い込まれることがあった。[3]

これらの対策として、スパイラルに同時にアプリケーションとプラットフォームを開発、構築していくことにより、システムの最小単位から業務システムとしての評価を可能とするスパイラル開発を実践した。具体的な対策を以下に示す。

- システムモデルのユニットを単位として、アプリケーションの評価環境を早期に構築・提供する。
- アプリケーション開発フレームワークを規定し、ユニットとの親和性を保障する。
- アプリケーションの開発をメイン業務から優先付けて行い、評価環境での動作確認・性能評価を早い時点で行う。
- ユニットの種類の追加、規模の拡大とステップを踏みながらシステムプラットフォームを構築するのと並行して、アプリケーションのサブ業務の開発を行い、評価環境での動作確認・性能評価を行いながら、問題点の早期発見に努める。

9.2 スパイラル開発技法の適用例と実績

スパイラル開発は、システムの最小単位（１/n モデル）での業務システムの評価が可能になることにより構築、作り込み時点でのバグや不整合を最小限に抑えることが可能となり業務システムとしての保証が早い段階で実証できる。また、最小機能構成で業務単体性能から検証して、評価モデルに応じた性能評価を実施可能となる。

事例として、大規模 ISP システムの性能検証のケースを以下に示す。

- 商用機の 1/4 の相似形を構築し大規模環境(サーバ 202 台)の検証、チューニングを実施
- 5 回のスパイラル検証、チューニングにより検証の漏れを徹底排除
  ①単体（擬似 AP、実 AP) → ②隣接コンポーネント間 → ③主要サービスルート → ④システム全体(試験機) → ⑤システム全体(商用機)
- 実トラヒック、突発トラヒックの忠実な再現、1 週間連続の 10 万トランザクション/秒の超高負荷の発生により、高負荷時に突発トラヒック、擬似障害等を発生させ、障害局所化、輻輳制御等のアーキテクチャを確認
- ピーク時のスケーラビリティ、安定性検証のため、各サーバの主要ルートに対して 3 倍の高負荷により確認
- 大規模構成の効率的、漏れのない検証のためのフックとして、サービスログ出力、

d Parallel Stream Architecture

Vol.2012-GN-82 No.16
Vol.2012-CDS-3 No.16
2012/1/20

TAT(Turn Around Time)集計機能を設計時から作りこみ、このフックから自動収集、集計、判定することで、ピークトラヒック 3.6 億トランザクション/時の膨大なトラヒックの検証効率化、漏れのない検証実施

本検証ケースでは、５回のスパイラル検証全てにおいて性能目標値をクリアし、後戻り工数実質ゼロを実現した。結果、実際のシステム構築を１年という極めて短期間で実現した。この事例が示すように、むしろサーバ数百台規模のシステムではスパイラル開発技法を用いなければ、プロジェクトの途中で空中分解に陥るだろう。

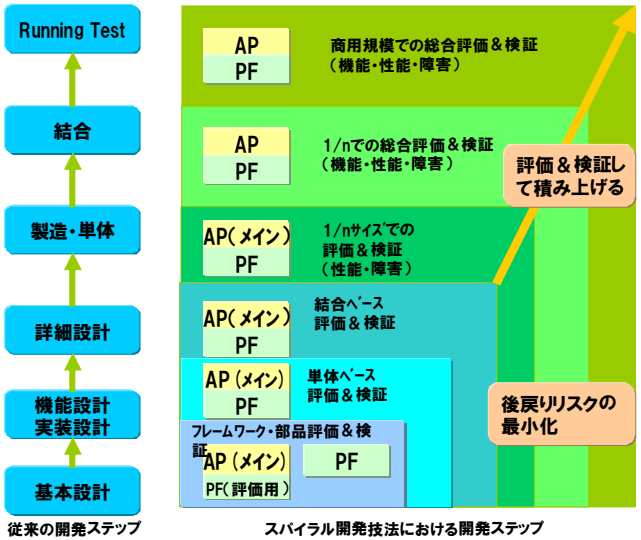


図５：従来（ウォーターフォール型）と比較したスパイラル開発の開発ステップ

10. 次世代 IT 基盤の“あるべき姿” （システムモデルの発展形として）

昨今、業種によらず事業環境不透明さもあり、以前にも増して効率的な IT 基盤を求める声が高まっている。この背景には急激に広まった各種クラウドサービスの影響も大きい。エンタープライズにおけるプライベートクラウド基盤に対する要望を整理すると“徹底的なコストダウンの追及”が見えてくる。この要求に対して OMCS では“リソースプーリングシステム”を定義し展開してきた。7 節で触れた SAP ベースの弊社社内 ERP システムも適用事例の一つであり、そのエッセンスは以下のようになる。

- モデルベースのシンプル統合監視で、監視コストダウン
- モデルベースのシンプル運用で、運用コストダウン（特に AP 運用）
- 仮想化技術を駆使した高密度実装で、フロア単金ダウン
- サービスレベルを松竹梅で定義し、システムモデルでメニュー化
- システムモデルで製品スタック固定化し、製品集約し、調達・維持コストダウン

また、“スピード感を持った構築とサービス提供”も重要であり、それを実現するため、以下を実現している。

- 「現状分析・コンサルありき」ではなく、「まずはじめよう」の考え方（Try and Error）
- 必要な基盤が、すぐに準備できる（ex. 翌日リリース）
- 稼動後の資源再配置（re-balancing）による効率活用

10.1 RPS のアーキテクチャ概説

RPS のシステムアーキテクチャは、システムモデル領域とリソースプール領域の二階層から成る。図６にイメージを示す。

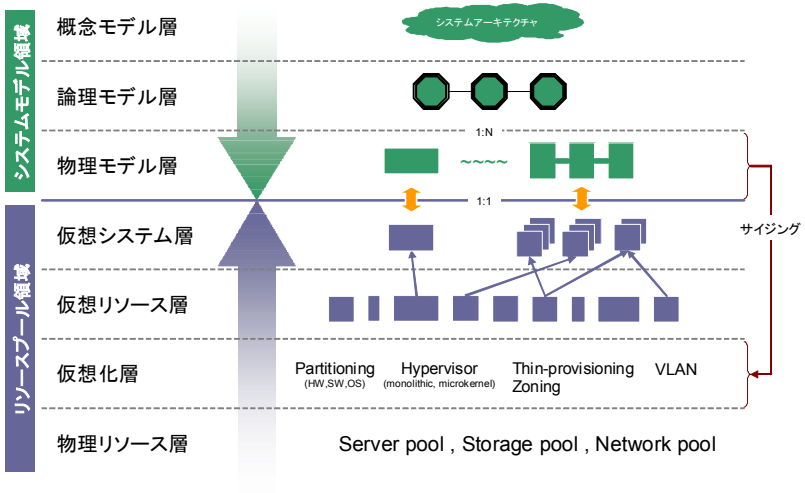


図６：RPS のシステムアーキテクチャ

上位層のシステムモデル領域は、前述のシステムモデルの構成要素である三層から成る。すなわち、概念モデル、論理モデル、物理モデルである。下位層のリソースプール領域は、物理リソース層、仮想化層、仮想リソース層、仮想システム層の四層から成り、それぞれ下記の役割を持つ。

- 物理リソース層：サーバ、ストレージ、ネットワーク各ハードウェア機材を予め準備したもの
- 仮想化層：物理リソース層にプールされた各種ハードウェアから適量を切り出す機能群

Vol.2012-GN-82 No.16
Vol.2012-CDS-3 No.16
2012/1/20

- 仮想リソース層：仮想化層の機能で最適に切り出された仮想リソース群
- 仮想システム層：仮想リソース群を組合せて構成された仮想化されたシステム

10.2 仮想システムの切り出し手順

システムモデルを構成するシステム構築ユニットで定義されたサイジングファクタを確定することで、物理的に必要なリソースが確定する。単品の SI 作業の場合、確定したリソースを発注し、納品・現調を経てプラットフォーム構築へと移るのであるが、RPS の場合は、必要なリソースを物理リソース層から切り出し、仮想化されたシステムを生成するため、物理モデル層で確定したサイジング結果は、仮想化層への入力情報となる。典型的な三層システムの例を図７に示す。

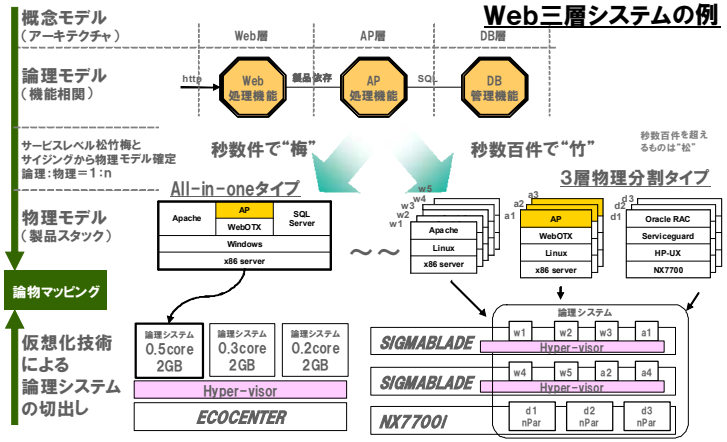


図７：仮想プールからの切り出しイメージ

11. まとめ

オープン・システムの初期の３層モデルを大幅に拡張して新たに考案した拡張型システムモデルを利用し、かつ、それをベースに開発したコンカレントエンジニアリングコンセプトのスパイラル開発技法を利用して、世界最大規模のオープン・システム開発を達成した。システムモデルとそれを構成する MUA（Multi Unit Architecture）、並びにスパイラル開発技法が大規模なオープン・システムへ適用されたのは、世界的にも先進的な事例であると考える。

NEC においては、システムモデルによるプラットフォーム構築はその後も実績を積み上げており、2010 年度には新たに 15 のシステムモデル（ユニット）が追加されている。

今後は、インターネットのユビキタスコンピューティングの進展により、新しいシステムモデルの出現とシステムモデル自身の更なる発展が期待される。

（１）システムモデル自身の更なる発展

- スマートフォン等の端末の進化に伴う平均使用パケット量の飛躍的（２～３桁）増大や震災対応の堅牢なデータセンターの構築に伴う OMCS 構築技術の更なる発展が期待される。
- 現在 Google 等がサービスしている DWH（Data Ware House）は検索中心であるが、真のリアルタイム経営のための新しい OLTP システムモデルは、PSA 型を発展させた大福帳システム（更新型 DWH）になると考えている。

（２）システムモデルを表現するモデル記述法の開発による設計ツールや評価ツール等の登場により、システムモデルによる構築作業のエンジニアリング化と効率化の進展が期待される。

謝辞

システムモデルによるプラットフォーム構築は現実のシステム開発における必要性から生まれたものであり、現場への適用に際して、色々と検討いただいた OMCS 各プロジェクトの皆様に感謝したい。更に具体的なシステムモデル開発においては、一部製品の改造も含め製品の提供・サポートにご尽力いただいた製品パートナーでもある ISV/IHV の皆さんへも感謝申しあげる。

参考文献

1) 相澤正俊他：「ＯＭＣＳ構築技術の研究」
2) 相澤正俊他：「ＯＭＣＳのＰＳＡシステムモデル」
3) 星野友彦：「KDD、空前の難プロジェクトを貫徹」 日経コンピュータ 1999.11.8 号