情報処理学会論文誌　コンシューマ・デバイス & システム　Vol.2　No.3　1–15 (Dec. 2012)

コンシューマ・サービス論文

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[1,a)]　東 健二[2]　川浦 立志[2]

受付日 2012年6月6日, 採録日 2012年7月26日

概要：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．本論文で述べるシステムモデル開発の特徴は，Mission Critical 性の定義に始まり，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）の再利用等，モデルをどのように作り，どのように利用・発展させてゆくかの統一的手法を定めたことにある．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，これらは NTT ドコモの i モードシステムやおサイフケータイを使ったクレジット決済システムの構築に活用された．また，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．

キーワード：オープンシステム，ミッションクリティカル，システムモデル，OMCS，MUA

System Model for OMCS Platform Development

Masatoshi Aizawa[1,a)]　Kenji Higashi[2]　Tatsushi Kawaura[2]

Received: June 6, 2012, Accepted: July 26, 2012

Abstract: 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. These technologies are used in i-mode system of NTT DoCoMo and credit system for wallet-mobile-terminal. This paper also describes Resource Pooling System as an extension of System Model.

Keywords: open system, mission critical, system model, OMCS, MUA

1. はじめに

OMCS（Open Mission Critical System）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（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．

[1] 国際社会経済研究所
Institute for International Socio-Economic Studies, Minato, Tokyo 108–0073, Japan
[2] 日本電気株式会社
NEC Corporation, Minato, Tokyo 108–8001, Japan
[a)] m-aizawa@bc.jp.nec.com

1990 年代，パーソナルコンピュータとインターネットの普及により，ネットワークを介した情報共有や情報検索が一般化したが，その後，携帯電話からのインターネット接続が普及することで，その利用者数は飛躍的に伸張した．さらに，ここ数年のスマートフォン急増により，その利用形態が多様化，従来の参照系主体の処理に加え，商品購買にともなう決済処理や口座間の資金移動等，高いサービス品質が必要とされる処理が増加傾向にある．今後さらに，M2M（Machine to Machine）による大量イベント処理やビッグデータ処理による多様なサービスが出現すると想定され，コンシューマ向けのサービスを提供するシステムには，従来以上の高い性能，高い拡張性，高い信頼性が求め

られる．

NEC では 1990 年代から他社に先駆けて，オープン製品やオープン技術を駆使した大規模基幹システムの構築プロジェクトに着手し，これまで多様な基幹システム構築プロジェクトの実践を通し，システムインテグレーション技術（SI 技術）を確立してきた．その成果の代表的事例は，NTT ドコモの i モードシステムである．数千万の携帯電話からのメールやインターネットアクセスを可能とするために，1 千台を超えるサーバやネットワーク機器が稼動している．この巨大システムに万一大きな障害が発生した場合の社会的インパクトは容易に想像できるであろう．また，おサイフケータイのクレジット機能の決済システムや銀行の勘定系システムも代表的な OMCS 事例である．昨今の携帯電話やスマートフォンによるネットバンキングやクレジットの要求は，背後にある銀行の基幹システム（「勘定」を行うシステムなので勘定系システムと呼ばれる）やクレジット決済処理を行うシステムに通知され，処理される．これらのシステムに障害が発生すれば，銀行 ATM にまで被害が及ぶこともあり，その社会的インパクトは，計り知れない．こういった大規模基幹システムを安全かつ確実に構築することを支えたのが，OMCS の SI 技術であり，この確立された OMCS の SI 技術は，今後ますます多様化するコンシューマ向けサービスを提供するシステム構築に必要となる技術である．

OMCS を支える中核テクノロジは，Mission Critical な特性を持つ「システム部品」の組合せからなるシステムモデルとそれを用いたシステム構築技術，ならびに “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. 似て非なるシステムの乱立とその弊害

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，大量イベント処理等，代表的ないくつかの処理形態に分類可能であることが分かる．

オンライン処理やバッチ処理は，勘定系，予約系，企業基幹系等，従来から存在する普遍的な処理形態であり，主に，MFR [*1] 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 [*2] としての処理や，ISP [*3] に求められる処理形態であり，大量イベント処理は，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．これについては参考文献 [2] に詳しい．

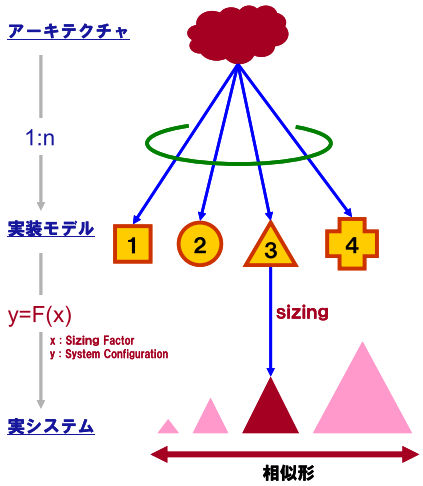


図 1　似て非なるシステム構成の出現
Fig. 1　Variety of system configurations.

このように，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．

すなわち，クライアントサーバ型や 3 Tier Architecture [*4] [3] 等，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かかわらず，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．

このような乱立を回避し，均質なプラットフォーム構築を目指すのが，「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．

3. 解決すべき課題と目標

3.1 従来の手法と解決すべき課題

各社各様の様々な製品が入り乱れ，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の脱却を目指す試みは，過去にいろいろ行われており，その代表的手法は，検証された製品のセットを用意し，それを利用

[*1] MainFrame Replacement
[*2] HUB とは，スター型 LAN を構成する際の集線装置のことであるが，IT システム構築の場面では，複数のシステムをつなぐ役割を持ったサーバ等を「HUB サーバ」のように呼称する．
[*3] Internet Service Provider
[*4] システムをプレゼンテーション層，アプリケーション（AP）層，データベース（DB）層の 3 つの層に分解することにより，それぞれの層の独立性を高め，柔軟なシステム構築を可能とする考え方．Open Environment Corporation（OEC）の John J. Donovan 氏により提唱された．

するという考え方である．参考文献 [4], [5], [6] に示される例は，その典型的なものである．しかし，その製品のセットをいかに作るかについての統一的手法について明示されたものは見受けられない．

さらに，検証された製品のセットを作る検証作業には，多大な手間と時間が必要である．それは，各社の製品は各社が個別に設定したポリシのもとに作成されていることが大きな影響を与えている．これは業務プログラムの開発やメインフレーム上の製品群と比較してみると理解しやすい．同一のアプリケーションフレームワークに従って作成された業務プログラムであっても，プログラム間の整合性を保障するために多くの労力が割かれているのが現実である．従来のメインフレーム上の OS，ミドルウェア等は，1 つのベンダの統一的ポリシに従い作成されており，製品間の整合性はベンダにより保証されていた．しかし，種々のオープン製品群は，個々のベンダごとに微妙に異なるポリシや思想に従って開発されたもので，個々の製品群が組み合わされて間違いなくうまく動作することを検証することは非常に困難であることは容易に想像できるだろう．逆にいえば，事業としてのシステム構築を考えると，1 度検証された製品のセットを，いかに効率的に活用するかが重要なポイントとなるが，参考文献 [4] に見られるように，「同一のシステムの複製」を作ることに活用することは知られていても，「1 つの検証された製品のセット」をいかに発展させ，別の用途に活用してゆくかの道筋を示したものは見受けられない．

このように，1 つの検証された製品のセットをいかにつくるか，そしてそれをいかに活用し発展させてゆくかについての，統一的道筋を定めることが本論文で解決すべき課題である．

3.2　課題解決のための基本的アイディアと検討すべき内容

まず，本論文で対象とするシステムについて定義する．

システムの不具合が発生したとき，甚大な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．

さらに，基幹システムの本来提供すべき機能（業務アプリケーションが提供する機能．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）以外のハードウェア，OS，ミドルウェア，アプリケーションフレームワーク等を総称してシステムプラットフォーム，ないし，単にプラットフォームと称することにする．本論文で述べるのは，基幹系システムのシステムプラットフォーム構築に関する方法論である．

部品化やプログラム構造の規定等，プログラムの自由度を制約することにより，高品質なプログラムのより効率的な開発を目指すものが，アプリケーションフレームワークであるが，システムプラットフォームについても，同様に，システムエンジニアの設計の自由度を制約して，高品質なシステムを効率的に構築するための手法の基本的アイディアは部品化である．このため，我々は MUA（Multi Unit Architecture）を定め，導入することとした．

この部品を組み合わせてシステムプラットフォームを構築するという基本アイディアを前提として，従来の問題点として考慮すべきことは，以下のようなことである．

(1) 基幹システムの要件（部品の要件）をいかに整理するか

基幹システムの構築に先立って，性能や信頼性等の非機能要件について利用者・供給者間での合意することが重要であることは，経済産業省「情報システムの信頼性向上に関するガイドライン第 2 版」[7] でも述べられているが，本来行われるべき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．基幹システムの要件をきちんと洗い出し，整理するための統一的手法が必要である．

非機能要件の整理という観点では，参考文献 [8] に類似の試みが記載されているが，我々は，独自に MC 性（Mission Critical capability）を定義し，活用した．

(2) 部品を構成する製品をどのように選定するか

利用実績もなく，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．種々多様な製品群の中から，要件に合致する製品群を選び出すための統一的手法が必要である．

我々は MC 性を起点として製品選定を行う，システムメトリックスを定義し，実践した．

(3) その部品を構成する製品群の品質（特に，製品群の組合せに依存する品質）の確保をいかに担保するか

前述したように，この品質の確保には多大な時間と工数を要する．この品質確保のための検証作業は，一般には有識者や熟練者による網羅的評価が行われることが多く，まさにシステムインテグレーションを行う会社の資産ともいうべきノウハウである[*5]．その効率化や，統一的手法の確立が望まれていることは確かであるが，本論文ではこの検証に関する具体的手法については言及しない[*6]．

[*5] ある事例では，オープン製品を提供しているベンダ（ISV/IHV）の社内に，顧客のアプリケーションを持ち込んで評価環境を作らせ，ベンダ自身に評価をさせる手法をとった．これも 1 つのノウハウである．

[*6] 本論文の主眼は，検証された製品のセットを「同一のシステムの複製」を作ることに活用することだけでなく，検証された製品のセットをいかに発展させ，別の用途に活用してゆくかの道筋を示すことにある．しかし，その過程の中で，製品のセットの品質を保証する行為は避けて通ることができない．そこで課題として提示するにとどめている．

(4) 部品群をいかに組み合わせて，客先のシステムを構築するか

個人の知識とノウハウ，経験に頼りきったプラットフォーム設計技法およびプラットフォーム構築技法により，同じ処理形態，同じ規模感，同じ製品構成であるにもかかわらず，バラバラのシステムができあがり，そのでき具合（システム品質）もバラバラとなるケースが散見される．

検証された部品群が揃ったとして，基幹システムの要件から，実システムを作り出す流れの統一的手法が必要であることは明らかである．

我々は 3 レベルのプラットフォーム構築アプローチを定義し実践した．

(5) 性能問題の早期発見

システム構成の自由度がないメインフレームの場合と異なり，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あるがゆえに，アプリケーション群を開発するプロセスに加え，さらに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という 2 つのプロセスが必要となり，以下のよう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及ぼし，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 [9]．

特に，アプリケーションのロジックの検証は，比較的早期に行うことが可能であるが，性能評価については，それを意識した開発とそれを意識した環境整備が必要で，従来のウオーターフォール型の開発では，最後に大きな性能問題が発覚するケースが散見された．また，擬似環境（エミュレータ等）では性能評価は困難で，なるべく商用環境に近い形での早期の評価が重要である．

我々はスパイラル開発技法を定義し，実践した．

3.3 目標

これらの課題に対する施策を行うことにより，システムプラットフォーム構築の工数を 1/2 に削減することを目標とする．また，性能問題の早期発見については，開発期間を 1/2 に短縮することを目標とする．

4. OMCS のシステムモデルによるプラットフォームの構築

以下，3.2 節で述べた課題に対する対応を詳細に説明する．

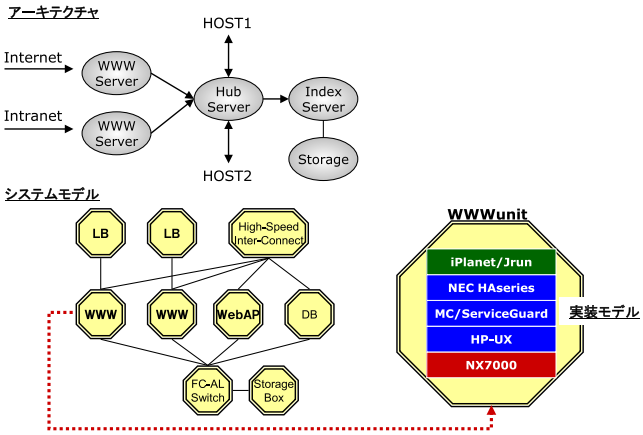


図 2　MUA のイメージ（事例：企業統合にともなうシステム連携 HUB）

Fig. 2　Image of Multi Unit Architecture.

4.1 システムプラットフォーム構築における部品化の考え方（MUA）

システムプラットフォーム構築の方法として，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，様々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” に構築するための方法である．

システムプラットフォームを，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．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 制御機能等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，ロードバランサ等ネットワークアプライアンスとして提供される機能等であり，これらは，長年のシステム開発の歴史の中で，徐々に蓄積されてきたものである．

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（八角形の図）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を単位として，システム構築ユニットの組合せによりシステムを構築してゆく考え方がマルチユニットアーキテクチャである．

4.2 基幹システムの要件（システム構築ユニットの要件）整理（MC 性の定義）

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．すなわち，システムプラットフォームが具備すべき特性が MC 性である．

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 6 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，ソフトウェアのバグ等様々で，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すべ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すべ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等，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，情報量増加等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ともなわない，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，旧来メインフレーム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．

これらの特性は，より細かく細分化され，各々の特性につき，客先キーマンと事前調整が行われる．実際には長年のシステム構築の経験から蓄積されたパターンが数多く存在し，それが適用されることが多い．

しかし，昨今の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が求められるシステム等，これまでになかったような新たなシステムの開発を迫られるときには，この 6 つの MC 性に立ち返り，要件を整理するのである．

個々の特性に関する詳細な説明や細分化は省略するが，その一端は次の節の中で触れる．

4.3　システム構築ユニットを構成する製品選定（システムメトリックスの導入）

メトリックスとは，定量的尺度のことで，1982 年ソフトウェアメトリックスとしてソフトウェアの分野にも導入されている [10]．昨今は，開発工程そのものを対象とした測定方法により，開発工程の監視や制御を行えるようにしたものが主流になりつつあり，NEC の品質会計制度もその 1 つといえる [11]．また，経済産業省の委託により，参考文献 [12] のような報告書も作成されている．ソフトウェア品質を計測する基準としてソフトウェアメトリックスがあるように，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）を新たに定義し，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方法を規定している．

システムメトリックス体系は下記の 3 つのレベルからなる．

(1) 要件カテゴリ　前述の MC 性を構成する 6 つの特性ごと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 3 つのレベルを具体例で示す．要件カテゴリのうち高拡張性を例にとり，下記のようなシステム要件が定義されたとする．

① HUB 層[*7]，AP 層，DB 層等，各層が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，ネットワーク機器等，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 AP 層を構成する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 DB 層を構成するデータ

[*7] プレゼンテーション層，AP 層，DB 層を持つ，3 Tier Architecture に追加された層で，AP 層を構成する複数のサーバに対する負荷分散や他システムへの接続等を実現する層．

表 1　MC 性とそれを実現するファシリティの例
Table 1　Mission critical capability and facilities.

	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	ジョブグループ単位の自動リラン,スケジューリング 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)

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

上記 3 つの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求められるデータオーバフロー対策が実現できるのである．メインフレームの場合には，上記 3 つの要件は 1 社に閉じた世界で担保されていたが，オープン・システムの場合には，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の製品の組合せで実現することとなった．ここが，メインフレームにおける 1 社垂直統合とオープン・システムにおけるマルチベンダ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．システムメトリックスは，システムプラットフォームの MC 性を示す評価パラメータだけでなく，システムプラットフォームの設計手法として活用されているのである．

表 1 は，製品選定作業の結果，選定された製品群により提供される機能（ファシリティ）の例である．これらのファシリティが，システム構築ユニットに組み込まれることになる．

4.4　システム構築ユニットから基幹システムを構築する考え方（システムモデル）

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．いくつかの設計事例を分析した結果をもとに，システムプラットフォーム設計技法における抽象度のレベル分けを下記 3 つのレベルで定義することとした．

- 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）
  基幹システムの要件（性能や可用性，拡張性，機密性等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：3 Tiers Architecture におけるプレゼンテーション層，AP 層，DB 層等
  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，トランザクション制御機能，アプリケーションフレームワーク等
- 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，運用モデル等．
  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ル，高可用モデル等が存在する．

図 3 に 3 つのレベルからな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い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

表 2　システムモデルとシステム構築ユニットの関係
Table 2　System models and system units.

ユニット	ServerUnit					NetworkUnit		StorageUnit		
システムモデル	HA	OLTP	BATCH	WebAP	HUB	Inter-Connect	Load-Balancer	FC-AL Switch	Storage	実システム
大規模バッチモデル	○		○			○			○	通信業顧客料金システム
無停止オンラインモデル	○	○	○	○		○			○	通信業基幹システム
超高信頼性モデル	○	○	○		○	○			○	地銀向けバンキングシステム
HUB統合モデル	○			○	○	○	○	○	○	メガ損保システム
SPFシステムモデル	○		○	○	○	○	○	○	○	ISP システム

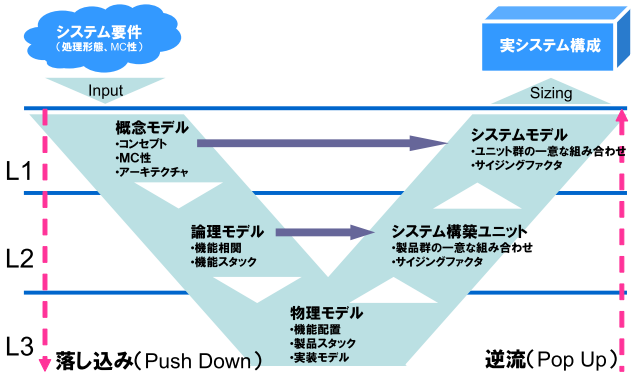


図 3　3 つのレベルからなるプラットフォーム構築アプローチ
Fig. 3　Three level approach for system platform development.

固定化された状態で存在する．

システム構築ユニットに対してサイジングファクタを確定することにより，小さなシステムから大きなシステムまで，相似形のシステム（実システムを構成する部品）を構築できることに注意されたい．結果として，相似形の実システムが構築できることを意味する．この特性が後に述べるスパイラル開発技法で大きな意味を持つ．

4.5　システムモデルの拡張

システムモデルの拡張には，「既存のシステムモデルに，あらたなシステム構築ユニットが追加されてゆく」，いわば種類の拡張と，「従来ミドルウェア領域までの規定であったものが，アプリケーションフレームワークやパッケージアプリケーションの領域まで規定されてゆく」，いわば規定の範囲の拡張がある．以下，実例を示して説明する．

表 2 はシステムモデルとシステム構築ユニットの関係を示したものである．個々のシステムモデルに関しては，参考文献 [1] に詳しい．

大規模バッチモデルは，1998 年に稼動した通信業顧客料金システムをベースとしたもので，最初のシステムモデルである．超並列バッチ処理（3 億呼/日，6 万バッチ/日），24 時間 365 日無停止無人センタ運用，マルチセンタ管理（激甚災害対策）等の特徴を持つ．この最初のモデルは，3 Tier Architecture がクライアントサーバ型の処理をプレゼンテーション，AP，DB の 3 つに分割したものであることを参考にして，バッチ処理を「収集–加工–蓄積」の 3 つの段階に分解し，収集を行うサーバ群，加工を行うサーバ群，蓄積を行うサーバ群を配置することで超並列バッチ処理を実現するもので，BATCH ユニットが中心となって構築されたシステムモデルである．

無停止オンラインモデルは，1999 年に稼動した通信業基幹システムをベースとしたもので，高性能トランザクション処理（330 件/秒），24 時間ゼロサービスダウン，業務無停止によるシステムの構成拡張，DOA [*8]指向の業務 AP 開発（6M Step）等の特徴を持つ．無停止オンラインモデルは，大規模バッチモデルに OLTP [*9]ユニット，WebAP ユニット[*10]が追加されていることが分かる．

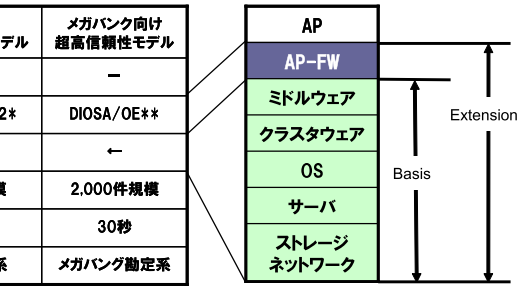
超高信頼性モデルは 2003 年に稼動した地銀向けバンキングシステムに代表されるシステムで，30 秒高速サーバ切り替え，OT [*11]指向の業務 AP アーキテクチャによるオープン勘定系パッケージ BankingWeb21（BW21）（9M Step）等の特徴を持つ．3 Tier Architecture に HUB 層を新たに付加した構造[*12]が決定され，結果，ここでは新たに HUB ユニットが追加されている．

このように，表 2 では，従来とは異なる新たなシステム要件が現れるたびに，過去に作成したシステム構築ユニッ

[*8] Data Oriented Approach
[*9] OnLine Transaction Processing
[*10] Web ブラウザの高機能化にともない，プレゼンテーション機能は Web ブラウザを共通で利用する考え方が普及し，それにともない，3 Tier Architecture の AP 層を Web サーバならびに Web サーバ上のアプリケーションで実現する考え方が出現した．これを Web 三層アーキテクチャと呼称している．この Web サーバならびに Web サーバ上のアプリケーションの実行環境が WebAP ユニットである．
[*11] Object Technology
[*12] データセンタ内に，HUB 層，AP 層，DB 層の三層が存在することから，センタ内三層モデルと呼称している．

表 3　システムモデルの変遷
Table 3　Transitions of system model.

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	メガバング勘定系



*OpenDIOSA2,**DIOSA/OE：いずれもNEC製ミドルウェアOpenDIOSAの進化形である
いずれも、AP-FWを規定している

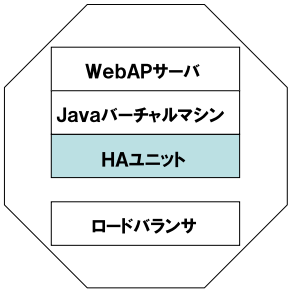


図 4　WebAP ユニットの構造
Fig. 4　Structure of WebAP unit.

トを再利用し，新たに必要なシステム構築ユニットを追加しながら，システムモデルの種類を充実させてきたことが示されている．これはシステムモデルの種類の拡張である．

なお，表 2 のすべてのシステムモデルに共通に使われているのが，HA（High-Availability）ユニットである．HA ユニットはサーバ，ネットワーク，ディスク，アプリケーション等のシステムの構成要素における障害を隠蔽し，動作継続を保証する機能を提供するユニットであり，すべてのユニットの基盤となるユニットである．基本的な考え方は，ハードウェア・ソフトウェアの構成要素を二重化ないし多重化し，障害が発生したときには，障害部分をすみやかに切り離すか，多重化構成された正常な部分へ切り替えることにより，システムの継続動作を可能にするものである[*13]．MUA の考え方では，このような単位で，システムに必要な機能を切り出し，それを組み合わせることにより，システムを構築するのである．OLTP ユニット，WebAP ユニット，DB ユニット等の他のユニットは HA ユニットにさらに必要なソフトウェア等を加えることにより実現されている．したがって，HA ユニットはすべてのシステムモデルで共通的に使われている．

図 4 は WebAP ユニットの構成図である．HA ユニットに HW や SW が追加されて作られている．ここにロードバランサとは，外部からの要求を，同等の機能を持つ複数のサーバに分散して渡す機能を持つ HW ないし SW である．

一方，システムモデルの拡張には，規定の範囲の拡張もある．

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 三層アーキテクチャ[*10]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ともない，同一システムモデルであってもシステム特性が異なるシステムが散見されるようになった（**表 3**）．

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．当初のアプリケーションは，独自の AP-FW 上でそのつど開発することが主体（スクラッチ開発型）であったが，Java の浸透とともに，世間で流通している AP-FW を採用する傾向が強まってきた．このため AP-FW までをシステムモデルに取り込み，規定の範囲とする必要がでてきた．この AP-FW の取り込みが規定の範囲の拡張の一例である．従来のミドルウェア領域までのシステムモデルを Basis と呼び，AP-FW までを含んだシステムモデルを Extension と呼んでいる．

システムモデルの拡張の流れをまとめてみると，以下のようになる．そのイメージを**図 5** に示した．

3 Tier Architecture を起点に，バッチ処理を「収集–加工–蓄積」に分解して，超並列バッチ処理を実現した大規模バッチモデルからスタートし，システム構築ユニットの再利用や追加，超高信頼性モデルにおける HUB 層の追加等アイディアの追加を積み重ね，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．

一方，AP-FW を規定範囲に取り込むことが必要となり，Basis から Extension へと拡張されてきた．昨今は，パッケージ AP の利用が増え，NEC でも 2 年かけて ERP[*14]領

[*13] 本論文は，MUA の考え方に従い，システムに必要な機能を切り出して再利用してゆく手法について説明するものであり，個々のシステム構築ユニット自体の詳細を説明することが目的ではないが，HA ユニットを例にとれば，障害の検出・切替え機能を持ったミドルウェアや，冗長化されたハートビート LAN，冗長化されたストレージが搭載されたサーバ群で構成されている．

[*14] Enterprise Resource Planning

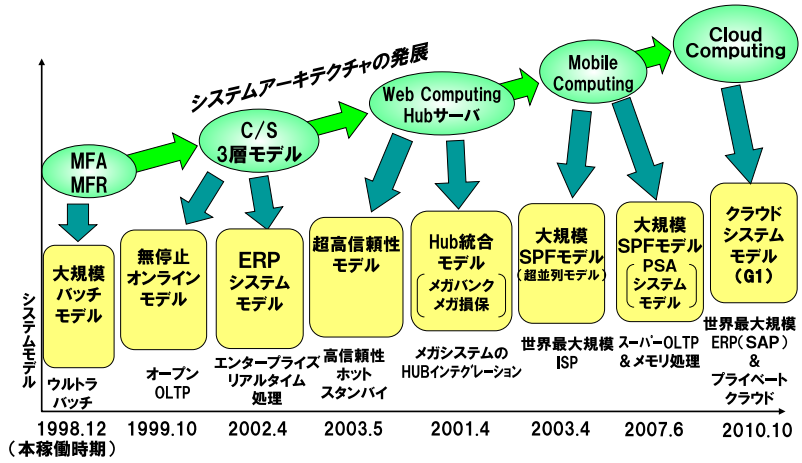


図 6　システムモデルの歴史的変遷

Fig. 6　Historical transition of system model.

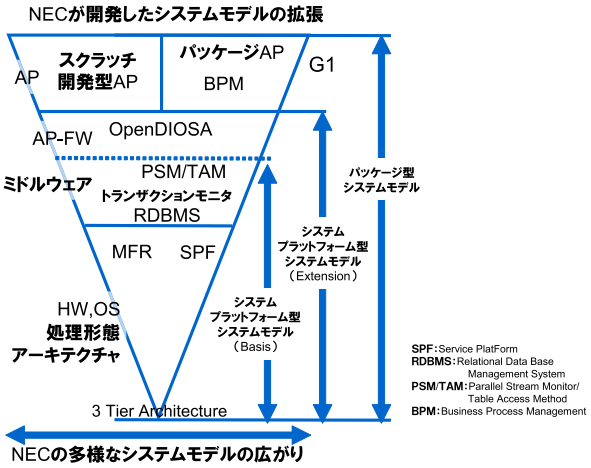


図 5　システムモデルの拡張のコンセプト

Fig. 5　Concept of extensions of system model.

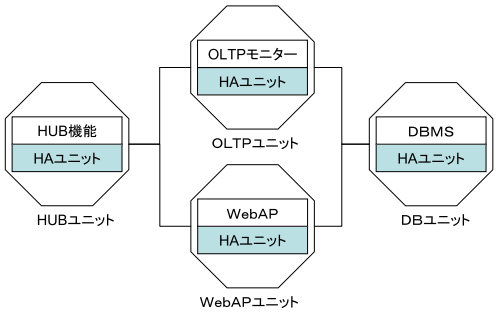


図 7　超高信頼性システムモデル

Fig. 7　Ultra-high-availability system model.

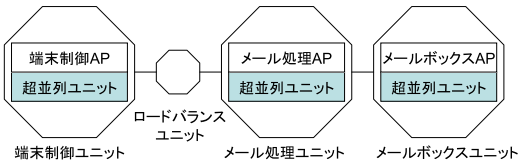


図 8　超並列システムモデル

Fig. 8　Ultra parallel system model.

域の社内基幹システム（G1: Global One）を SAP で構築した．これにともないパッケージを利用する ERP 領域のシステムモデルを規定し，これをパッケージ型と呼称することとした．これも規定の範囲の拡張の一例である．

これまで述べてきたように，新たなシステム要件が出現するたびに，MC 性の確認に立ち返り，必要ならば新しいシステム構築ユニットの追加，ユニット再利用，システムモデルの規定の範囲の拡張を繰り返すことにより，数多くのシステムモデルが作られ，利用されてきた．これらの活動成果を時系列に描いたものが，**図 6** である．

このように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 [13]
  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 [14]
  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 [*15] システムモデル [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 OLTP としてのリアルタイム処理システム基盤

図 7 は，超高信頼性システムモデルのイメージ図であり，**図 8** は，超並列システムモデルのイメージ図，**図 9** は PSA システムモデルのイメージ図である．

[*15] Parallel Stream Architecture

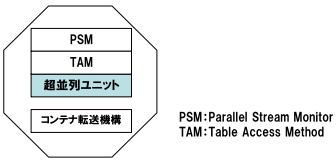


図 9　PSA システムモデル
Fig. 9　PSA system model.

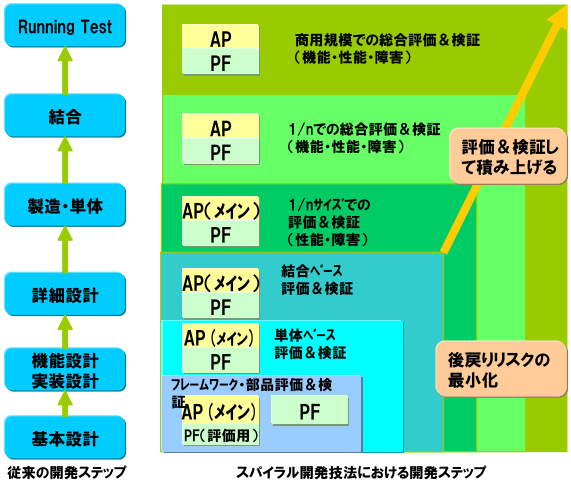


図 10　従来（ウォーターフォール型）と比較したスパイラル開発の開発ステップ
Fig. 10　Development step in water-fall type and spiral type.

4.6　性能問題の早期発見（システム構築ユニットを用いたスパイラル開発手法）

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．システム構築ユニットが，機能単位に構成されており，かつ，サイジングファクタを与えることにより，小さなシステムから大きなシステムまで，相似形にシステムを拡張できることに注意されたい．

この性質を利用して，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の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．**図 10** は従来のウオーターフォール型とスパイラル開発の開発ステップを比較したイメージ図である．

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．当初は相似形の小さな環境からスタートする．
- 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．

5.　システムモデルによるシステムプラットフォーム構築の効果

MC 性を定義したことにより，客先キーマンとの間で基幹システムの要件をきちんと整理し，合意することができ，実績のあるシステム構築ユニットを部品として使用することにより堅牢なシステム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 11 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 3 つの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 構成要素の変更にともな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（設計コストのほか，MC 性検証で大きなコストが発生）

5.1　地銀向けバンキングシステム

OMCS の代表的事例である地銀向けバンキングシステムの事例で効果を工数比較（相対値）で説明する．MC 性に基づく要件定義以降の工程を比較する．

（1）2003 年の最初の地銀向けバンキングシステム

- 実施メンバ　　最大 50 名
- 事前準備期間　　20
- 設計　　20
- 構築　　10
- 評価　　25
- 分析　　25
- 合計　　100

（2）2010 年に構築した地銀向けシステム

このシステムは基本的に，2003 年のシステムの流用で，一部，ユニットの改造を要したケースである．

- 実施メンバ　　最大 20 名
- 事前準備期間　　8　　流用部分の削減効果は大きいが，一部要件に違いがあったことにより増加（完全流用ではなかった）．
- 設計　　8　　一部新たな設計が必要であった部分を含む．
- 構築　　9　　構築自体のコストはあまり変わらなかった．
- 評価　　15　　流用部分の工数削減効果が大きい．
- 分析　　11　　流用部分の工数削減効果が大きい．

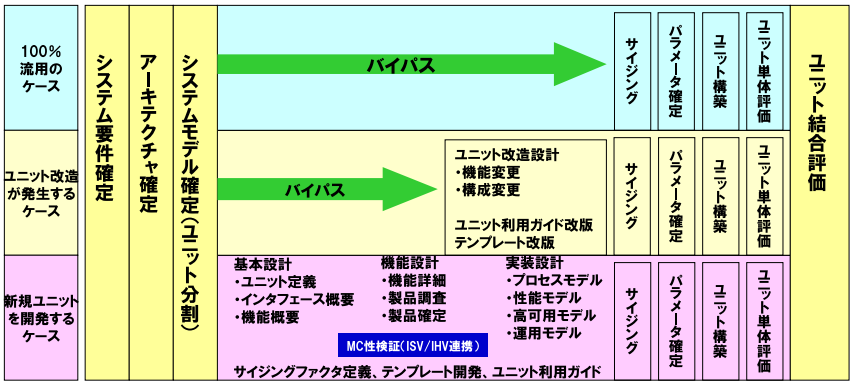


図 11　期間短縮によるコスト削減のイメージ
Fig. 11　Cost reduction by shortening process.

- 合計　51　2003 年のケースの約 1/2.

この事例では，システムプラットフォームの構築工数を約 1/2 に削減できた．

5.2　WebAP ユニット

WebAP ユニットの構築例を示す．昨今のインターネットを介したやりとりは，WWW ブラウザの GUI を利用したインタフェースが主流であり，携帯電話やスマートフォンでの利用形態も例外ではない．その処理を具現化する WebAP ユニットは非常に重要なユニットの 1 つである．

当初作られた WebAP ユニットは，商用 Unix をベースに検証されたが，昨今のシステム価格低減の要求に従い，Linux 版の開発が求められた．その Linux 版の開発時とその後のシステムモデル適用時の工数比較（相対値）を示す．

(1) Linux 版開発時

- 実施メンバ：マネージャ 1 人 + 熟練担当 3 名
- 事前準備期間　12　商用 Unix 版の MC 性要件整理は流用できたので，この工数で済んでいる．Linux 版にするために，製品選定から実施．
- 設計　16
- 構築　12　評価の結果，後戻りがあってやや増加．
- 評価　30　初評価の製品群なので時間がかかっている．
- 分析　30　商用 Unix 版との比較分析が行えたので，この工数で済んでいる．まったく新規の評価の場合にはもっと工数がかかるだろう．
- 合計　100

(2) Linux 版流用時

- 実施メンバ：担当 3 人（マネージャのスポット支援あり）
- 事前準備　0　すでに確立したユニットゆえ，0.
- 設計　0　同上
- 構築　8
- 評価　12
- 分析　12
- 合計　32　Linux 版開発時の約 1/3.

このように，システム構築ユニットの活用で，工数を 1/2 以下に削減することができる．また熟練者を必要としないところも大きな効果である．システム構築ユニットの組合せによりシステムを構築するというシステムモデルの考え方を利用すれば，システム全体で工数が 1/2 に削減できることが期待できる．

5.3　大規模 ISP システムにおけるスパイラル開発の効果

また，スパイラル開発の事例として，大規模 ISP システム（NTT ドコモの i モードシステム）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，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

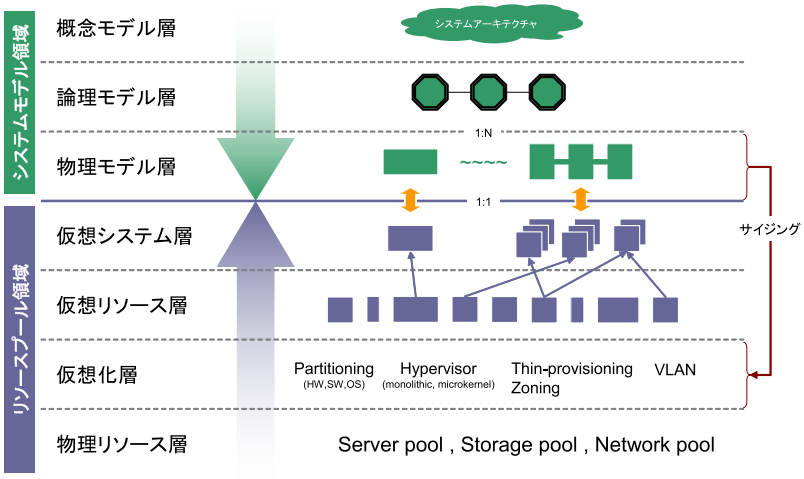


図 12　RPS のシステムアーキテクチャ
Fig. 12　Architecture of RPS.

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すべ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 年というきわめて短期間で実現した．メインフレームやスパイラル開発適用以前の大規模なオープン・システムのプロジェクトでは，通常 2〜3 年の開発期間を要したことと比較し，1/2 以下を達成したといえる．

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．

ただし，この開発期間の大幅削減については，スパイラル開発が大きく貢献することは確かだが，その他の多くの改善が同時に行われたことにより実現されたと見るべきである．それら多くの改善については，参考文献 [15] に詳しい．

6.　システムモデルと仮想化技術との融合

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．クラウドサービスが実現できるようになった背景には，あらかじめ用意したサーバ，ストレージ，ネットワーク等，各ハードウェア機材からダイナミックに必要な資源を確保する仮想化技術が浸透してきたことがある．これらの仮想化技術は一般的なもので，各社の仮想化技術の発表にも類似した技術が紹介されている [16], [17], [18]．

また，参考文献 [17] にも記載があるように，一般に仮想化技術を導入することにより，運用コストを含む各種コストの削減やシステム提供までの期間の削減が期待できるものである．しかし，仮想化技術の導入は，システムの MC 性を担保することとは直接はつながらない．システムの MC 性を担保するためには，システムモデルによるプラットフォーム構築技術が有効である．

本論文での本章の目的は，システムモデルによるプラットフォーム構築技術とこれらの仮想化技術が一体化できることを示すことにある．この一体化により，MC 性を満たした基幹システムの実現と，運用コストを含む各種コストの削減やシステム提供までの期間の削減とを両立させることが狙いである．この狙いの実現のため，仮想化技術とシステムモデルを結び付けたものが，OMCS のリソースプーリングシステム（RPS: Resource Pooling System）である．

6.1　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 2 階層からなる．**図 12** にイメージを示す．

上位層のシステムモデル領域は，4.4 節で述べた 3 つのレベルで層分けをしている．すなわち，レベル 1 の概念モデル層，レベル 2 の論理モデル層，レベル 3 の物理モデル層である．

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 4 つの層からなり，それぞれ下記の役割を持つ．なお，これらの仮想化技術は一般的なものであると考えてよい．

- 物理リソース層：サーバ，ストレージ，ネットワーク等の各ハードウェア機材をあらかじめ準備したもの
- 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
- 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み合わせて構成された仮想化されたシステム

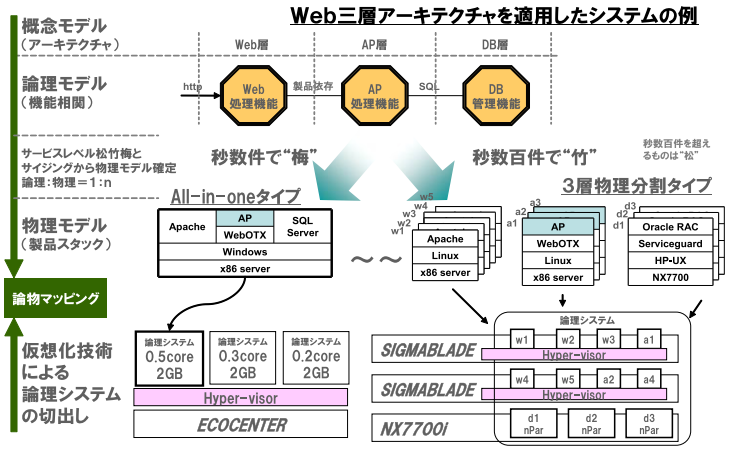


図 13　仮想プールからの切り出しイメージ

Fig. 13　Ensure resources from pool.

6.2　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（現場調整・現地調整）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ダイナミックに切り出し，仮想化されたシステムを生成することになる．すなわち，物理モデル層で確定したサイジング結果を仮想化層への入力情報としてリソースを切り出し，即座にプラットフォーム構築作業にとりかかることができるのである．

Web 三層アーキテクチャを適用した典型的なシステムにおける仮想プールからの切り出しイメージを図 **13** に示す．

一般的に仮想化技術の利用により，あらかじめ用意されたリソースの中から必要なリソースが切り出されるので，短期間にリソースを確保することができるが，OMCS が目指す基幹システムの場合，リソースの切り出しの後，必ずシステムとしての評価を行ってから，システムが提供されることには注意されたい．それでも，従来の単品の SI 作業に比べ，発注・納品・現調等のステップが省略されることから，提供までの期間が大幅に短縮されることは明らかである．

6.3　RPS 適用事例

仮想化技術により実現されたクラウドシステムは，高い MC 性を持つ大規模基幹システムで利用された実績はあまりないと考えられるが，我々のシステムモデルとの融合により，大規模基幹システムの構築にも十分活用できる．4.5 節で触れた SAP ベースの NEC 社内 ERP システム（G1: Global One）は適用事例の 1 つである．このシステムは，大規模基幹システムとしての高い MC 性を満たしていることに加え，仮想化技術の適用から得られる，運用を含めたコスト削減や提供までの期間の短縮といった恩恵を受けることができた．特に，コスト削減効果は，仮想化技術を利用しないケースに比べ，約 27% [*16] の削減効果があった．

7.　まとめ

3 Tier Architecture を起点に，バッチ処理を「収集–加工–蓄積」に分解して，超並列バッチ処理を実現した大規模バッチモデルからスタートし，システム構築ユニットの再利用や追加，超高信頼性モデルにおける HUB 層の追加等アイディアの追加や規定の範囲の拡大を積み重ね，新たなシステム要件に対応するためのシステムモデルを順次創出し，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，NTT ドコモの i モードシステムに代表されるような，世界最大規模のオープン・システムの開発を達成した．システムモデルとそれを構成する MUA（Multi Unit Architecture），なら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．図 6 で示した数多くのシステムの稼動実績[*17]は，本論文で述べたシステムモデルによるプラットフォーム構築手法の有効性を示すものと考える．

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 のシステムモデルが追加されている．

8.　将来への展望

インターネットによるクラウド化の進展，さらにはビッグデータのトレンドを背景として，今後も新たなるシステ

*16　この値自体は，仮想化により一般的に享受できる効果であり，システムモデルによる構築技法と仮想化を組み合わせることにより追加の削減効果があると主張しているものではない．
*17　これらのシステムは現在でも安定稼動を続けている．

ム要件が出現してくることだろう．それにともない，新しいシステムモデルの創出が必要になると同時に，システムモデルの再利用性をいっそう高めるための工夫が必要になるのではないかと考えている．

（1）新たなシステム要件に対応するためのシステムモデルのさらなる拡張

- スマートフォン等の端末の進化にともな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の構築のニーズが新たなシステム要件を生み，新たなシステムモデル創出のきっかけになることが考えられる．
- 現在 Google 等のサービスを支える Web Scale Technology や大規模 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）になると考えており，それが新たなシステムモデルの創出のきっかけになることも考えられる．

（2）システムモデルの再利用性を高める工夫

- システムモデルを標準的に記述するモデル記述法を開発することにより，実システム設計の自動化（ツールの開発）等，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できるのではないかと考えている．

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，いろいろと検討いただいた OMCS 各プロジェクトの皆様に感謝したい．さら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，東　健二，川浦立志：OMCS 構築技術の研究，CDS 研究会，CDS3-11 (2012).
[2] 相澤正俊，富山卓二，斎川幸貴：メモリ DB を活用したスーパー OLTP を実現する OMCS の PSA システムモデル，情報処理学会論文誌　コンシューマ・デバイス&システム（CDS），Vol.2, No.2, pp.40–53 (2012).
[3] Wikipedia, Multitier architecture, 入手先 ⟨http://en.wikipedia.org/wiki/Multitier_architecture⟩.
[4] 秋沢　充，岩渕史彦，前田博之ほか：サービス指向アーキテクチャ適用を成功に導くシステム構築アプローチ，日立評論，Vol.90, No.07, pp.588–589 (2008).
[5] 田中隆一，村井　孝，津田高至ほか：TRIOLE テンプレートによるプラットフォームインテグレーション，FUJITSU.56, 1, pp.16–21 (2005).
[6] 井手ノ上淳：システム基盤 AtlasBase による品質・生産性への取組み，*UNISYS TECHNOLOGY REVIEW*, No.99, pp.65–79 (2009).
[7] 経済産業省：情報システムの信頼性向上に関するガイドライン第 2 版，平成 21 年 3 月 24 日 (2009).
[8] 長谷川正巳，石田英理，小川久範：変化に即応できるインフラ設計手法，*IBM ProVISION*, No.50, pp.68–75 (Summer 2006).
[9] 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，pp.136–145 (1999).
[10] DeMacro, T.: *Controlling Software Projects: Management, Measurement & Estimation*, p.3, Yourdon Press, New York, USA (1982).
[11] 誉田直美：ソフトウェア品質会計—NEC の高品質ソフトウェア開発を支える品質保証技術，日科技連出版社 (2010/06) ISBN-13: 978-4817193483 (2010).
[12] 経済産業省・ソフトウェアメトリクス高度化プロジェクト・プロダクト品質メトリクス WG：システム/ソフトウェア製品の品質要求定義と品質評価のためのメトリクスに関する調査報告書 (Mar. 2011).
[13] 相澤正俊，斎川幸貴，川浦立志：オープンバンキングシステムを実現した OMCS の超高信頼性システムモデル，DICOMO2012-1199 (2012).
[14] 相澤正俊，大藤豊喜，川浦立志：世界最大規模の ISP システムを実現した OMCS の超並列処理システムモデル，DICOMO2012-1203 (2012).
[15] 相澤正俊，上窪真一，小河原孝一：OMCS 構築のプロジェクト管理における見える化，情報処理学会論文誌　コンシューマ・デバイス&システム（CDS），Vol.2, No.2, pp.29–39 (2012).
[16] 松本一志，江崎　裕：クラウド環境におけるダイナミックリソース管理技術，*FUJITSU*.62, 1, pp.14–20 (2011).
[17] 松村真一，清水泰雅，印南雅隆：IT システムの価値創造を支える日立グループの仮想化技術，日立評論，Vol.90, No.07, pp.606–607 (2008).
[18] 立花幸治，浅井保行：MiF における仮想化技術の利用，*UNISYS TECHNOLOGY REVIEW*, No.100, pp.35–44 (May 2009).

商標について

相澤 正俊　（正会員）

1971 年東北大学大学院工学研究科電気通信工学専攻修士課程を修了し，1972 年日本電気株式会社に入社．主に基本ソフト（OS）開発と IT ソリューション事業を担当．2008 年代表取締役執行役員副社長に就任，2011 年より株式会社国際社会経済研究所（NEC グループ会社）理事長に就任．

東 健二　（正会員）

1985 年関西大学大学院工学研究科電子工学特論博士課程前期を修了し，同年日本電気株式会社に入社．OSI 通信系ソフト開発や UNIX 系 OS 開発を担当後，オープン系大規模分散システム開発に従事．2010 年 OMCS 事業本部長に就任．

川浦 立志　（正会員）

1983 年東北大学大学院理学研究科数学専攻修士課程を修了し，日本電気株式会社に入社．主にメインフレーム，ストレージの基本ソフト開発と組込みソフト事業を含む IT ソリューション事業を担当．