

アプリケーション配置アーキテクチャーの効果的な可視化と 効率的なギャップ分析手法

及川 修一 大嶽 隆児 山地 幸子

A Method for Effective Visualization and Efficient Fit-gap Analysis of Application Deployment Architecture

Shuichi Oikawa, Ryuji Ohtake and Sachiko Yamaji

膨大な数の IT システムを所有しているグローバル企業では、プラットフォーム集約化、センター集中化、SOA 化などを目的とした IT 戦略計画のニーズが増加している。これには、将来の姿と現行の姿をアーキテクチャーのモデル図によりギャップ分析することが必要になるが、数百から数千もある現行 IT システムについてアーキテクチャーのモデル図を作成し、ギャップや移行の困難さを可視化することは、容易に実施できない。本論文では、次の 2 点を特徴とするギャップ分析手法を提案している。①現行 IT システムのアプリケーション配置図を星座式パターンで表記して、将来システムのあるべき姿と比較した課題や提言を効果的に可視化する。②可視化した配置図と等価で単純な文字列パターンを使って、現行 IT システムのアプリケーション配置アーキテクチャーを記述して、表計算ソフトにより統計的に効率よく分析する。この手法によって、300 程度のシステムを 2 カ月あまりで効果的かつ効率的に分析することができた。

Global companies that have numerous scattered IT systems need to create a strategic IT planning consolidate IT platform, to centralize the data center, and transform the silo system to a reusable SOA-based system. It is necessary to perform a gap analysis between current and future architecture models; however, it is hard to draw the architecture for each of the hundreds or thousands of current IT systems and to visualize the gaps and difficulties of transformation on the architecture models. This paper proposes the following effective and efficient method: 1) For effective visualization, using "constellation" patterns to describe application deployment diagram. 2) For efficient statistical analysis using spreadsheets, using simple text patterns as the same as the deployment diagrams for current IT systems. Using this method, three hundred systems could be analyzed effectively and efficiently within two months.

Words & Phrases: エンタープライズ・アーキテクチャー, SOA, オペレーショナル・モデル, アプリケーション配置
Enterprise Architecture, SOA, Operational Model, Application Deployment

1. はじめに

多くの企業では、ホスト集約、クライアント・サーバーによるエンドユーザー・コンピューティング、Web コンピューティングなど、多種多様なアプリケーション配置アーキテクチャーからなる多数の現行 IT システムを所有している。特に海外進出している企業が業界再編により合併した場合、規制政策、文化・風土、市場環境、競争環境、立地環境などが地域・極（米州、欧州、アジアなど）で異なる上に進出国ごとに局所最適された情報システムが統合されず、数百から数千もの大小の現行 IT システムが存在していることがある。このような企業では、プ

ラットフォーム集約化、センター集中化、SOA (Service Oriented Architecture) 化を目的とした IT 戦略計画として、エンタープライズ・アーキテクチャー（以下、EA と略す）を策定するニーズが高まっている。

IBM では、将来システムのリファレンス・アーキテクチャーを策定する EA の手法 [1] にオペレーショナル・モデル (Operational Model: 以下 OM と略す) [2] [3] [4] を活用している。OM では、ビジネス、アプリケーション、データ層からの要件を反映させた概念オペレーショナル・モデル (Conceptual OM: 以下 COM) から、製品独立の技術仕様を決定した仕様オペレーショナル・モデル (Specification OM: 以下 SOM)、製品を決定し

提出日: 2007 年 9 月 3 日 再提出日: 2008 年 7 月 4 日

た物理オペレーショナル・モデル（Physical OM: 以下 POM）へ具体化して将来アーキテクチャーを策定する。次のステップとして、現行システムのアプリケーション配置アーキテクチャーと将来アーキテクチャーのギャップ分析を行い、その評価に基づいた移行方針を策定することが求められる。

しかし、実際にギャップ分析を行う場合、次のような課題が存在する。

(1) 移行難易度の可視化

[2] [3] [4] で紹介されている OM の表記法で現行アーキテクチャーを描いた場合、システム接続の複雑さやアプリケーションやデータの地理的分散具合を可視化することはできるが、ギャップ分析結果として求められる将来アーキテクチャーへの移行難易度を可視化することが難しい。

(2) 多数の現行 IT システムの効率的分析

企業全体の方針の策定となると、膨大な現行 IT システム一つひとつについて OM の表記法でアーキテクチャーのモデル図を作成し、将来アーキテクチャーとのギャップ分析することは期間やコストの面で現実的ではな

い。また、モデル図では、移行難易度を統計的に分析することが困難である。

本論文ではこれらの課題を解決するために、配置ユニット（Deployment Unit: 以下 DU と略す）間の接続によりアプリケーション配置アーキテクチャーを可視化した“星座式 DU パターン”の利用と、さらにはそれを文字列化した“文字列式 DU パターン”を提案し、それらを使って効果的かつ効率的にギャップ分析する手法を提案する。

以下、2章で“星座式 DU パターン”を使用した効果的な可視化手法について、3章でさらにそれを文字列化した“文字列化 DU パターン”を使って効率的に整理する手法を紹介する。最後に4章で提案アプローチの実施例を紹介し有効性を述べる。

2. ギャップ分析内容の可視化

2.1 従来 OM による可視化の課題

OM は、ネットワークで接続されたロケーションに設置したノード（実行環境）に DU（システムの構成要素）を配置して地理的に分散したアプリケーション配置アーキテクチャーをモデル化する。例えば、図 1 の SOM^{*1} の例では、クライアント・サーバー・システムと Web ベース・システムが混在しており、支店（L1）のクライアント（SN_支店クライアント）は、支店の DB サーバー（SN_見積 DB）と ODBC で、東京システムセンター（L2）にある Web サーバー（SN_受注アプリケーション）と HTTP で接続されていることをモデル化している。しかし、ユース・ケースの実行に関連する DU 間の接続^{*2} が可視化されていないため読み取りにくい。将来のセンター集中の移行容易性を明確にするためには、ノード間だけでなく DU 間の接続も可視化して、ネットワークやセキュリティの制約の下でどの DU がセンター集中できるか判断する必要がある。また、将来のサービス化への移行容易性を明確にするためには、どのインターフェースがサービス化できるのかを示す必要がある。しかし、図 1 のように、ユーザー・インターフェース（U_見積画面、U_受注登録画面）とシステム間インターフェース（U_価格表送信、U_価格表受信）をどちらも同じプレゼンテーション DU（“U”）で識別しており、明確に判別ができない。

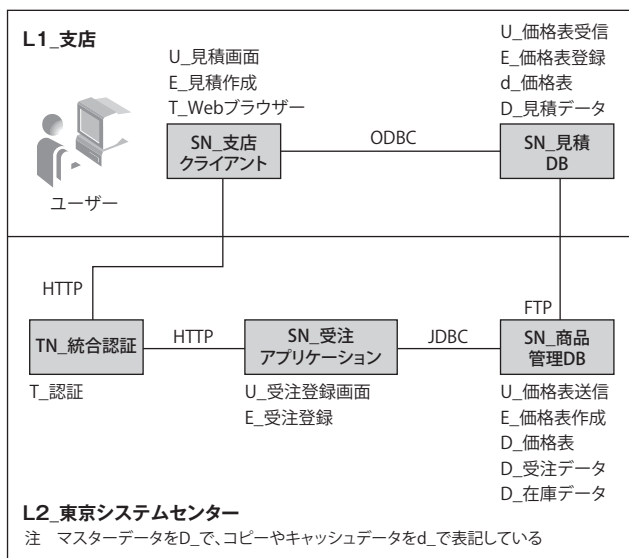


図 1. 仕様オペレーショナル・モデルの例

*1 SOM は、アプリケーション DU とアプリケーションノード、および、それをサポートして非機能要件を実現する技術 DU（“T”）と技術 DU だけが配置される技術ノードによりモデル化される。本論文では、表記が複雑になることを避けるために、技術ノードを示すこと、およびサービス・インターフェース DU（“S”）を追加可能な拡張ポイントを明示的に示すことだけに限定して、技術 DU を可視化している。

*2 OM では、ユース・ケースの実行に関連する DU 間の接続をシーケンス図による動的なモデルで可視化している。しかし、シーケンス図では、全体を見渡すことができないため、本論文では、配置図による静的なモデルで DU 間の接続も可視化している。

表 1. 代表的アプリケーション配置アーキテクチャーにおける星座式 DU パターン表記と評価

アーキテクチャー・パターン例	アプリケーション配置アーキテクチャー		サービス化容易性の評価	センター集中容易性の評価
	星座式DUパターン表記	星座DUパターン表記(サービス化後)		
Webベース (uは動的ダウンロード)			○:ビジネス・ロジック(E)が分散コンポーネントになっており現行システムを改修せずにサービス化できる △:ホスト端末アダプターでラップすることで、現行システムを改修せずサービス化できる	○:パフォーマンスなど非機能要件が許せばバックエンドのノードをセンター集中することが可能
ホスト・ベース				
分散コンポーネント			○:分散コンポーネント(メッセージ/RPC/ストアド・プロシージャ)を利用して現行システムを改修せずにビジネス・ロジック(E)をサービス化できる	○:パフォーマンスなど非機能要件が許せばバックエンドのノードをセンター集中することが可能
クライアント・サーバー (ファット・クライアント)		該当しない	×:ビジネス・ロジック(E)が非分散コンポーネントになっているため現行システム改修が必要となる ×:分散データ・アクセスにより、現行システムのデータ(D)だけは再利用できるが、ビジネス・ロジック(E)はサービスとして再利用できない	△:パフォーマンスなど非機能要件が許せばバックエンドのノードをセンター集中可能だが、ユーザー・インターフェースおよび実行DUが稼働するフロントのノードはシン・クライアント化が前提となる
スタンド・アローン		該当しない	×:ビジネス・ロジック(E)が非分散コンポーネントになっているため現行システム改修が必要となる	×:シン・クライアント化によりセンター集中は可能だが、ユーザー作成プログラム・マクロが中心のためセンター管理の意識が薄い
システム間接続 MQ, RPCなど			○:MQ, RPCなどでやりとりされるメッセージをXMLに置き換えることによりサービス化される	○問題なくセンター集中可能 ロケーションをまたがらない方が高効率
システム間接続 データ・レプリカ, ファイル転送		該当しない	×:分散データ・アクセスによりデータ(D)は再利用可能ではあるが、ビジネス・ロジック(E)が介在しないのでサービス化に適さない	○問題なくセンター集中可能 ロケーションをまたがらない方が高効率
システム間接続 UIのカプセル化			○:画面情報をXMLに置き換えることによりサービス化できる	○問題なくセンター集中可能 ロケーションをまたがらない方が高効率

U...ユーザー・インターフェース DU S...サービス・インターフェース DU X...外部システム・インターフェース DU
E...実行 DU D...データ DU T...技術 DU

2.2 星座式 DU パターンによるアプリケーション配置アーキテクチャーの表現

本論文では将来のセンター集中化やサービス化への移行容易性を判別するために、ユース・ケースの実行に関連する DU 間の接続によりアプリケーション配置アーキテクチャーを可視化する星座式 DU パターンによる表記を提案する(表 1)。サービスとしての再利用容易性を判断するために、プレゼンテーション DU (“U”)を拡張して、外部システム・インターフェース DU (“X”)とサービス・インターフェース DU (“S”)を追加し、ユーザー・インターフェース DU (“U”) ※3 と明確に区別している。これにより、実行 DU (“E”)のインターフェースの配置と、ネットワークやセキュリティの制約の下で技術的にサービス化可能か否かを明確に表せるようにした。

2.3 星座式 DU パターンによるアプリケーション配置アーキテクチャーの評価

各アプリケーション配置アーキテクチャーのサービス化やセンター集中の評価については表 1 に記す。

サービス化による再利用の容易性は、Web ベース・コンピューティングやホストベース・コンピューティングがよ

※ 3 実行時にダウンロードされるプレゼンテーションDUは小文字(“u”)で表現している。

く、ファット・クライアントやクライアント・サーバーでは難しくなる。なぜなら前者ではビジネス・ロジック部である実行 DU が分散コンポーネントであり、改修せずにサービスとして再利用できる可能性が高くなるからであり、後者は実行 DU がクライアント側のノードで非分散コンポーネントとして稼働しており、サービスとして再利用するには改修が必要となる可能性が高いからである。

また、システム間接続パターンにより、既に接続している現行 IT システムのサービス化による再利用の容易性を判断することができる。これはシステム間接続がメッセージ・キューイングや RPC (Remote Procedure Call) などの手段で実現されている場合は、それらを XML にラップすることでサービス化が容易である。なお、現行のシステム間接続数だけを見ても、接続数が少ない方が一般的に他システムへの影響が少なくなり容易性が高くなる。

ただし、本論文でのサービス化やセンター集中の評価

表 2. ノードの抽象化の例

OM レベル	抽象度	DU 数	ノードの抽象化
SOM / LOM	高	少	Unix® / Open 系 Server
POM	↑	↓	AIX® / System p™ HACMP™
物理構成	低	多	AIX5.2 / p690 8 台

は、あくまでも技術的実現性の側面から評価しているにすぎないことに留意する必要がある。業務実行レベルのサービスとしての妥当性の評価については、Service-Oriented Modeling and Architecture (SOMA) [5] など他の手法を適用する。

2.4 可視化に使用するモデル図の抽象度

現行 IT システムの物理構成からそのまま POM を作成すると、抽象化されている将来リファレンス・アーキテクチャーより明らかに複雑であることはわかるが、何が問題なのかを可視化することが難しいことは述べた。現行 IT システムの設計書や物理構成図と将来リファレンス・アーキテクチャーは抽象度が異なり、直接比較できないため現行アーキテクチャーを将来リファレンス・アーキテクチャーと同じ LOM (論理 OM: Logical OM) ※ 4 上に一度抽象化して可視化していく。

既に表 1 で整理されている星座式 DU パターンを使って、関心事のある現行 IT システムに対して LOM を作成して可視化し、将来あるべきリファレンス・アーキテクチャーとギャップ分析した現行の評価や課題・提言を可視化していく。

2.5 集約された論理ノードによる LOM の白地図

IT システムの基盤は、物理構成、POM、SOM/LOM の順に抽象度を上げてタイプに分類することにより、表 2 のようにノードと DU の数を少なくすることができる。

各現行 IT システムに導入されている物理構成を、POM レベルの共通仕様に抽象化した論理ノードとして集約し、集約された論理ノードで作成したノード関連図の白地図 (以下、省略して白地図と呼ぶ) は、IT 標準化が実施されていた場合に最低限必要となる論理ノードを示している。(図 2) 白地図を背景として利用することにより、

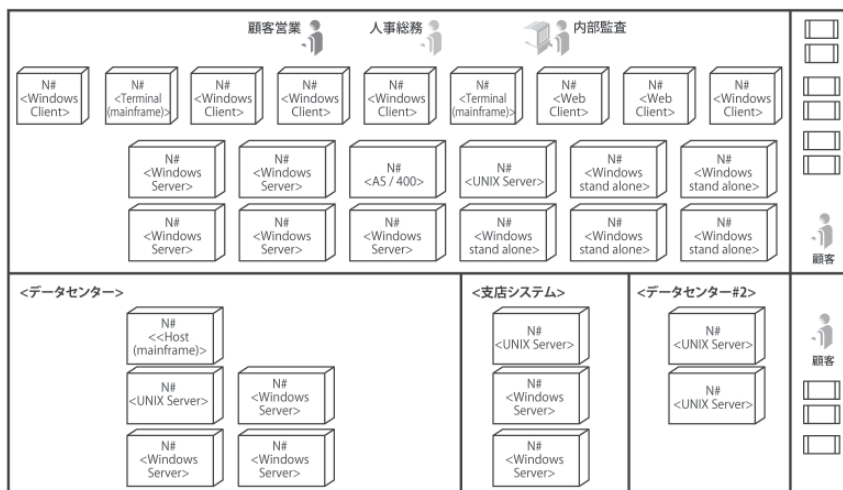


図 2. 論理ノードを使った白地図

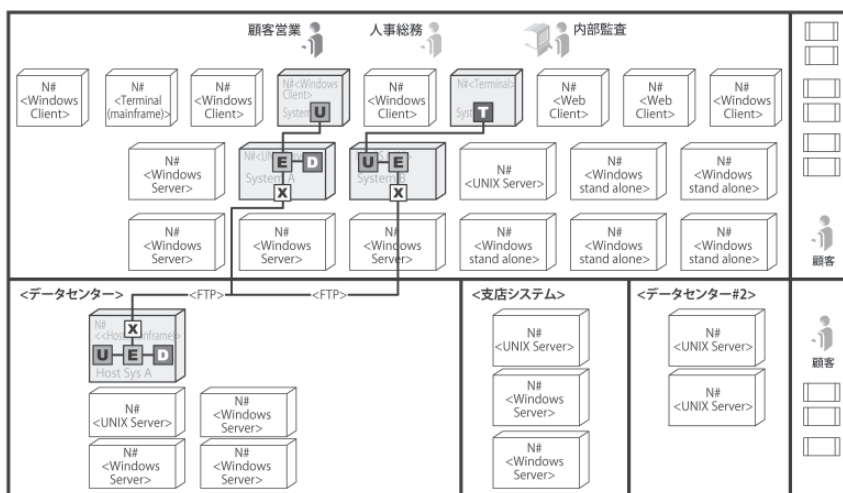


図 3. 白地図上に DU を配置し、接続線で結ぶ

例えば拠点別に LOM を作成する上で、作業効率を高くすることができる。なお、DU およびそのパターンを配置していない白地図だけでも、“技術仕様の異なる論理ノードの種類が多い”、“センターではなくローカルに分散配置されたノードが多い”、など将来リファレンス・アーキテクチャーとのギャップを直感的に可視化することができる。

2.6 星座式 DU パターンによるギャップ分析結果の可視化

代表的な重要業務に関連するシステムのアプリケーション配置アーキテクチャーを判定し、その星座式 DU パターンを使って、白地図上の論理ノードに DU を配置し、DU 間を接続線で結ぶ (図 3、図 4 参照) ことにより、具体的な業務がどのように実行されているかを可視化することができる。この手法により、ステークホルダーに理解しやすい星座のように DU パターンを強調して (星座式

※ 4 LOM は、COM のアプリケーションを稼働させる SOM と異なり、製品独立した技術仕様をモデル・ノードで抽象化した基盤アーキテクチャーである。[6]

表 6. 拠点別評価の例 (数値は筆者によりマスク)

拠点	評価
拠点 A	インフラストラクチャー・アーキテクチャー観点からはサービス化して再利用可能なシステムは全体の XX% である。IT 標準へ対応できているシステムは YY% であり問題は少ない。
拠点 B	サービス化容易と判定されたシステムが X 割にも満たず、使用システム数も少ないため、既存の再利用よりも新規システムへの移行を考えるべきである
拠点 C	システム G は他システムとの直接接続数が ZZ を超えているため、他システムへの影響の観点から移行には慎重を要する。

② 文字列式 DU パターンでの現行アーキテクチャーの整理と分析における表計算ソフトの活用

①では、現行アプリケーション配置アーキテクチャーを星座式 DU パターンとして LOM 上に記述し、モデル図とすることで現行アーキテクチャーの課題を可視化することが可能になる。

また②では現行アプリケーション配置アーキテクチャーを一つひとつモデル図に書き起こすのではなく、OM の DU を活用した文字列パターンで整理し、その分析を効率的に可能にする手法を提案した。この手法によって、企業全体の多数の現行 IT システムの移行方針を包括的に立案する際の材料を短期間で効率的に作成することが可能になる。

参考文献

[1] 林口英治,石田英理: "EA(エンタープライズ・アーキテクチャ)の実践方法と価値について," ProVISION, No.41, pp.67-73 (2004). <http://www.ibm.com/jp/provision/no41/>

[2] R. Youngs: "A standard for architecture description," IBM System Journal, 1, Vol. 38, No. 1, pp.32-50 (1999). <http://www.research.ibm.com/journal/sj/381/youngs.html>

[3] 佐藤浩司: "Webホスティング・インフラストラクチャーのアーキテクチャ検討と基本設計," ProVISION, No.35, pp.46-55 (2002). <http://www.ibm.com/jp/provision/no35/>

[4] 日経SYSTEMS編: "ITアーキテクトのためのシステム設計完全ガイド 2008," 日経BP社, pp.42-47 (2007.8).

[5] Ali Arsanjani: "Service-oriented modeling and architecture -How to identify, specify, and realize services for your SOA-," IBM Developerworks (2004). <http://www.ibm.com/developerworks/webservices/library/ws-soa-design1/>

[6] 大嶽隆児: "表形式テンプレートにより網羅性と一貫性を確保したITシステム基盤アーキテクチャ設計手法," ProVISION, No.47, pp.83-91 (2005). <http://www.ibm.com/jp/provision/no47/>

[7] 長井浩: "情報システムの設計思想 第5回運用基盤的側面のモデル化(オペレーショナル・モデリング)," 日経ITプロフェッショナル, 2004年7月号, pp.148-153 (2004.7).

[8] 山本久好, 榊原彰: "オペレーショナル・モデリングにおける非機能

要求の効果的検証方法," ProVISION, No.41, pp.85-92 (2004). <http://www-6.ibm.com/jp/provision/no41/>



日本アイ・ビー・エム
システムズエンジニアリング株式会社
アーキテクチャー・イノベティブ・
ソリューション
ビジネス・アーキテクチャー ITアーキテクト

及川 修一 Shuichi Oikawa

[プロフィール]

1997年に日本IBM入社。2003年より日本IBMシステムズ・エンジニアリング(株) 出向。ITアーキテクト。JavaおよびWebアプリケーション・サーバーを使ったシステム構築・技術支援を担当の後、EAを活用したSOAベースのインフラストラクチャー・アーキテクチャー策定や技術標準策定を担当。
soikawa@jp.ibm.com



日本アイ・ビー・エム株式会社
エンタープライズ・アーキテクチャー・
アンド・テクノロジー
エグゼクティブITアーキテクト

大嶽 隆児 Ryuhji Ohtake

[プロフィール]

1981年に日本IBM入社。エグゼクティブITアーキテクト。IBM Academy of Technology会員。長野オリンピックの大会情報システム(Info'98)をはじめさまざまな業種の複雑なシステム開発プロジェクトでリードアーキテクトを担当。IBMのITアーキテクト研修講師も兼任。
ohtakro@jp.ibm.com



日本アイ・ビー・エム
システムズ・エンジニアリング株式会社
ビジネス・インフラストラクチャー・ソリューション
WebインフラストラクチャーITスペシャリスト

山地 幸子 Sachiko Yamaji

[プロフィール]

2003年に日本IBMシステムズ・エンジニアリング株式会社に入社。ITスペシャリスト。IBM Tivoli® Security製品の技術サポートを経て、EAを活用したSOAベースのインフラストラクチャー・アーキテクチャー策定するプロジェクトに参画。現在はWebインフラストラクチャーに関する技術サポートを担当。
yamajis@jp.ibm.com