メタファーからアナロジーへ：アーキテクト像の変遷

Metaphor to Analogy : Trace the History of Expected Architect

羽 田 昭 裕

要 約 近年，エンタープライズ・アーキテクチャ，サービス指向アーキテクチャ，モデル駆動アーキテクチャなどアーキテクチャが関心を集めている．アーキテクチャは本来，アーキテクトの技という意味である．そこでアーキテクトとはどういうものか，建築のアナロジーはソフトウェア・アーキテクチャにどのように適用できるかについて述べる．そのうえで，本特集の各論文の位置づけを紹介する．

Abstract In recent years, “Architecture”, such as Enterprise Architecture, Service Oriented Architecture, and Model Driven Architecture, becomes zeitgeist. Originally architecture means an architect's techniques and art. With describing topics of this technology review, this paper tries to show how an analogy of “architect” is applicable to software architecture.

1. は じ め に

メタファー（Metaphor，隠喩）は新たな知識を形成する際に使われる思考方法である．コンピュータ科学やソフトウェア工学は，比較的歴史の浅い工学分野であるため，他分野の成果をメタファーとして用いることが多い．ところで，メタファーは基底となるカテゴリ（例えば，製造業）の知識の特定の局面が強調されるだけであるが，アナロジー（analogy，類推）は基底と目標となる知識カテゴリ（ここではソフトウェア開発）との間に構造的な写像関係を持つ[*1]．例えば，プロジェクト・マネジメントは，知識体系（Project Management Body of Knowledge）が明確になることによって，プロジェクト・マネジメントの役割関係や基準などについて建築業界と IT 業界のアナロジーが成り立つようになっている．

“ソフトウェア工学”という用語は製造業における工学のメタファーを用いている．ソフトウェアの開発についてのもうひとつのメタファーは，建築業のメタファーである．今回の特集号のテーマである，アーキテクチャはまさにこのメタファーを採用している．このメタファーも古くからあり，IBM のシステム 360 および OS/360 のプロジェクトを指導したブルックスも『銀の弾丸などない』[1]で，1958 年に友人からこのメタファーを得て，「彼は一瞬にしてソフトウェア作成プロセスに関する私の見方全体を広げてくれた」と述べている．最近では，アーキテクトの重要性の認識が強まり，その役割や育成方法が課題になっている．そこで，“アーキテクト”をメタファーからアナロジーに発展させることによって，この課題に回答しようという試みが始まっている．

アーキテクト像の変遷をたどり，現在期待されるアーキテクト像の仮説を述べた上で，本特集号の論文を紹介する．

2. アーキテクト像の変遷

コンピュータ技術に関する関心は，ハードウェアからソフトウェア，そしてサービスへと移ってきている．ここでは，時代の流れに沿ってアーキテクト像がどのように変遷したかを振り

返り，今後のアーキテクト像についての仮説を示す．

2.1 ハードウェアのアーキテクチャ

アーキテクチャという言葉は，紀元前 6 世紀には，ギリシャにおいて成立していたと考えられている．その意味は，“アーキテクト（原理を知る工匠）の技”であり，原理とは“秩序”であった[2]．

このように歴史的には，“アーキテクチャ”という言葉の用法は，“アーキテクト”や“原理・秩序”が何であるかに依存している．Brooks[3]では，アーキテクト（技術主任）の役割を次のように定義している．

> 「では，技術主任はどうだろうか．開発するデザインを考え，下位部分を決定し，外側からどのようにみえるか記述し，内部構造を描く．デザイン全体の統一性や完全性を提供して，複雑さに歯止めをかける．技術的な問題が発生したら，個別にその解決策を考え出すか，あるいは必要に応じてシステムデザインを変える．…彼のコミュニケーションは，主としてチーム内に向けられる．その仕事は完全に技術的なことに終始すると言ってよい」

すなわち，クライアント（施主）の代理人としてのアーキテクトというメタファーを出発点として，代理人としてユーザから見たシステムの一貫性を定義する．そして，そのコミュニケーションの相手はクライアントではなく，チーム内に向けられる．

Baldwin and Clark[4]は，F.P. ブルックスの功績も踏まえ，「モジュール化」を“秩序”として，ENIAC から始まり，システム/360, UNIX にいたる歴史を考察している*2．その主張を，アーキテクチャという観点から要約すると次のようになる*3．システムは構造と機能からなり，システムの価値は機能に由来する．アーキテクチャは，構造の独立性と機能の統合性とを成り立たせる枠組みだが，アーキテクトの主な対象は構造である．構造には，設計されたシステムの構造と，その設計を進めるためのタスク構造がある．設計は選択の連続であるが，それは設計とタスクに内在する階層性や相互依存性を深く理解することにより，価値を高めることができる．逆に，アーキテクトの見識が，タスク遂行に耐えない場合，統合に失敗する，としている．その例証として次のものを挙げている．コンピュータ（ハードウェア）においては，1960 年代にシステム/360 が初のモジュール化を実現することで，市場を制覇した．しかし，オペレーティング・システム（OS）に関する相互依存性に関する明確な見識が十分に蓄積されていなかったため，OS/360 の設計についてはモジュール化を達成できなかった[4]．その後，相互依存性についての蓄積に基づき，高度なモジュール性を実現した OS が UNIX　である．

このようにコンピュータのハードウェアや OS は，モジュール性を秩序としたアーキテクチャが実現されている．Brooks[3]は，OS/360 での経験を例に，大規模ソフトウェア・プログラムの開発プロジェクトの管理について考察している．しかし，Baldwin and Clark[4]も述べているように OS 以外の大規模ソフトウェア・プログラムの開発においては，モジュール化は実現可能な目標とは認識されていない．そこで，以下では，OS 及び仮想マシン（Virtual Machine）を含まないソフトウェアについて検討する．ソフトウェアにおけるアーキテクチャは，どのような性質を持つのだろうか．

2.2 ソフトウェアのアーキテクチャ

ギリシャの建築思想を集大成したのがウィトルウィウスの建築論である[2]．ウィトルウィウ

スの建築論は，強さ（firmitas）・用（utilitas）・美（vernustas）の三つを立脚点とし，それぞれ現代の建築論では，構造学，プランニング，造形理論に相当している．

スウェルは，このウィトルウィウスの建築論のアナロジーとして，ソフトウェア・アーキテクト論を構成している．Sewell[5]は，ウィトルウィウスの建築論との対応と，ソフトウェア開発における役割を次のように述べている（図 1）．“用”はニーズのことでありクライアント（発注者）の役割である．“強さ”は，構造のことであり，システムの堅牢性や信頼性の最適化はエンジニアの役割である．“美”は，デザインのことであり，アーキテクトの役割である．つまり，アーキテクトは，クライアントと構築者の橋渡し役であり，「ソフトウェアのデザインを通じて，ニーズに対応し，問題を解決し，有用性を向上させる」[2]ことが目標となる．

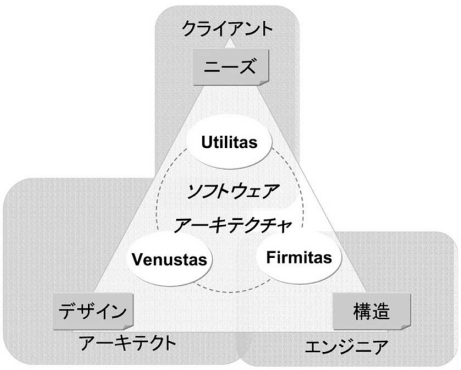

図 1　ウィトルウィウスの三角形とソフトウェア・アーキテクチャ

Sewell[5]の議論を敷衍すると，同じ建築のメタファーを用いているが Brooks[3]の “技術主任”はエンジニアに近く，本来のアーキテクトはクライアントとのコミュニケーションも主な役割となる．このような観点は，可変性や不可視性などのソフトウェアの独特な性格[*4]がクローズアップされた 1980 年代のさまざまな提案の方向性と一致する．例えば，ソフトウェアに対応する概念構造体の図式によるモデル化[*5]，ユーザ視点でモデル化[*6]，ユーザ中心の設計[*7]などである．後者の二つはユーザ視点により，可変性を扱えるようにしようという方向を示している．そのためには，前者の二つのようにモデル化とネットワークなどの位相的な表現によって，アーキテクチャを可視化し，複雑性を抑え，同調性の影響を少なくしようとしている．

2.2　サービスのアーキテクチャ

現在，企業の IT がバックオフィスや業務の IT 化から，組織全体の最適化へと進み，さらに企業の壁を超えた適用へと進むなかで，アーキテクチャの対象はソフトウェア領域だけではなく，企業におけるビジネスの構造や機能を明確に理解するために利用されるようになった．

モデリング技術の進化によりビジネスと IT の双方の関係者が共有できる全体像の可視化ができるようになり，さらに Web サービスやアプリケーション統合技術などの分散テクノロジや統合テクノロジの進化，ソフトウェア開発技術の進化が現在のビジネスの特徴である動的，統合化，分散化に対応可能になったことでビジネスと IT の融合が促進されている．このような状況の中，ユーザに視点を置いたモデリング技術の焦点は，ビジネス・アーキテクチャへと変化している．ビジネス・アーキテクチャとは，企業活動を特定の目標を持つシステムとして

捉えたモデルである．

現在，提唱されているソフトウェア・アーキテクチャは，ビジネス・アーキテクチャとの対応を意識している．例えば，MDA (Model Driven Architecture) における CIM (Computation Independent Model)，SOA（Service Oriented Architecture）の Service などがそうである．しかし，そこではアーキテクト像や設計タスクの構造は明示されていない．そこで筆者らは，次のように想定している（図 2)．クライアントの役割は，ビジネス・アーキテクチャを描くことである．エンジニアの役割は，SOA に基づいた技術方針に従って情報システムを構築することである．そしてアーキテクトは，設計図（Business Blueprint）をもとに，クライアントのニーズとエンジニアの技術を結びつける．

3. 各論文の内容

本特集は図 2 のようなサービス指向のアーキテクチャを想定した，アーキテクトの役割に関連する論文を集めている．その内容は，ビジネス・アーキテクチャとの橋渡し（図 2 の①)，エンジニアリングとの橋渡し（図 2 の②)，ビジネスと IT のオペレーショナルな橋渡し（図 2 の③）に分けられる．

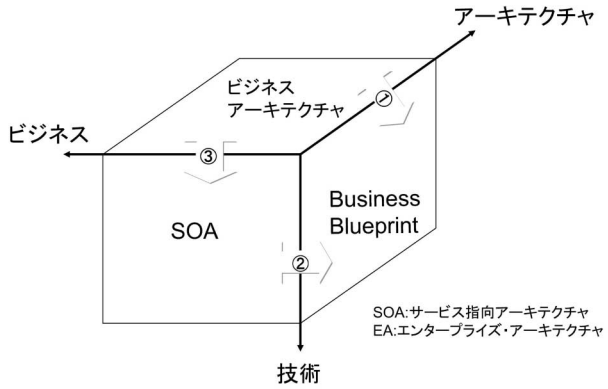

図 2 次世代ソフトウェア・アーキテクチャ

3.1 ビジネス・アーキテクチャとの橋渡し

クライアントとエンジニアの橋渡しとなるソフトウェア・アーキテクチャの役割を企業レベルの観点から見ると，企業の目標や機能を明確にして，それらを実現する複数のシステムの関係や IT 適用方法の指針を示すことにあたる．

企業は特定の目的や目標を持つ複雑なシステムであり，ビジネスはその目標を達成するために，さまざまなリソース間の相互作用によって実施される活動である．ビジネスの改善，革新を推進するためには企業の構造や機能，それらの実現手段である組織やビジネスプロセスを明確にするためにアーキテクチャが利用される．ビジネス領域のアーキテクチャの適用によって環境変化にともなうビジネスの変更要求に対する変更点の特定や対応が迅速になる．現在，ビジネスの実現に IT の影響は大きくなっている．ビジネス領域のアーキテクチャでは，ビジネスの要求を情報システムに正しく反映するためにビジネスと IT の双方の関係者に理解できるモデリングが必要となる．

そのための方法のひとつがEAである．論文「エンタープライズ・アーキテクチャを実現する可視化アプローチ」は，仮想化と可視化という観点からEA構築のポイントを示し，ITガバナンスの強化に果たす役割について触れている．

実際の橋渡しには，ドメイン特有のソフトウェア・アーキテクチャが必要である．論文「組織の学習を支えるアーキテクチャ」は，学習サービスやナレッジ・マネジメントに対する考察を踏まえ，組織の人的資源を有効活用するためのアーキテクチャモデルを提示している．

3.2　サービス指向のエンジニアリングへの橋渡し

これまでビジネス戦略やビジョンをITによって実現する「Concept To Code（概念からコードへ）」の実現を目指してビジネスとITを統一して扱うことができるソフトウェア・アーキテクチャやモデリング技術が提案されてきた．しかし個々のフレームワークやツールに依存した記述やモデルを前提とするため，現実のビジネスにおける要件や要素のモデリングには制約となり，期待する効果が得られなかった．これらの問題点を解決するためにSOAやMDAが期待されているが，現在はその標準化や対応ツールなどの環境が整備されつつある段階である．

論文「MDAと現実の開発プロセス」は，MDAについて概要と開発プロセス，さらに現実の開発においてどのように適用できるかを論述している．

論文「サービス指向アーキテクチャによるビジネス・プロセス統合」は，組織間や企業間を有機的にシステム統合するための指針であるSOAを，ビジネス環境の変化に対応することに加え，変化を知ることによって効果を実現するという目的のために，設計タスクをどのように進めるかという方法を提示している．

論文「LUCINA for .NETによるアーキテクチャ中心のプロジェクト・アプローチ」は，SOAに適したフレームワークとして有望な.NET frameworkをベースに，アーキテクチャの確立を効果的に進めるためのアプローチを提案している．

3.3　ビジネスとITのオペレーショナルな橋渡し

企業レベルのアーキテクチャは継続的にメンテナンスしていくことが必要となる．たとえば，業界における制度や法令の変化にしたがってビジネスとITの仕組みの双方を変更しなければならないことがある．このため企業レベルのアーキテクチャは，日々のビジネス上のオペレーションとITの関係を維持管理する仕組みが必要となる．このような課題に対する回答として，ばらばらな知識空間の相互運用性を実現するために語彙とセマンティックスを定義する，オントロジーという技術が提案されている．

オントロジーを具体的な問題領域の上で実現する技術の一つがXBRL（eXtensible Business Reporting　Language）であり，これは財務情報や事業報告を記述するためのXMLベースの言語である．論文「XBRLの技術動向」は，XBRLの技術的な特徴と，自動的な妥当性検査などの最近の技術などを紹介し，適用の効果と課題について論及している．

オントロジーをより広範囲な問題領域で実現しようとするのがSemantic　Web　である．論文「Semantic　Web：機械によるWeb　コンテンツの解釈と自動処理のための知識処理アーキテクチャ」は，Semantic　Webの背景となる現在のWeb　利用における問題点を，情報の検索，共有，サービスの実行の観点から示し，Semantic Webのアーキテクチャの概要を述べる．

6. お わ り に

アーキテクチャの意味やアーキテクトの役割については，先輩である岩田氏による『技報』の解説「オブジェクト・モデリングへの期待」[9]において詳細に，分かりやすく論じられている．あえて同様のテーマについて記述したのは，この5年間での状況の変化を反映する必要を感じているためである．主な変化は，以下の三点である．まず，現在は情報システムの構築を企業全体のグローバルな観点で捉える情報システム・アーキテクチャ計画が，個別のシステム開発よりも重視される傾向にある．次に，当時はアーキテクトとモデラーの分離が重視されていたが，現在ではその統合が求められている．そして，当時は分散オブジェクト環境での設計の相互依存性について見識が十分に蓄積されていなかったが，現在はその蓄積を反映して，J2EE，.NET, Web　サービスなどビジネス・アプリケーションの基盤となる技術がモジュール化，仮想化されている．

日本ユニシスでは，5年前の特集で集めた“期待”をひとつの契機として，分散オブジェクトに関する経験や技法の体系化を進めることができた．その成果の一部は，その後の「技報」に報告している．今回の特集を念頭において，アーキテクチャ技術の開発・実践に取り組んでいきたい．

＊1 「電流系を水流系の知識を結び付けて理解するのは（アナロジーの）一例で，［水源，水流，水圧，水力，漏水］→［電源，電流，電圧，電力，漏電］のような知識カテゴリー間の構造的な写像関係を利用する」（「メタファーの基本用語」，『言語』，Vol.31，No.8，大修館書店，2002）

＊2 Baldwin and Clark[4]の6章（ENIAC），7章（システム/360，OS/360），13章（UNIX）を参照のこと．このモジュール化というアイデアも，建築家であるアレクサンダーの論考[6]をひとつの源泉としている（Baldwin and Clark[4]，BOX 3–1，邦訳 p.77–78）．

＊3 Baldwin and Clark[4]では，アーキテクトを「ある設計・タスクの構造に関して異なるアプローチの方法を見出し，代替案を評価し，適切にトレードオフを判断できる設計者」（邦訳 p.151）であるとしている．そして，モジュールを「構造的に独立しているが，一緒になって働く大きなシステムの中の単位である」（邦訳 p.75–76）と定義し，この構造性の独立と機能面の統合を可能にする枠組みをアーキテクチャと呼んでいる．

＊4 1987年にIEEE　Computerに掲載されたBrooks[1]も，“技術主任”の有効性については確信を深めているが，ソフトウェアの本質的な特徴は，複雑性・同調性・可変性・不可視性であり，「前もって正確に指定できず，複雑すぎて欠陥なしに構築することができない」（邦訳，p.197）ならば，ソフトウェア作成プロセスに建築というメタファーは有用さを失ってしまったと述べている．

＊5 Brooks[3]も“図式が思考とデザインの支援”に有効であり，「ソフトウェアの概念構造体の多くが，本来位相的であり，それらの関係は空間的・図形的表現できる」（邦訳　p.203）ことを認めた上で，その概念構造体は，「異なる局面を取り出した複数の図式を用意することが必要」（邦訳　p.204）としている．

＊6 1980年代初頭のライフサイクル論争では，ソフトウェアはユーザの変更要求にこたえる必要があるが，変更要求は事前に推定できないため，柔軟な構造が必要であることが強調された．そして，そのためには“要求仕様”の記述とそれを忠実に実現することではなく，ユーザの観点でシステムの“目的”を記述し，それに対応する設計を行うことが重視されてきた．ジャクソンのJSD[7]はその代表である．

＊7 認知心理学者のD.A．ノーマンが進めてきた．Norman[8]によれば，情報をベースにしたテクノロジーは，「目に見えず，形態がない．構造を与えるのは，情報や知識を運ぶ側のものである」（邦訳，p.141）として，自動化（automate）するのではなく情報や選択肢を与えてインフォメイトすることにより“人間の責任と柔軟性を尊重するソフトなテクノロジー”（邦訳　p.325–326）が有効であると述べている．

参考文献
[1] F. P. Jr. Brooks, “No Silver Bullet”, IEEE Computer 20, April, 1987［滝沢ほか訳,「銀の弾などない−本質と偶有」,『人月の神話−狼人間を撃つ銀の弾はない』, 第 16 章, ピアソン, 2002］
[2] 森田慶一, 建築論, 東海大学出版会, 1978.
[3] F. P. Jr. Brooks, “The Mythical Man−Month”, 2 nd ed., Addison−Wesley, 1995［滝沢ほか訳,『人月の神話−狼人間を撃つ銀の弾はない』, ピアソン, 2002］
[4] C. Y.Baldwin, K. B.Clark, “Design Rules, Vol.1 : The Power of Modularity”, MIT Press, 2000.［邦訳：安藤訳, デザイン・ルール, 東洋経済, 2004］
[5] M. T.Sewell, L. M.Sewell, “The Software Architect’s Profession : An Introduction”, Prentice Hall DTR, 2001.［邦訳：倉骨訳, マーク・スウェル他著, 職業としてのソフトウェア・アーキテクト, ピアソン, 2002］
[6] C.Alexander, “Notes on the Synthesis of Form”, Cambridge, Mass, 1964.［稲葉訳,『形の合成に関するノート』, 鹿島出版会, 1978］
[7] M.Jackson, “System Development”, Prentice−Hall, 1983.［山崎, 大野訳, システム開発−JSD 法−, 共立, 1989］
[8] D.A.Norman, “Things that make us smart ? Defending Human Attributes in the Age of the Machine−”, 1993,Addison−Wesley.［邦訳：佐伯監訳,『人を賢くする道具 ソフトテクノロジーの心理学』, 1996, 新陽社］
[9] 岩田裕道,「オブジェクト・モデリングへの招待」, ユニシス技報 60 号, 1999, 日本ユニシス.

執筆者紹介　羽　田　昭　裕（Akihiro Hada）
1984 年一橋大学卒業. 同年日本ユニシス(株)入社. 意思決定支援ソフトウェアの開発・適用に従事. その後, 業務システムとその基盤の要求分析・開発に従事. 現在はテクノロジ・イノベーション・オフィスに所属し, エンタープライズ・アーキテクチャ, システム開発方法論など IT ガバナンスを支援する IT コンサルティングに従事. 情報処理学会会員. 著書『競争優位のビジネスプロセス』(共著, 中央経済社　2003).