2012年度　第 8 回特別講義　レポート

日時	2013 年 1 月 18 日（金）　10:00～12:00
会場	日本科学技術連盟・東高円寺ビル　2 階講堂
テーマ	「IT アーキテクチャ設計と IT アーキテクトの着眼点」
講師名・所属	榊原　彰氏（日本アイ・ビー・エム株式会社）
司会	SQiP 研究会第 3 分科会主査　細川　宣啓氏（日本アイ・ビー・エム株式会社）
アジェンダ	1. アーキテクト、アーキテクチャそしてアーキテクティング 2. IT アーキテクチャの構築 3. IT ソリューションの構築手法決定 4. IT ソリューションの最適化 5. ビジネス戦略・課題の領域から IT　領域まで 6. ニュー・パラダイムへの挑戦
アブストラクト	本講義では IT アーキテクチャ設計の基本からはじめ、アーキテクトの思考法、使いこなすべきテクニックを解説する。また、IT アーキテクチャ設計がより効力を発揮するために、IT の視点のみならずビジネス的な観点からの最適化、講演者が現在取り組んでいるスマーター・シティ施策におけるアーキテクチャ設計の位置づけまでを論ずる。

＜講義の要約＞

◆アーキテクト、アーキテクチャそしてアーキテクティング

アーキテクチャとは、「コンポーネントを統合したシステムの基本的な編成、コンポーネント相互およびコンポーネントと環境間の関係、そしてシステム設計と進化を導く原則(IEEE Std.1471-2000)」である。つまり、全体をどう切り分け、分けたものをどう関係付けるか、という「分け方とつなぎ方」である。

ソフトウェアやシステムで一般的に言われることであるが、アーキテクチャは目に見えないため、アーキテクチャ記述を書くことで目に見えるようにする。アーキテクチャ記述を読みやすくし、物事を明確に共有するために、モデル(共通のルールで表現されたもの)を用いる。

アーキテクトとは、アーキテクティング(アーキテクチャを構築する活動)を実践し、アーキテクチャを構築するプロフェッショナルを指す。
大規模なプロジェクトでは、アプリアーキテクト、インフラアーキテクト、データアーキテクトといった複数のアーキテクトがそれぞれアーキテクチャの意思決定を行う。そしてプロジェクト全体にお

いてアーキテクチャの調整を行うのがリードアーキテクトである。リードアーキテクトはプロジェクトが終わった後もシステムがそのライフサイクルを通じてきちんと動くことに対する責任を持つ。プロジェクトマネージャとシステムに関わるタイムスパンが異なるのである。

ソリューション・アーキテクチャとは個々のアーキテクチャの集合であり、ビジネス・アーキテクチャ、アプリケーション・アーキテクチャ、配置(データサービス/アプリケーション)、インフラストラクチャ・アーキテクチャ等が含まれる。また例えば企業社内において複数のソリューション・アーキテクチャがつながりを持つ様に全社的にまとめたものがエンタープライズ・アーキテクチャである。

ステークホルダーとは、「システムに関して興味または関心事を有する個人、チーム、組織(IEEE Std.1471-2000)」である。アーキテクトがアーキテクチャ記述する事でステークホルダーの関心事を根拠付ける事が出来る。関心事を細かく分けるとビューポイント(特定の視点)になる。ビューポイントをもとにビューを作成する。

変更の難しいアーキテクチャに関しては開発の初期段階で意思決定を行う必要がある。しかし最初の時点では実際のシステムは存在せず、少ない情報の中でアーキテクチャ上の意思決定を行わなくてはならない。そのために過去の情報収集やパイロットプロジェクトでの検証が必要となる。また、決められたチェックポイントで再評価を行い、いかに段階的・継続的にリスクを減らしていくかを考える。

IT アーキテクトが押さえるべき 4 つのアプローチは以下である。
・IT アーキテクチャの構築
・IT ソリューションの構築手法決定
・IT ソリューションの最適化
・ビジネス戦略・課題の領域から IT 領域まで

◆IT アーキテクチャの構築
ステークホルダーに対し対象システムの本質を表現し理解させるためには、複数の適切な視点(ビューに基づく表現)が必要となる。

ビューとは、「関連する一組の関心事の視点からシステム全体を表現したもの(IEEE Std.1471-2000)」であり、ビューポイントとは「ビューを作成し、使用するための約束事の詳細仕様(IEEE Std.1471-2000)」である。また、ビューポイントとは「ビューの目的や利用者を明確にして、それぞれのビューを作成するためのパターンもしくはテンプレートであり、それらを作成し分析するためのテクニック(IEEE Std.1471-2000)」である。

一般的にビューに基づいてダイアグラムが描かれる。しかしこのダイアグラムそのものがモデルではなく、ダイアグラムで表されるものがモデルである。複数のダイアグラムで一つのモデルを表すことも、一つのダイアグラムが複数のモデルを表すこともある。

ビューポイントには要求ビューポイント、機能ビューポイントといった、独立した基本ビューポイントと、セキュリティービューポイント、パフォーマンスビューポイントといった、基本ビューポイント全体に対する横断的ビューポイント(パースペクティブ)がある。基本ビューポイントと横断的ビューポイントの組み合わせにより必要な成果物を決めて作成に取り掛かる。

アーキテクチャの構築対象となるシステムの特徴により、作業時に重点をおくところは異なる。従ってアーキテクチャ記述で重要なのは、メリハリをつけたアプローチである。具体的には、既知のベストプラクティスの適用や未経験の領域への挑戦、先進技術の適用などには重点を置く必要がある。例えば新しい技術を取り入れるならばどこでリスクを解消するか、ということも記述する。また、採用した技術についてその選択根拠を記載することもある。

アーキテクチャは多様な側面から設計する必要がある。例えば家を建てるための設計図は平面図や外観図など複数必要になる。システムも同じである。全体をいかに「切り分け」て、いかに「組み合わせ」るかが重要となる。また切り分けることで各アーキテクトに設計を割り振ることが出来る。アプリケーションアーキテクトは機能モデル、インフラアーキテクトは配置モデル、データアーキテクトはデータモデルを成果物(ワークプロダクト)として作成する責任を持つ。これらのモデルをまとめ、リードアーキテクトは成果物としてアーキテクチャ評価、アーキテクチャ上の判断、アーキテクチャ概要、アーキテクチャ構想実証、ソフトウェアアーキテクチャ文書などを作成する。

アーキテクチャの機能的側面を表現するときにはコンポーネントをベースとして考えると良い。コンポーネント設計においては、いくつかの重要な設計原則があり、Open-Closed Principle(「(モジュールは)拡張に関してして開いており、修正に関して閉じていなければならない」という Bertrand Meyer 氏が提唱した原理)や、GRASP(General Responsibility Assignment Software Patterns:汎用責務割り当てパターン(5 つの基本パターンと 4 つの応用パターンを定義している))などが挙げられる。

IT アーキテクチャ設計の手順としては、まず概念モデルから設計し、詳細モデルへ、そして物理モデルへと落としこんでいく。それぞれ機能的側面と稼働的側面があり、両者に対し NFR(非機能要求)の考慮が必要になる。
アーキテクチャ設計プロセスはフェーズに分け、フェーズごとにタスクがあり、アクティビティを行う際は常にタスクを参照する。タスクはロールにより実行され、ロールは成果物の作成を行い、成果物に対し責任を負う。各タスクでの記述内容としては、目的、ロール(主担当と副担当)、入力、出力、ステップ(具体的に行う事)、アーキテクトのロール(詳細な役割)等が挙げられる。

アクティビティには要件定義アクティビティとアーキテクチャ作成アクティビティ等がある。

例えば、予約システムを対象として具体的に各アクティビティを説明すると以下の様になる。
・ビジネスエンティティ・モデルを用いてビジネスプロセス・モデルを描き、ビジネスルールを明らかにする
・システムコンテキスト図を描き、利用者と対象システムが繋がる外部システムを明らかにする
・ユースケース図を描いたうえでユースケース記述を行い、機能要求と非機能要求を書き出していく
・アーキテクチャ概要図とアーキテクチャ上の意思決定(課題に対する候補と選択された案および案を正当とする理由)の記述を行う
・BCE 分析(Boundary(境界)Control(制御)Entity(エンティティ)で責務を識別)でラフにコンポーネントを導出する
・コンポーネントをアーキテクチャ・パターンへマッピングする

アーキテクチャ構成の基本原則として以下の 4 つの事項が挙げられる。
・機能要求のモデル化(コンポーネントベースでの分析・設計)
・テクノロジー・セレクションのためのモデル化(インフラ設計の抽象化)
・コンポーネントをどのようにノードに配置するか(配置実装)
・関心事の分離(Separation of Concerns)

アーキテクチャの稼働インフラ側面を表現する際に、インフラ設計の抽象化を行う。プロキシやサーバ、クライアント等はノードと呼ばれる。ノードはロケーション(本社、視点、データセンターといったものが含まれる)に配置される。ロケーションにはゾーニング(内部ネットワーク、外部ネットワークなどといったゾーン分け)がある。ノード間にはコネクション(二重化させる、迂回させる、などのつなぎ方)がある。
ノードに対して、機能的責務をもたせる。その後、ノード間をつなぐためにシステム上必要なテクニカルノード(ファイアーウォールやルータ等)が必要になる。テクニカルノードは詳細化する時点で追加すればよい。

このような稼働インフラ・アーキテクチャを表現するモデルとして、オペレーショナル・モデルがある。オペレーショナル・モデルとは、システムのユーザーや関連する外部システムとシステムの構成要素及びその配置や相互の関係を表現した、新システムの全体像を示すものである。
オペレーショナル・モデルには、静的な関係(システムの構成要素の配置)を示すノード構成図と、動的な関係(システムの構成要素間の協調)を示すエンド・トゥ・エンドでのメッセージ検証図がある。エンド・トゥ・エンドでのメッセージ検証図により、システムの動作を検証する。例えば「ここでこの程度時間がかかるから相当のハイバッファが必要だ」という事を検証する。

このようにすべての設計に対し、筋道をたてて根拠を示すことが、アーキテクチャ構築の最重要課題である。言い換えればアカウンタビリティ(説明責任)を重要視するのである。
構築したアーキテクチャは後に評価される必要がある。初期の要求と制約を満たし運用が可能なアーキテクチャであることを確認する。また、ステークホルダーの承認を得るために、客観的な根拠を提示する。

◆IT ソリューションの構築手法決定
開発規模と負荷の相関について説明する。開発規模が小規模であれば負荷の大きさは個人差による影響が大きくなる。大規模の場合は個人差よりも欠陥除去作業による負荷が大きくなり、また開発全体に占める負荷の割合自体も大きくなる。開発規模を上手く小さくするように努力する。きちんとしたコンポーネント分割をしていれば、救いの道がある。

IT 構築は基本的に手探りで進むため、実装終了間近まで要求の実現可能性を確信しづらい。Standish Group ″Chaos Report 2007″によれば、プロジェクト成功率は 28%である。このような背景から、IT ソリューションの構築手法の選択はリスクヘッジの組み込みでもあると言える。リスクヘッジの手法には、プロトタイピング、モデル駆動型開発、反復型開発プロセス、スモールリリースと継続的インテグレーション、ビジネスプロセス・モデリングとシミュレーション等があり、軽減したいリスクに対応した手法を選択することが肝要である。

◆ IT ソリューションの最適化
IT ソリューションの最適化として、IT ソリューション実現のコストや方法の最適化だけでなく、顧客の本質的な要求を解決するためにソリューション自体のパラダイムチェンジを行う場合もある。例えば自販機の在庫管理で説明すると、当初顧客から自販機に缶を補充する際の最適経路を計算するシステムが要求されたとする。このシステムは実現が難しく、改めて顧客の本質的な要求を調査したところ「売り切れによる機会損失を無くしたい」という要求が見えてきた、そこで自販機から在庫情報を自動送信する仕組みを構築することで当初よりも安価でシンプルな IT ソリューションを実現する事が出来た。このように適切なアーキテクチャは、パラダイムを変えて問題を解くソリューションを提供する。

最適なソリューションを生むために必要なものとして、ドメイン知識、実現技術を選択するための知識、問題分析とパターン化の知識(要求工学プラクティス)が挙げられる。
NFR の特徴として、主観的(人により解釈と評価が異なる)、相対的(対象システムの特性により重要な点が異なる)、相互干渉的(一つの NFR を達成しようとすると他の NFR を損じる)ということが挙げられる。扱いが難しいが、システムの重要成功要因として大きな比重を占めている。

◆ビジネス戦略・課題の領域から IT 領域まで
アーキテクチャは、何も IT システムに限った話ではない。
ビジネスと IT ライフサイクルは、表裏一体がよい。ビジネスのミッションを達成するためにシステ

ムを作り、ダメであれば修正する。そういった大きな改善のサイクルがある。ビジネス駆動型でなければならない。アーキテクトはビジネス・アーキテクチャをも把握しておく必要がある。

◆ニュー・パラダイムへの挑戦

新しい技術はアーキテクトが「やろう」と言う責任がある。プロジェクトマネージャは、新しいことへの挑戦を決断しにくい立場にあるからである。

クラウド・コンピューティングはアーキテクチャ設計技術に大きな変化をもたらすであろう。API を備えている PaaS(Platform as a Service)の急成長に伴い、開発スピードが上がる。そのことにより運用のスピードが早くなる必要がある。インフラ構成も、プログラムで書きそのコードを構成管理する時代となった。Infrastructure as Code という技術がある。そして DevOps(開発と運用の一体化)により要求をタイムリーかつ継続的に実現する。DevOps の狙いとしては、開発と運用の連携不足による「老朽化で全面更新」の防止であり、継続的なデリバリーによるシステムコストの平準化である。また、データは時間的にも空間的にも細かな精度で得られるようになり、流動するビッグデータの解析に速さが求められるようになっている。

IT が進みすぎて、企業システムの価値や求められる優先順位が変化している。QCD というキーワードの時代は過ぎ去りしものとなり、フォーカスされなくなっている。近年は Agility(連続的な変化への柔軟かつ迅速な対応)が必要とされてきている。更に将来は Dynamic / Optimization(動的な最適化)が必要になるだろう。

技術の変化に追随しつつ、技術の断片に惑わされない原則を身につけることが、アーキテクトの醍醐味である。

＜質疑応答＞

Q：要求分析の工程では、アーキテクトはどのように関わるのか？

A：可能な限り、アーキテクトが IT における実現を考えながら要求を分析する。

デザインパターン、テクノロジーの他に、対象システムのドメイン知識が頭に入っていなければできない。

Q：リードアーキテクトは、ドメイン知識の他にどのようなスキルが求められるか？

A：ドメイン知識よりも、ビジネスモデルがどのように構築されるか？を考慮する必要がある。

ビジネスモデルを構築するための勉強が必要。

Q：バージョンアップによりアーキテクチャが崩れていくことや、ビジネスのアジリティに追いつけないことがある。新しいものを導入するのが良いかこれまでのものを利用するのが良いか悩むこともある。どうしたらよいか？

A：アプリケーション・システムの特性により最初から満点でいかなければならないものもあるが、多くのものは「最初から100点満点を作らない」という方に向かうと思う。ネットワーク化されることでアップデートが簡単になる。ネットワーク化されるかどうかがカギとなる。

API で保護(システム動作が保証)される必要がある。バックエンドを入れ替えてもプログラムに何の影響もでない、というケースが増えていくと思う。

Q:アーキテクトは成果物に対しフィードバックを受けて評価するということだが、具体的にはどうするのか？
A:ATAM など評価の方法は様々だが、アーキテクチャそのものを評価するものではなく、どう判断したのかを評価する手法もない。また、アーキテクチャそのものを評価しても、その上に載るアプリが機能していなければならない。
「どこでどのような判断をして決定したか」という事に対して、本当にそれで良かったのか？軌道修正の必要があるか？などを評価する。

Q:IT アーキテクトの評価をするガイドラインはあるか？
A:昇進プロセス上では上位のアーキテクトがインタビューを行なって評価するが、実際は業務や成果物を直接目で見る必要がある。また、知識よりも説明責任をきちんと果たせるかが重要である。

Q:ビジネスモデルに対する発想を広げる前に、説明能力を鍛えることが必要か？
A:説明能力が欠如しているときちんと伝えられないため、一定以上の規模をリードすることができない。
ビジネスモデルに対する発想が豊かであるかどうかは、業務経験の多さだと思う。多くの業界で多くの経験を持つほうが良い。

＜講義の感想＞
アーキテクチャ設計とはどんなことをするのか？アーキテクトはどのような役割を担当するのか？今まで理解することが難しかったことをわかりやすく丁寧に解説していただき、理解が深まりました。また、アーキテクトが様々な経験や知識や技術を備える必要があることもよくわかりました。現場を見ていても、根拠を示したうえでステークホルダーが納得するように説明できることは非常に大切なことだと思います。そしてアーキテクチャの構築はアーキテクトに任せておけば良いというものではなく、プロジェクトに関わるすべての人がもっと関心を持つことでよりよいプロジェクトになるのではないだろうか、と感じました。