基盤アーキテクトの役割とは？

ますます高まる基盤アーキテクトの重要性

基盤ソリューション本部
プロフェッショナル IT アーキテクト

伊藤　幸司

ITo Koji
koji-ito@exa-corp.co.jp

昨今のさまざまなハードウェアの急激な進化において、多様化するお客様のニーズにこたえるべく、最適なインフラ環境を提供するシステム設計を行う基盤アーキテクトの重要性が高まっている。

本論では、このような環境における基盤アーキテクトがプロジェクト全工程で果たすべき役割を提案し、基盤アーキテクト不在のシステム構築で実際におきた設計課題を取り上げ、基盤アーキテクトの重要性を説明していく。

また、最後に若手技術者育成の参考として基盤アーキテクトを含めた基盤技術者の育成について説明を加えた。

1. はじめに

昨今のIT技術の潮流であるクラウド、ビッグデータなど基盤を取り巻く環境は、急激な変化をおこしている。加えてモビリティ、ソーシャル技術を活用したITシステムが当たり前の時代になりつつある。

こうした中、製品ありきでお客様に提案し、部分最適の積み上げの結果、全体として必要以上に複雑な構成になってしまい、パフォーマンス面など保守・運用上に課題を抱えてしまうケースが散見される。

急激に進化を続けるハードウェア環境を生かすも殺すもシステム全体のアーキテクチャによるところが多く、アーキテクトの重要性は高くなっていると感じている。

クラウドや仮想化を利用することで、オペレーティングシステム（Operating System、以下OSとする）の上のミドルウェア（Middleware、以下MWとする）だけを設計して利用するケースも増えてきており、そうした中で運用面でのトラブルが発生する事例も良く目にする。

またここ数年、ハードウェアメーカ側でOSに加えてMWまで導入済みのアプライアンス製品が出荷されており、これらを活用したシステム導入事例も出てきている。

さらに基盤システムを設計するにあたってはクラウドの採用やデータセンタ統合などデータセンタ活用の検討も必要な要素となってきている。

最近は、新規にITシステムを一から作るケースはほとんどなく、システム統廃合、システムリプレースといったケースが多くなってきている。

このような状況下で基盤技術者にとって、自分たちが従来蓄えてきた技術も大きな進化の前では、なすすべもないのであろうか？

また、基盤アーキテクトの果たすべき役割は何か？　あるいはクラウドやアプライアンス製品でシステムを構築すると基盤技術者が対応することは、なくなっていくのか？

実は果たさなければいけないことが実はまだまだ多いということを案外気づいていないかもしれないのである。

次章以降で、基盤アーキテクトの役割とその重要性について事例を取り上げながら説明する。

2. 基盤アーキテクトの役割

ここでは、基盤アーキテクトの技術要素、一般的に認識されている役割、あるべき役割について説明する。

2.1. アーキテクトの定義と技術要素

アーキテクトはそもそもどう定義されているのか、ここでは基盤だけでなくIT全体のアーキテクトについて説明するとともに基盤アーキテクトの技術要素について説明する。

ITアーキテクチャを技術領域で階層化すると図1のようになる。（ITスキル標準V3　IPA（独立行政法人 情報処理推進機構）[1] 参考）

図1　アーキテクチャの階層構造

ITILスキル標準では、以下の内容でアーキテクチャを定義している。

アプリケーション層は、ビジネスおよびIT上の課題を分析し、機能要件として再構成する。機能属性、仕様を明らかにし、アプリケーションアーキテクチャ（アプリケーションコンポーネント構造、論理データ構造など）を設計する。設計したアーキテクチャがビジネスおよびIT上の課題に対するソリューションを構成することを確認するとともに、後続の開発、導入が可能であることを確認する。

インテグレーション層は、全体最適の観点から異種あるいは複数の情報システム間の統合および連携要求を分析し、統合および連携要件として再構成する。統合および連携仕様を明らかにし、インテグレーションアーキテクチャ（フレームワーク構造およびインタオペラビリティ）を設計する。設計したアーキテクチャが統合および連携要求を満たすことを確認するとともに、後続の開発、導入が可能であることを確認する。

基盤層は、ビジネスおよびIT上の課題を分析し、システム基盤要件として再構成する。システム属性、仕様を明らかにし、インフラストラクチャアーキテクチャ（システムマネジメント、セキュリティ、ネットワーク、プラットフォームなど）を設計する。設計したアーキテクチャがビジネスおよびIT上の課題に対するソリューションを構成す

ることを確認するとともに、後続の開発、導入が可能であることを確認する。

基盤層は、技術要素範囲が広く技術要素で図２のように細かくアーキテクチャを分解する必要がある。

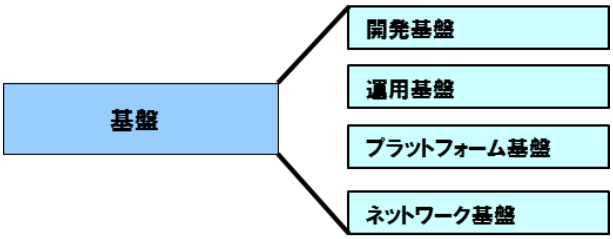

図２　基盤層のアーキテクチャ分解図

表１に各基盤に関する技術要素を紹介する。

表１　基盤技術要素

開発基盤	WebServer/ApplicationServer、WebSphereMQ
	RDB(Relational Database)
	帳票、トランザクションモニターなど
運用基盤	監視、バックアップ、自動化
	システムマネジメント
	スケジュール管理、リリース管理など
プラットフォーム基盤	ハードウェア(Server/Storage/ネットワーク)
	仮想化,OS、高可用性、クラスターなど
ネットワーク基盤	ネットワークシステム(Mail,DNS,FTPなど)
	FireWall、負荷分散装置、セキュリティなど

表１に示すように基盤の技術要素は幅広く、１人ですべての技術領域を対応するのは難しいが、自分の専門領域を中心に全体像を描き、不足する技術領域は各分野のスペシャリストの支援を受けながら対応する。

システム規模によっては、基盤アーキテクトと個別の専門スペシャリスト複数名で構成する体制が必要となるケースもある。

2.2.　基盤アーキテクトの役割

基盤アーキテクトは、システムを作り上げる中で、基盤に対する種々の要件を整理し、

・機能要件に応じたグランドデザイン

・非機能要件定義

を策定する役割を担う。

以下に具体的に説明する。

(1) 機能要件整理

システムの利用目的、なぜそのシステムが必要なのかという背景を明確にし、アプリ要件を踏まえたシステム概念図を作成する。併せて、データの流れ、通信の流れの概略を可視化する。

(2)非機能要件整理

機能要件を満足しても「遅くて使えない」、「アクセスが増えると遅くなる」という事態にならないように性能要件を数値化して定義しておく必要がある。その際、チューニングポイントなど、あらかじめ検討課題を整理しておく。

その他信頼性要件、拡張性要件、セキュリティ要件、検証機・開発機要件などを明確にする。

(3)基盤製品選定

(1)、(2)を踏まえ、具体的にハードウェア、OS から MW 製品を選定してシステム設計を行う。

ポイントとしては、

・アプリケーションの動作保証制限

・構築実績

・システム予算

などを考慮して製品選定を行い、システムを具現化する。

(4)運用設計

さらに、運用面を配慮した運用ツールの製品選定を行い、要件に応じた運用システムも設計する。既に運用システムが共通基盤として提供されているケースもあれば、システム個別に運用システムを作るケースもある。

(5)テスト支援

システム構築後の負荷テスト・障害テストの結果を受けてチューニングする必要がある場合、状況によって基盤アーキテクトが支援を行う。

2.3.　基盤アーキテクトのあるべき役割

ここでは、基盤アーキテクトが、プロジェクト局面にフォーカスして、どのような役割として動けば良いのかを説明する。

一般的にアーキテクトは、前述のように提案段階から要件定義・外部設計フェーズまでは、プロジェクトにかかわり、その後の開発・導入フェーズでは、フォローを中心に対応していけば良いといわれている。しかし筆者は運用フェーズも含めて最後まで責任を負ってプロジェクトにかかわるべきであると考える。(図3)

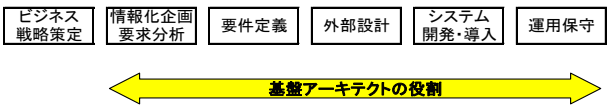

図３　フェーズにおける役割概念図

以下に各局面における基盤アーキテクトの役割について説明する。

(1)要件定義まで

要件検討局面でお客様と一緒に企画立案するケースもあるが、ここではプロジェクト実施観点で説明する。

プロジェクト開始時、提案したシステムに対し、お客様の要件をヒアリングしてその実現性に問題はないかどうかを見直す。特に運用面でのヒアリングは重要である。

お客様が既存のシステム運用面でかなり工夫しているケースも多く、基盤製品に自前の運用ツール（シェルプログラム）を組み込んで利用している場合には特に注意を要する。

機能要件的にはほぼ同じ挙動をする製品に変更するケースは多々あるが、運用要件を見過ごすと後で大きな課題となる。

また、基盤アーキテクトは、プロジェクト体制の都合上参加しないケースもある。その場合はプロジェクトを推進する上で懸念される課題事項をプロジェクト・メンバにプロジェクト開始のキックオフ時に提案引継ぎという形で引き継いでおくべきである。

(2)外部設計

要件定義をインプットとして外部設計を行うが、提案時点で基盤製品は決められているケースが一般的である。したがって、要件定義を踏まえて、提案した製品がそのまま適用できるかどうかを検討しながら外部設計を行う必要が出てくる。この時点では、製品が変更あるいはバージョンアップしている場合もあり、期待した挙動をするかどうかがわからないからである。そうした危険性をあらかじめ排除しておくために、机上調査だけでなく製品の挙動を実際に検証するなどの事前チェックが重要となる。

(3)システム開発以降

基盤アーキテクトは要件定義やあるいは外部設計局面までの参画で良いと認識されている方もいるがこの理解は大きな誤りである。なぜならば、開発・導入フェーズあるいは運用フェーズに入ってから製品のバグや運用制限による問題が発生する場合も多く、その際に個別最適なではなく、システム全体を配慮した対策が必要となるからである。

システム開発フェーズ以降運用フェーズまで、基盤アーキテクトは、基盤PMあるいは基盤PMの技術フォローとして支援参画するか、一部の製品パートを担当するなど、何らかの役割でプロジェクトには可能な限り最後まで参画することが望ましいと考える。構築／テスト／運用フェーズに入ってからも製品の挙動のトラブルに遭遇するケースが生ずる可能性があるからである。

そうした事態に備えて、プロジェクト側では常に基盤アーキテクトと相談できる体制を整えておくべきである。

3. 基盤構築ケースと基盤アーキテクトの役割

昨今のインフラを取り巻く変化により、システムを一から構築するケースは少なく、システム統廃合によるシステム再構築が多いのが実情である。また、クラウドを利用するケースやアプライアンス製品を使用するケースが増えてきている。

そのため、アプリケーション要件に応じて基盤を設計するよりは、現行システムとの整合性を見て非機能要件が満足されているかどうかを確認する観点が重要になる。

システムリプレース、アプライアンス製品およびクラウド環境を利用したケースで基盤アーキテクトが対応する役割について考えてみる。

また、運用基盤やプラットフォーム基盤アーキテクトの必要性は、あまり理解されていないと感じる。次項では、運用基盤・プラットフォーム基盤にフォーカスを当て、構築ケース別にその重要性を説明する。

3.1. システムリプレース

例えば、ハードウェアの導入から数年たち、サポート切れとなるケースを考えてみる。

基本的にはハードウェアの更改と同時に、OSとMWの変更あるいはバージョンアップを伴うことが一般的である。その際に機能面の差異を十分に精査し、現行システム同様の動作保証することは当然であるが、特に下記の点でリプレースに伴う影響を確認しておくことがアーキテクトに課せられた使命となる 。

(1)運用要件

お客様の運用要件をプロジェクト開始の早い時点でヒアリングし、要件に応じた運用システムの設計・仕組みを検討し、方向性の策定・提案および複雑性と運用ミスを排除する仕組みを作成していくことが重要である。現行運用者に対して、ハードウェアの変更に伴い手順が数か所変更されるのは、やむを得ないと説明し理解を求めることは容易かもしれないが、従来の仕組みから大幅に変更する場合は、利点を明確に説明して理解してもらう等の調整作業もアー

キテクトの役割の一つとなる。

(2)移行要件

システムリプレースにおいては、現在稼働中のシステムからの切替え方式について十分検討することが重要である。

例えば、

・ある一定期間システムを並行運用するのか？

・切替日を設けて実施するのか？

という観点である。

実現性、現システムへの影響性、切替え失敗時のリカバリープランなどを十分配慮して移行案を策定し、プロジェクトをリードしていくことが重要な役割となる。

(3)その他非機能要件

例えば、可用性要件に対しては、冗長性を実現するためにあらたな製品選定が必要となるケースなどが考えられる。

また、性能要件に対しては選定した製品で満足するようにチューニングすることが求められる場合もある。

基盤アーキテクトが、どこまで性能要件に対して責任を負う必要があるかについては、ケースによって異なってくる。アプリケーション要件に応じて基盤側で製品を選定している場合には、その挙動について、基盤側が責任を持つことは当然である。アプリケーションがソリューションパッケージである場合には、アプリケーション側に性能要件を保証する責任があると考えるが、技術面に関しては、要請に応じて臨機応変にサポートすることにより、問題の肥大化を未然に防止するといった心構えも備えておくべきである。

その他、信頼性、拡張性、セキュリティ要件に関しても、基盤アーキテクトの担う役割は多い。

3.2. アプライアンスを利用するケース

オープンシステムにおいても仮想化が当たり前の共通基盤として利用されてきている中で、ハードウェアとその上に搭載されるSW（仮想OSからゲストOSに加えてDBやアプリケーションサーバ）を同梱したアプライアンス製品がハードウェアメーカから出荷されてきている。クラウド環境に提供しているケースもあり、今後多くのシステムに利用される可能性が高まっている。

アプライアンス製品を利用したシステムに対しては、基盤アーキテクトはどうかかわって対応すれば良いかを考えてみたい。

アプライアンス製品は標準化基盤システムの一つと見ることができる。まず、アプライアンス製品に付加する動作保証されたMWの選定を行い、非機能要件の一つである可用性や移行性方式を検討する。

次に、お客様の運用要件を確認し、その内容によってはアプライアンス製品の制約により、運用の仕組み変更を依頼することが必要となる場合がある。

プラスの面としてアプライアンス製品は、拡張性要件の検討から開放されるといえる。従来は、拡張性要件に対し、スケールアウトあるいはスケールアップの構成可否を検討し、要件に対応したハードウェア製品の選定に頭を悩ませていたが、容易に拡張が可能なアプライアンス製品は、基盤アーキテクトにはとても魅力的なものになりえる。

今後、アプライアンス製品が広く普及すると、ブラックボックス化されたシステムが増えてくるようになる。システムのアーキテクチャがパッケージ化されてくるので、基盤アーキテクト１人でシステムデザインを担当することが求められる。その結果、細かい領域を担当するよりは細分化されていた基盤アーキテクトの役割が拡がっていく可能性が高く、従来以上に視野を広めることが重要である。

3.3. クラウド環境を利用するケース

クラウド環境を利用したシステムも前項のアプライアンス製品を利用するケースと同様のことがいえる

ただし、クラウド環境では運用システムを共通運用システムとして提供しているケースが多く、共通運用システムを取り入れて利用することが一般的である。

クラウド環境では、ネットワーク、可用性、セキュリティ、運用性という観点が課題となる。利用しようとしているクラウドサービスの提供範囲を確認し、特に非機能要件の可用性、拡張性、セキュリティ、運用性についてお客様システムに適用した場合における整合性を検討する。また、自社でシステムを持つケースに比べて、セキュリティ的に問題がないということを担保し、クラウド環境で提供サービスとしてどこまで対応できているかを調査することも重要となる。

また、クラウド環境を活用することで、災害対策を含めた大規模なシステムが安価に提供されることを可能としており、いままで以上にアーキテクトが対応する範囲が拡がってくる。

前項で述べたアプライアンス製品およびクラウド環境と従来型システムを組み合わせた環境も今後は多く出てくる

と考えられる。今後の基盤アーキテクトは、そうした製品・環境の特性を調査、理解し、従来型のアーキテクチャとアプライアンス製品およびクラウド環境のアーキテクチャを組み合わせたシステムの提案や設計を行うことが求められるようになる。

前項で挙げたケース別基盤アーキテクトの役割を以下にまとめる。

① システムリプレース

・現行保証

・運用性

・移行性

に注意を払いシステム設計を行う。

特に既存システムとの融合性に意識をおき、バージョンアップ、製品の変更による運用面での差異が発生しないようにすることが重要である。

②アプライアンス製品・クラウド環境システム

・運用性

・可用性

・移行性

に注意を払いシステム設計を行う。

今後の基盤アーキテクトは、新しい技術であるアプライアンス製品やクラウドの動向キャッチアップを行い、必要に応じて設計に取り入れることが求められる。

また、クラウド環境は、動的な拡張性を実現し、可用性は意識しなくても良くなってくる。そのため、お客様のニーズに合わせた複数のクラウドサービスの中から最適な組合せを選択提案していき、運用サービスに特色を出す必要がある。

4. 事例に基づく基盤アーキテクトの重要性

次に、基盤アーキテクトがかかわらないでプロジェクトを進めてしまった例を紹介する。基盤アーキテクトがプロジェクトの早い局面で参画していれば課題とならなかった事例である。

4.1. 事例１

最初は、筆者が最近かかわった案件の例である。

ハードウェアの老朽化（保守切れ）に伴うシステム更改案件であるが、ハードウェアが他社製品へ変更となり、OS・MW も大半は他社製品へマイグレーションを行い、一部はバージョンアップする案件である。

上位のアプリケーションは、いずれ大きなアプリケーション更改を控えており、今回は現状アプリケーションからストレート・マイグレーション（業務仕様を変更せずに移行する）を行うというものであり、そのため、基盤を含めて現行同等の動作を遵守するという方針でプロジェクトは開始された。

基本的なアーキテクチャが変更されるものではないので、アプリケーション・基盤ともアーキテトを置かずに推進したプロジェクトである。

今回のシステムは大規模ウェブシステムで、OS は、HP-UX®、Solaris®から AIX®、Linux®に変更、Windows®はバージョンアップ、アプリケーションサーバの MW 製品が X から Y に変更されるという要件である。アプリケーション要件としては、MW の製品変更の影響を受けずに、同じ挙動で動作することである。ここで考慮しなければいけないのは、製品変更に伴う動作の違いに対する十分な検討であり、少しでも動作が異なる箇所があれば、運用に影響をおよぼさないという確認が重要となる。

実際にリリース運用要件は、負荷分散機能を組んだサーバ構成に対し、例えばサーバ#1 には App1 の A’と App2 の B’を先行してデプロイファイルを同期した後に、サーバ#2 には App2 の B’のみを同期化することを可能にすることである。（図 4）

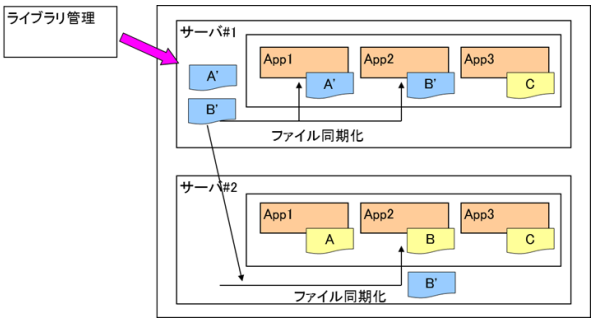

図 4　アプリケーションサーバのデプロイメント概念図

X 製品では、アプリケーション毎に同期デプロイ指定が可能となっており、上記要件を満足していたが、Y 製品ではノード単位にしか同期指定が設定できず、サーバ#1 の一部のアプリにデプロイファイルを配布しようとしてもサーバ#2 のすべてのアプリが同期化してしまうことになる。（図 5）

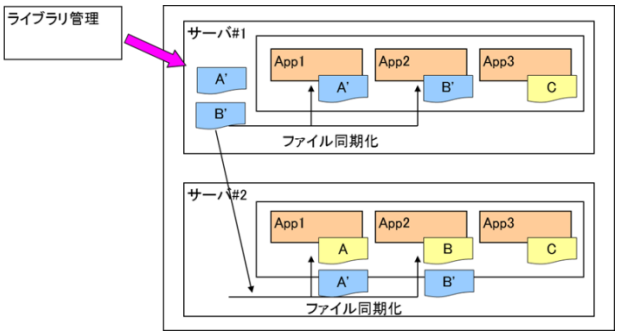

図５　Ｙ製品のデプロイメント挙動

すなわちＹ製品の基本機能ではサーバ#1とサーバ#2に異なるデプロイファイルを配布することができないことが判明し、製造フェーズ以降にサーバ構成とデプロイ運用方式を大幅に変更せざるを得ないという問題が発生した。

プロジェクトの早い時点でアーキテクトが参加し、現行システムの機能・非機能分析を行い、製品変更に伴う差異分析を前もって実施し、上記のような課題事項を洗い出してプロジェクト側に認識させておくとともに、非機能要件の中でも特に運用要件に対する考慮を行ってから設計を進めておけば防げた課題であったと考える。

4.2.　事例２

事例１と同じ案件であるが、バックアップ運用にかかわるところでも大きな課題となった例である。バックアップ運用として、現行システムのシステムデータ・バックアップはテープで取得する運用を実施しており、本案件でも同様の要件となっている。（図６）

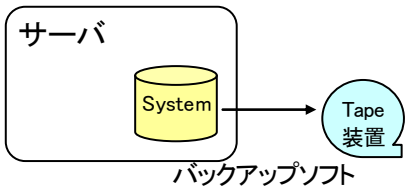

図６　テープバックアップの概念図

OS を停止せずシステムバックアップを取得するという要件が追加されたため、製品を利用して要件を実現しようとしている。ただし、選定した製品は、ハードウェアとソフトウェアの組合せ構成として従来事例がないケースであったことがプロジェクトに入ってから発覚している。

実際、選定したバックアップソフトを導入して動作確認するとテープからリストアする際に大幅に時間が掛かる（10 数時間程度）ことが判明した。そのために問題切り分け・原因調査・メーカとのやり取りに多くの労力・時間をプロジェクトとして費やされた。数か月後、バックアップソフトのバグということが判明したが、メーカからバグ修正版の提供も遅れており、プロジェクト残課題となっている。

ただし、ディスクへバックアップ・リストアした場合は問題のない時間でリストアできることは、切り分け調査の中で判明していたので、図６のように一度ディスクに取得した後、テープ装置に OS 標準コマンドで取得する等の方策も考えられたのだが、テープ装置への直接取得にこだわったことにより、方向性変更に時間を費やしている。

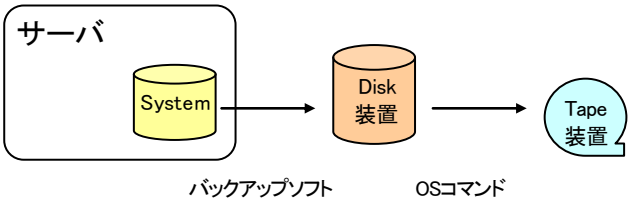

図６　ディスク to ディスク to テープ方式

本事例は、バックアップソフトの選定に基盤アーキテクトが検討参画しなかったことが大きな問題である。さらに、今回のような場合、製品選定後に十分検証を実施してからプロジェクトで導入を行うようにするべきであった。

また、本プロジェクトでは事前に動作検証することができなかったが、上述のようなことは想定されるリスクであり、あらかじめうまくいかなかったケースの方針を決めておけば、当初の要件に固執してメーカとのやり取り・切り分け作業に多大な時間を掛けなくても済んだといえる。

4.3.　事例３

最後は、増え続けるストレージに対する運用要件の一つであるバックアップに関するもので、製品ありきで進めたため、終わらないバックアップシステムとなってしまった事例である。

本事例では、大容量のデータをストレージに保存するシステムの統合で、バックアップ要件としてディスクストレージにあるデータをテープストレージへ保存することである。システム統合によりデータ容量が数百ＴＢかつ数億ファイルとなり、その容量に見合うテープライブラリを用意した。（図７）

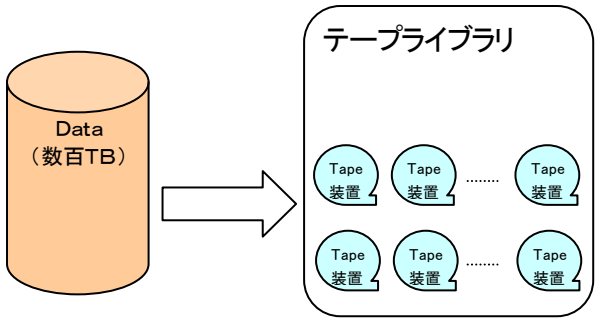

図７　大容量バックアップシステム

一般的にバックアップソフトは、バックアップ時には、ファイル単位でバックアップ情報データ作成を行い、リストア時はバックアップ情報データを検索する動作をする。その結果、ファイル数が多くなれば多くなるほど、バックアップ情報の検索処理時間にテープへの読み書き以上の時間を費やすケースも多く見られる。

本事例では、実運用に入るとバックアップ・ランタイムが異常に長く、すべてのバックアップデータを取り終わるまで１か月程度掛かっている場合もある。ただし、本案件での性能要件は、ある容量のバックアップを所定時間で完了することとなっており、その要件は満足している。

リストアに関しては、通常バックアップより約 1.5～2 倍時間が長くなることが想定され、ストレージ全障害時からの回復を考慮すると現実的ではないシステムとなっている。基盤アーキテクトが要件決定プロセスに参画して検討していれば、運用要件に対して問題の有無検討を行えたと考える。

本事例のようなケースでは、ファイル単位のバックアップ方式ではなく、ディスク（ボリューム）単位のイメージデータバックアップ方式によりリストア時間短縮を実現する、あるいはアーカイブシステムで良く用いられているデータを二重化して保存するような仕組みの導入を検討するべきであった。

上記で挙げた３つの事例は、個別要件最適化をおこなったことにより全体最適の整合性が取れなくなったものである。

そのため、運用要件・製品選定に対する全体最適を見通した検討の考慮不足が、後フェーズで露呈している。

運用の場合、実際やって見ないとわからないケースもあるが、今回の事例は、運用要件に対し、システムの機能・性能に問題なく対応できるかどうかを基本設計フェーズ時点で、基盤アーキテクトも交えて検討すれば防げた問題である。

また、製品を選定する場合、過去の実績などを十分考慮し、実績があっても未経験の機能などを利用する場合には事前に導入検証を行うなど十分な注意を払うことが重要である。

5. 基盤技術者の育成

最後に、基盤技術者の育成方法について一例を挙げてみたい。

今後の基盤取り巻く環境をみるとアプライアンス製品やクラウド環境の利用が一般的になり、構築より運用比重が高まってくる。従来は分業化されている場合が多いが、構築から運用まですべて対応する必要が出てくる。

運用に強い人は構築を意識、構築に強い人は運用を意識し、キャリアを拡げて基盤アーキテクトを最終ゴールとする SE となるべきである。もちろん、その過程で、常に業務を意識して最適なシステムを考え続けることを忘れないでほしい。まず、何か一分野で良いので基盤における専門技術を身につけてほしいと考える。その経験を積んだ後、技術幅を拡げることを考えていただきたい。プロジェクトの都合でいろいろ経験しているが専門性が身についていない SE であり続けると、本人にとっても会社としても不幸な状態である。

当社のようなハードウェア/ミドルウェアメーカでない会社は、強みとするメーカの製品だけでなく、お客様に対して製品の良し悪しを説明することができるように複数メーカの製品に精通していることが必要である。なぜならば、幅広くハードウェア／ミドルウェアに関する知識を単独メーカ製品に、かたよることなく知識を有していることがアーキテクチャを具現化するシステムを設計するためにも基盤技術者として大事になると考えるからである。

また、基盤技術者は、強みとする技術領域とは別にハードウェア技術・ネットワーク技術に強い方が望ましい。その理由は実際のプロジェクトでは、ハードウェアの搭載・故障、ネットワーク障害などによるトラブルも多く、現場に入って解決しなければいけない場合が多いからである。

さらに、製品構築技術に加えて、実際に構築されたシステムを実業務の中で運用経験を積むことがより重要であり、その経験の積み上げにより実例で挙げたような問題を防ぐことができると確信している。

6. おわりに

基盤を取り巻く環境が変化していく中、従来のように専門特化型技術でキャリアを積んで歩んでいくことは少なくなってくる。そのため、より深い専門家というよりは、自分の専門性を持ちつつ、幅広い知識を身につけていることが重要となる。

また、クラウド環境が世の中に広まってくると、従来のようなオンプレミス型基盤構築作業のニーズは少なくなってくることが予測されるので、運用に強い基盤アーキテクトを目標とするキャリアを積んでいく方が良い。

時代の変化・新技術にアンテナを張りつつ、それらを使う機会があれば、積極的にチャレンジして自分のものにするという意欲を持ち続けて対応していく姿勢が大事であり、時代の変化に対応したシステムで非機能要件の実現性にいままで以上に注意を払って設計を行うことが特に重要となる。

最後に仕事を進めていく上でプロフェッショナルは、以下のように、求められていることを忘れずに日々精進する姿勢が大事である。

・個人のプレーで完結するのではなく、チームとして成果を出すこと
・お客様の期待値をはるかに超える成果を出すこと
・何らかの専門分野で一流であること

参考文献

1) IT スキル標準 V3 2011 2 部キャリア編
職種の概要と達成度指標（４） ＩＴアーキテクト
IPA（独立行政法人 情報処理推進機構）
