第 23 章　アーキテクチャの設計

「アーキテクチャ」とは何か

「アーキテクチャ」とは難しい言葉だが、一体「アーキテクチャ」とは何だろうか。いつも引き合いに出している ISO と IEC、IEEE が合同で発行した Vocabulary（ISO／IEC／IEEE 24765：2010）には、次のように記述されている[ISO10a]。

1. システムの基本的な構成。その構成要素、それらの相互関係及び環境との関係、並びに、設計及び進化を導く原則、を具体化したもの。（ISO／IEC 15288：2008）
2. システムやコンポーネントの組織的な構造。
3. システムの組織的な構造とその開発のガイドライン。

これではまだよく分からないので同じ資料で「アーキテクチャの設計（Architectural Design）」を引いてみると、次の記述がある[ISO10a]。

1. コンピュータ・システムを開発するためのフレームワークを確立するために、必要とするハードウェアとソフトウェアのコンポーネントと、それらの間のインターフェースを定義するプロセス。
2. コンピュータ・システムを開発するためのフレームワークを確立するために、必要とするハードウェアとソフトウェアのコンポーネントと、それらの間のインターフェースを定義した結果。

つまりアーキテクチャの設計とは情報システムの枠組みや構成を決めることであり、ソフトウェアだけでなくハードウェアにも関わりを持つことが分かる。

なお共通フレーム 2013 では、アーキテクチャの設計を「ソフトウェア方式の設計」という言葉で表している[IPA13a]。

このアーキテクチャの設計の議論は一般論ではうまく話ができないので、例を挙げて説明を進めたい。

アーキテクチャの設計の基は非機能要求

要件定義書上に記載された機能要求と非機能要求の中、機能要求が全く関係しないとはいわないが、もっぱら非機能要求がアーキテクチャの設計の基になる。機能要求の方は、その要求を満たすためのプログラムやデータベースがその情報システムで適切に機能できれば良い、という程度の関係である。

ここで、例を一つあげて説明を進めたい。ある情報システムの非機能要求に、「徹底した可用性を実現すること。つまり、365 日 24 時間の稼働を実現すること」というものがあったとする。そうすると、どのような構造の情報システムを作るべきだろうか。

仮に私がこの情報システムのアーキテクト [1]だったとしたら、機能の拡張性も考慮して、負荷分散クラスター方式を採用するだろう。

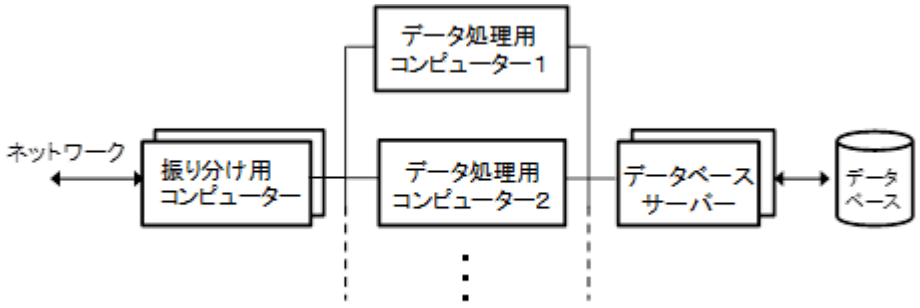
クラスター構成とは、複数のコンピュータを連結し、利用者や他の情報システムからは、全

[1] アーキテクチャを設計する人。

体で 1 台のコンピュータであるかのように見せる方式をいう[JUAS10a] 。

クラスター構成を採ることによって、複数台を同時に稼動させて並列に処理を行なわせる場合（負荷分散クラスター方式）と、1 台が稼動してもう 1 台は待機しておき、障害発生時に即座に待機系に切り替える場合（フェールオーバー・クラスター方式）など、様々な対応が可能になる。しかしフェールオーバー・クラスター方式では、仮に障害が起きた場合、本番系から待機系に切り替える間は情報システムを停止しなければならないので、徹底した可用性は実現できない。そのためこの場合に採用できる方式は、負荷分散クラスター方式が妥当な選択ということになる。

負荷分散クラスターの方式は、本来は 1 台のコンピュータでは処理できないほどの多量のデータを円滑に処理する方式である。しかし負荷分散クラスターの方式は非常に巧みにバックアップの機能も果たせるので、多量のデータ処理への対応だけでなく、障害対策のためにも使用されることがある。負荷分散クラスター方式を、図表 23-1 に示す [JUAS10a]。



図表 23-1　負荷分散クラスター方式（ [JUAS10a]より）

ここで問題になるのは、データ処理用コンピュータを何台用意するのか、振り分け用コンピュータはどういう構成にするのか、データベースは何組持つのか、などを決めることである。

仮にデータ処理用コンピュータは、当面 2 台で良いことにしよう。ただし必要に応じてこのコンピュータを所定の台数まで適宜増築できる構造にするという前提を置く。

振り分け用コンピュータは、正副 2 台の構成にするとしよう。正副 2 台の構成とは、正常時は本番系が稼働し、待機系は本番系が正常に稼働しているかどうかを監視していて、仮に異常を発見すると自動的に本番系を止めて待機系が処理を取って代わる方式にすることを意味する。

しかし過去に起きた障害では、待機系が障害を起こして、正常な本番系が障害を起こしたものと誤認して、その正常な本番系に障害を持った待機系が取って代わろうとして、情報システム全体に障害を起こしたケースが報告されている[JUAS10a]。それに対する対策はどうするのか。このようなことはめったに起こらないと割り切って、つまり無視して、その場合は一時的に情報システムの停止を許容するのか、あるいは振り分け用コンピュータを 3 台以上で稼働するようにして、本番系の正常稼働を多数決で確認する方式をとるのか、などを決めなければならない。

データベースについても、RAID 方式のディスクを使うことにして全体で 1 組だけにするのか、それでも 2 組を並行して稼働させ、同時に更新するのかといったことを決める必要がある。

データベースについていえば、そのバックアップは取るのか／取らないのかも問題になる。

取る場合は、当然情報システムの稼働中に取らないといけない。バックアップを取る場合にはそのバックアップをいつ、どのようにしてリストアするのか、そのリストアを情報システムを止めずにする必要があるのか、などが問題になるかもしれない。

その他のアーキテクチャに関わる事項

それ以外に、アーキテクチャに関わる事項として決めなければならない事項に、以下のようなものがある。

- OS や DBMS などは、LINUX や PostgreSQL のようなオープンシステムを使用するのか、マイクロソフトやオラクルなどが販売している商用のソフトウェアを使用するのか。
- アーキテクチャの設計で明らかになるミドルウェアの機能を、自社で開発するのか。あるいは適切なパッケージが入手可能か。
- アプリケーション・プログラムの開発に使用するプログラム言語は、何にするのか。

それ以外に、場合によれば次のような問題も解決しなければならない。

- この情報システムは、1 カ所のセンターで稼働させるのか。あるいは 2 カ所のセンターで稼働させるのか。
- 2 カ所のセンターで稼働させる場合、情報システムに正と副を作るのか、並行稼働させるのか。正副を作る場合、正側のデータベースの更新を副側のデータベースにどのように反映させるか。並行稼働する場合には、処理の対象を地域で分けるのか、一方の処理はダミーとして捨ててしまうのか。
- 止まることのないコンピュータ等の電源を、どういう方法で確保するか。
- 止まらないネットワークを、どう実現するか。
- ミドルウェアとアプリケーション・プログラムのテストをどのように行うか。（特にこの場合は、負荷分散が適切に行われていて、しかも一部に障害が起きても情報システム全体がダウンすることがないことの確認をどうするか。）
- 権限のない情報システムや不正なアクセスから、情報システムをどう守るのか。

アーキテクチャの設計として行わなければならない事項は、まだあるだろう。つまりアーキテクチャの設計とはアプリケーションに関わる部分を除いて、情報システムの枠組みや構成などをしっかりと決めることを意味する。具体的な設計作業の成果は、ミドルウェアの設計書だけかもしれない。しかしその作業を完遂するために決めなければならないことは多岐にわたり、影響範囲もたいへんに広い。

また不正なアクセスへの対応などセキュリティに関わる対応はアーキテクトだけで対応するのではなく、セキュリティの専門家との共同作業になるだろう。

アーキテクチャの設計は業務システムの設計に先立って行い、相互に整合性がとれた矛盾のない状態にして、その結果をしっかりと文書化し、利害関係者に適切に伝達しておかなければならない。

ただ、アーキテクチャの設計は情報システムの開発ごとに行わなければならないというようなものではない。基幹系システムの再構築のような大きなイベントに伴って新しいアーキテクチャを設計して、その後しばらくはその結果を再利用するというような方法をとるのが適切であろう。

アーキテクトという仕事

私はこれまで、生涯で一度だけアーキテクトとして仕事をしたことがある。その時の経験から、アーキテクトの仕事はソフトウェア技術者の仕事の中で、最高にやりがいのある、おもしろい仕事だと考えている。

この章では、負荷分散クラスター方式を例に引いてアーキテクチャの説明をしたが、アーキテクトはいくつもの情報処理の方式を精通し、その長所、実現上の留意点などについて、熟知していなければならない。つまり、この仕事を的確に行うために勉強しなければならない範囲は広く、奥行きも深い。アーキテクトが決める事項が影響する範囲は広く、仮に間違った場合には事後に簡単に修整できないケースが多いからである。

アーキテクトとして仕事をする機会に恵まれた人はその機会を有効に生かし、自分の仕事を楽しみ、しっかりと対応してほしい。

キィワード

アーキテクチャ、フレームワーク、要件定義書、非機能要求、アーキテクト、ミドルウェア

参考文献とリンク先

[IPA13a] 情報処理推進機構ソフトウェア・エンジニアリング・センター編、「共通フレーム 2013 経営者、業務部門が参画するシステム開発及び取引のために ソフトウェアライフサイクルプロセス 共通フレーム 2013」、オーム社、平成 25 年.

[ISO10a] ISO/IEC/IEEE, “System and software engineering – Vocabulary－ISO/IEC/IEEE 24765:2010(E),” ISO/IEC, 2010-12-15.

[JUAS10a] 日本情報システム・ユーザー協会、「情報システムの信頼性向上ガイド 障害を発生させない、被害を拡大させないための、システム対策」、日本情報システム・ユーザー協会、2010 年 7 月.

（2014 年（平成 26 年）6 月 8 日　新規作成）
(2016 年（平成 28 年）4 月 21 日　一部追加)