❷

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ー検討と基本設計
―短納期開発SIビジネスの立ち上げ―

Consideration and basic design of Web-hosting infrastructure architecture: Launch of short delivery time development SI business

日本アイ・ビー・エム株式会社
金融第二サービス事業部
保険ソリューション・システム部
コンサルティングITアーキテクト
佐藤 浩司

Hiroshi Satoh
Consulting IT Architect
INS Solution System
Sector Services - Financial Services No.2
IBM Japan, Ltd.

本論文では、ホスティング・サービスを前提としたSI開発プロジェクトを立ち上げるに当たって、「何が違うのか」「どうやったら短期間でできるのか」といった課題に対する解決策の方法論を検討してみます。特に、プロジェクト立ち上げ当初のインフラストラクチャーの要件定義局面においては、基本設計を行う上での技術的なバックグラウンドの押さえとプロジェクトを進める上での共通認識を得るためのアウトプットの作成が必要となるでしょう。短期開発のスピードを維持するためには、大局で方向性を誤らない程度の大胆さも必要です。網羅的な整理は必要ですが、完璧さは求めず、課題を残しつつも、コミュニケーションを良くして、物事を前に進めるための方法論も必要です。そのためのフレームワークとしてテクニカルには「アーキテクチャーの考え方」をベースに、プロフェッショナルとしては「自他ともに納得させるアウトプットの作り方」について考察していきます。

In this paper we take a look at the methodology employed in measures to solve questions relating to the launch of SI development projects premised upon hosting services such as the exact nature of differences and how short delivery times can be observed. In particular, at the stage when infrastructural requirements are being defined at the start of a project, it's essential to create output for pinning down the technical background to basic design and for obtaining a joint awareness for moving forward with the project.
In order to maintain the speed of short-term development, what's needed is a bold approach, although one in which no mistakes are made as regards the overall orientation. Exhaustive coordination is required, but there's also a need for a methodology for improving communication and moving matters forward without aspiring to perfection and with various problems left unsolved. As frameworks for this, we take a look at how, in our capacity as professionals, to create outputs likely to prove convincing both to ourselves and to others on the basis of an approach to technical matters rooted in the concept of architecture.

1. はじめに

日本経済の転換の過程では、短期間勝負のWebプロジェクトを立ち上げる能力が最近のビジネス・チャンスを作ると考えられています。

そのためのITビジネス会社のインフラストラクチャーが具備すべき機能としては、以下のようなことが挙げられます。

- 新しいビジネス・モデルを実現できるIT インフラストラクチャーを備えている。
- 企業間の共同投資のスキームを作りやすい。
- 短納期開発である。
- 仮に失敗しても、投資リスクを抑制できる。

これらの結果として、以下のメニューに絞り込むことができます。

- インターネットを用いたB to C、B to E、B to BのIT領域である。
- 過去のしがらみに依存しない、独立したシステムづくりを選択できる。
- 1年以内の開発期間で、即、プロジェクトを立ち上げられるITパートナーがいる。
- コンピューター資産を持たず、ITアウトソースの形態がとれる。

つまり、インターネット分野の“ SI ”と“ ホスティング・サービス ”を組み合わせることによって、市場のニーズにこたえることができるのです。

2. 設計はアーキテクチャーを考えるところから始まる

2.1. 進める上での課題

2.1.1. 提案時の注意

アーキテクチャーを考える活動は提案段階で、ある程度なされているはずです。提案書上でのシステム・イメージとして、粗い基本設計が必要です。システム構成と開発費用も、その前提に基づいています。つまり、ここで述べる思考方法は、提案時に粗く考えられたものを、要件に合わせて修正し、詳細化する作業ということになります。

ビジネスの観点からの注意点としては、コスト面への影響が挙げられます。そもそも提案時に示されたシステム要件と、具体的な要件定義局面のシステム要件では、時間の経過により、お客様のポジションの変化も手伝って、少なからず変わってしまうものです（少々の削除と多くの追加）。提案金額の縛りが強い場合には、大きな変更は難しくなるため、最初の見極めは重要です。前提条件をきちんと記載しておくことも肝心です。

2.1.2. 基本設計の技術的な心構え

プロジェクトである以上、限られた時間で結果が求められます。順を追ってアウトプットを出していく必要があります。タイム・チャートの上で、明日からの作業として、要件を確認しつつ基本設計の肉付けをする必要があります。重要な点の第1は、事前にどれだけの技術的な見通しと自信を持てるかということです。

- 基本設計について51%以上のイメージを持っている。
- システム全体イメージが視野に入っている。
- 他システムとのインターフェースの種類が見えている。
- 使用する製品とリリースがリストアップできている。
- 実績のある技術と、心配のある技術の区別がついている。
- IBMで進めることの強みが説明できる。

もちろん、すべてを自身で経験していないとしても、経験者の情報やTCTを受けることによって、技術的な評価をしておくべきです。

第2には、検討項目の目次とタイム・チャートを引くことです。目次を作るということは、シナリオ作りであり、経験とスキルのいる作業です。基本設計を構造化の手法で切り出していく必要があります。

- 全体システム像のアーキテクチャーを描く。
- 処理パターンをマッピングして、システム化の範囲を決める。
- コンポーネントとその関係を整理して、作業を分かりやすい単位に分割する。

ひな型の検討項目は、Webシステム開発ガイド（社内データベース）より、要件定義局面の作業を参考にするとよいでしょう。能書きもさることながら具体例が欲しいものですが、それがない場合は、実質的には個別に手作りとなってしまいます。作業者のスキルと決断力が求められるポイントです。

2.1.3. 安心感を与えるアウトプット

プロジェクトは、お客様とプロジェクト・メンバーの2系統の相手のいる作業です。進め方やチェック・ポイントを示して、働き方の道筋を付けて、安心させることが最初の仕事となります。また、検討項目の優先順位付けも必要です。時間のかかりそうなテーマには早く仕掛けることにし、お客様の課題もはっきりさせていきます。

- 重要な技術課題を見つけ出し、検討の早い段階に持ってくる。
- 項目としては重要だが、結論に大差のでない項目は優先度を下げる。

第3には、ドキュメンテーションの規約作りです。コミュニケーションの道具の選択である以上、メンバーの創意や文書ツールの嗜好（しこう）は無視できませんが、実績のあるひな型をベースに作成するとよいでしょう。必要な事務的な決め事は以下の通りです。

- 要件定義書のフォームを決める。

- 課題管理表のフォームを決める。
- 技術検討と意思決定のための会議体を決める。
- 文書による連絡票のフォームを決める。
- 議事録のフォームを決める。
- 進捗管理のWBSのフォームを決める。

以上ができていると、人との情報連携と責任共有ができ、気持ちに余裕が出てきます。後続の作業品質も上がり、プラス思考のサイクルが作られます。

2.2. アーキテクチャーを設計する

2.2.1. 進め方と技術課題

アーキテクチャーの設計は、図1に示す検討の流れに従って実施します。

ここでは、ホスティング・サービスの利用を前提とした、システム開発の具体例を用いて説明します。

2.2.2. ケースの説明

具体例は、同業種のお客様が共同出資して設立した、新シス

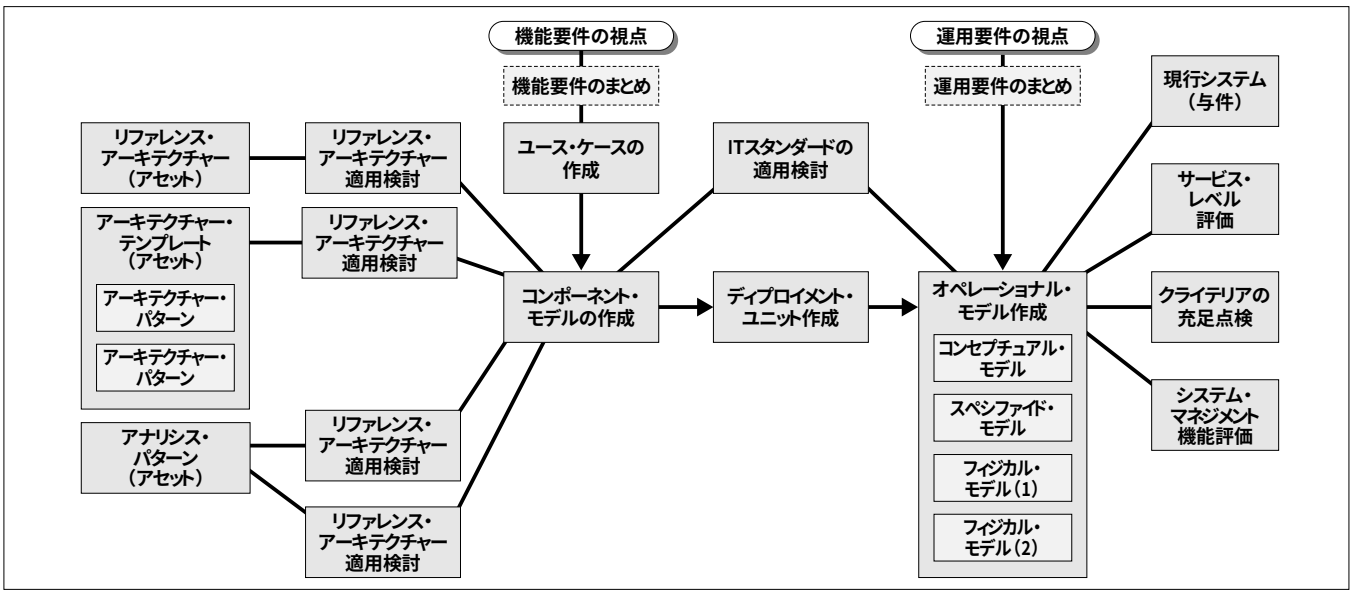


図1. アーキテクチャー検討の流れ

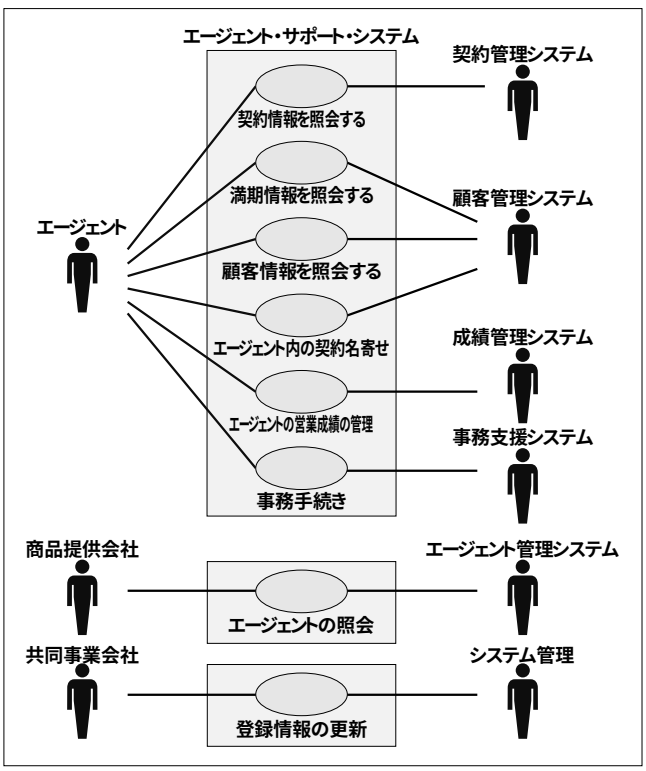


図2. ユース・ケース図の作成

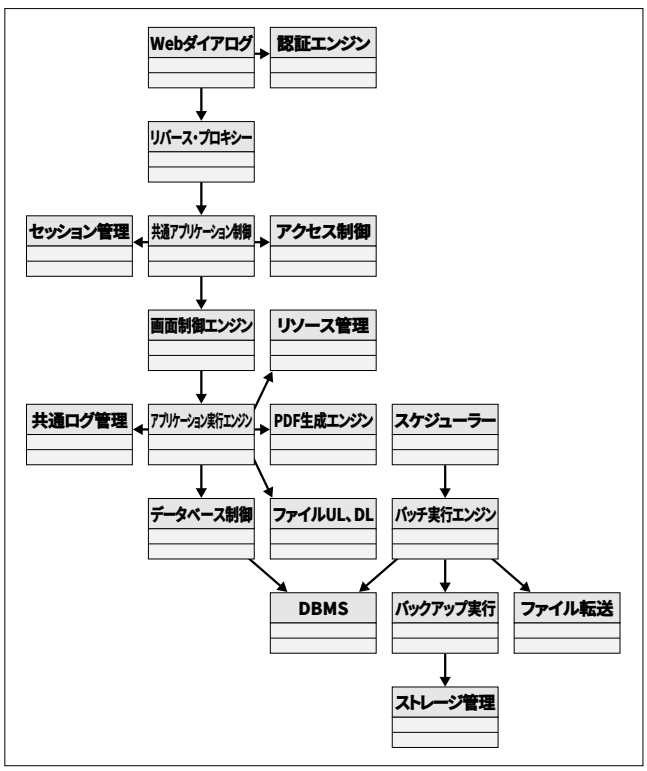


図3. コンポーネント・リレーション図の作成

表1. ディプロイメント・ユニットの作成

(1)外部アクセス・ポイント						
	アクセス・ポイントDU	対象	アクター	場所	サービス期間	同時ユーザー数
U1	ブラウザー	インターネットPC	エージェント	全国エージェント・オフィス	平日24時間 夜間は参照のみ	10,000以上
U2	ブラウザー	インターネットPC	プロバイダー	プロバイダー・オフィス	平日24時間 夜間は参照のみ	100
U3	ブラウザー	イントラネットPC	自社ヘルプ・デスク	委託先オフィス	勤務時間内	10
U4	ブラウザー	イントラネットPC	自社システム管理者	自社オフィス	24時間 保守時間を除く	10
U5	Telnet	PC	自社インフラストラクチャー管理者	自社オフィス	24時間	1

(2)アプリケーション・データ				
	実行DU	タイプ	使用車	使用頻度
P1	ブラウザー認証	インターネット	主に8:00～22:00	5件／秒
P2	ブラウザー参照更新	インターネット	主に8:00～22:00	1件／秒
P3	ブラウザー参照更新	インターネット	勤務時間内	数百件／日
P4	ブラウザー参照	イントラネット	勤務時間内	数百件／日
P5	ブラウザー参照更新	イントラネット	勤務時間内	数十件／日
P6	サーバー	UNIXリモート・ログイン	随時	10件／日

(3)パーシステント・データ				
	パーシステントDU	データ・タイプ	データ・サイズ	データ保存期間
D1	ローカル管理情報	PCツール・データ	約2Kバイト	最新情報のみ
D2	認証登録データ	RDBデータ	約1Kバイト	最新情報のみ
D3	契約・顧客管理情報	RDBデータ	約2Kバイト	解約後1年
D4	ファイル転送データ	フラット・ファイル	約2Kバイト	最大7世代
D5	システム管理情報	RDBデータ	約1Kバイト	最新情報
D6	統計情報	RDBデータ	約100バイト	2カ月

＊RDB：リレーショナル・データベース

テム開発と運用のための共同事業会社向けのシステム構築例です。以下に、前提を列挙します。

(1)共同事業会社システム

- 開発要員リソースを持たないので、すべて委託開発とする。
- コンピューター・リソースを持たないので、ホスティング・サービスを利用する。

(2)外部システムとのインターフェース

- 別途あるゲートウェイ・システムが、エンド・ユーザーのアクセス・ポイントとなる。
- ゲートウェイは、Webのシングル・サインオン機能とファイル転送中継機能を持つ。
- 共同事業会社は、ASPの位置付けでゲートウェイと接続する。

(3)プロジェクト

- 1次アプリケーションの本番までの9カ月が、与えられたインフラストラクチャー開発期間である。
- 主要なステークホルダーは、出資各社とゲートウェイ協議会である。

2.2.3. ユース・ケースの作成

ユース・ケースについてはアプリケーションを説明するため

表2. コンセプト・モデル表の作成

(1)ノード				
	ノード	説明	使用可能度	パフォーマンス
CN1	エージェント インターネットPC	全国のエージェントが使用	24時間	中速据置PC
CN2	プロバイダー インターネットPC	プロバイダーの管理者が使用	24時間	中速据置PC
CN3	自社 ヘルプ・デスクPC	ヘルプ・デスクのオペレーターが使用	勤務時間	高速据置PC
CN4	自社 システム管理PC	システム管理者が使用	勤務時間内	中速ノートPC
CN5	Telnet PC	インフラストラクチャー管理者が使用	24時間	低速ノートPC
CN6	ゲートウェイ・サーバー	認証を実施	24時間	―
CN7	自社アプリケーション・サーバー	インターネット系の顧客契約管理のデータベース・アプリケーションのサーバー	24時間	5～40件／秒
CN8	プロバイダー・サーバー	プロバイダーのホスト	各社運用	―
CN9	VANサーバー	VANの中継サーバー	24時間	―

(2)ロジカル・コネクション				
	ロジカル・コネクション	説明	使用可能度	パフォーマンス
LC1	インターネットと ゲートウェイ認証サーバ間	HTTPセッション	24時間	低速回線
LC2	ゲートウェイ認証サーバーと 自社サーバー間	HTTPセッション	24時間	大容量回線
LC3	ゲートウェイと自社サーバー間 のファイル転送	TCP/IP全銀手順	24時間	大容量回線
LC4	ゲートウェイとプロバイダーの サーバーの間	HTTPセッション	24時間	中速回線
LC5	ゲートウェイとプロバイダーの ファイル転送	TCP/IPファイル転送	9:00～22:00	中速回線
LC6	VANと自社サーバー間	TCP/IP全銀手順	24時間	低速回線
LC7	VANとプロバイダーの サーバー間	BSC全銀手順	24時間	低速回線
LC8	自社イントラネット端末	HTTPとTCP/IP	24時間	中速回線

に登場するアクターとシステムの関係を記述します。図2のポイントは、新システムでは、システム開発の与件が、エージェントがブラウザーを使用する点であるということです。これによって、エージェントでの、装備選択の特殊性をなくし、アプリケーションの変更も全国一斉に可能になります。

2.2.4. コンポーネント・モデルの作成

次に、図3のようなコンポーネント・リレーション図を描きましょう。機能要件を満たすための論理的コンポーネントと、その関係を示しています。ポイントは、この時点ではインフラストラクチャーや場所には依存せずに、純粋に機能をマッピングしている点にあります。

2.2.5. ディプロイメント・ユニットの作成

ディプロイメント・ユニット（DU）の記述は、幾つかの切り口で、機能に関与する対象の特徴を、性能や運用要件を含めて整理していきます（表1）。基本設計の局面なので、詳細化は外部設計に譲ります。ポイントは、インフラストラクチャーの機能要件にかかわる機能を網羅的に整理することにあります。

2.2.6. オペレーショナル・モデルの作成

ここは機能要件に対して、運用要件を加えてインフラストラクチャー構成を固めていくステップです。コンセプト・モデルとしては、表2に機能要件中心のノードとロジカル・コネクションを記述しました。これを図にすると、図4のようになります。ポイントは、ロケーションを物理的に三つの領域に分けていることです。エージェントとプロバイダーのユーザーは、インフラストラクチャーの視点からは機能的な区別はしていません。

一方、ゲートウェイがプロキシー機能を持つことから、バック・エンド領域には、AFeBの3層構造（PL、BL、DA）が存在しています。

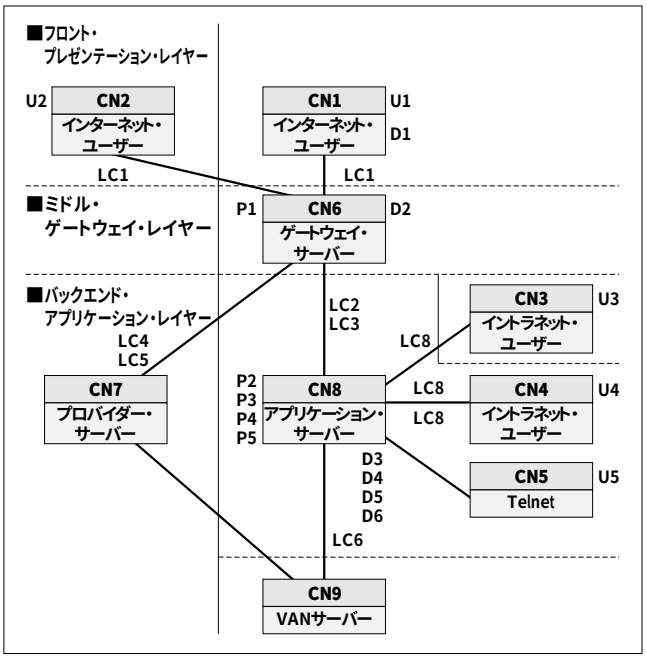


図4. コンセプト・モデルの作成

2.3. 技術課題の解き方

図5に最終形を示します。この図に至るまでには、幾つかの技術課題に直面し、これを解かなければ基本設計が完了しないハードルがあります。ここでは、技術課題5点について解説します。

2.3.1. 対外接続のIPネットワーク設計

IPネットワークについては、同一のイントラネット・ネットワークにいるメンバー間の整合性がとれている必要があります。ここが満足されないと、IPを前提としたシステムは構築できません。メンバーには、ゲートウェイと、もともと個別のアドレス体系を持つ各社と、VANと、当システムがあります。ゲートウェイはリバース・プロキシー・サーバーであり、URLの名前変換を行います。ホスト名とIPアドレスの解決ができるように、ドメイン・ネーム・サーバーの設置とその登録更新の課題があります。

重要なポイントは、実はネットワークの物理接続にあります。ゲートウェイと本システム間の直結回線接続ができないという問題がありました。図6の（2）のように直結回線がないと、IPのルーティングが定まらないので、ゲートウェイに個別のサーバーが必要になるという、おかしな構成になってしまいます。IPネットワークは初期

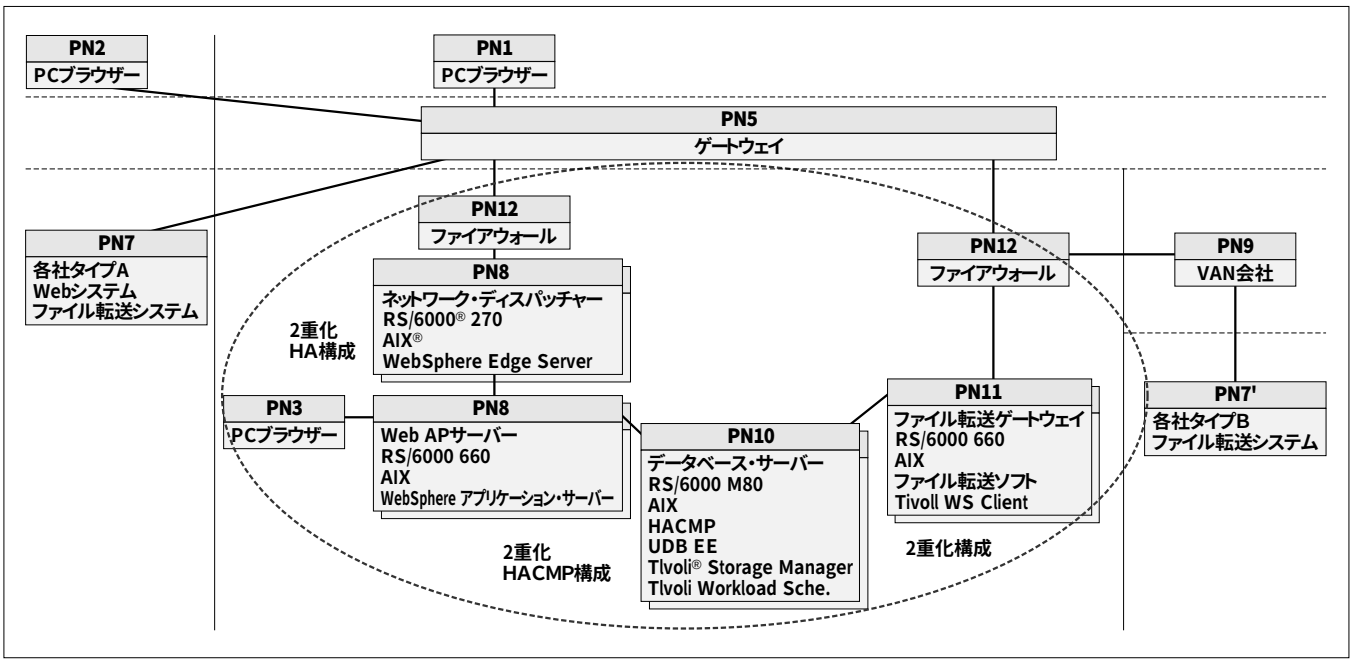


図5. フィジカル・モデルの作成（その2）

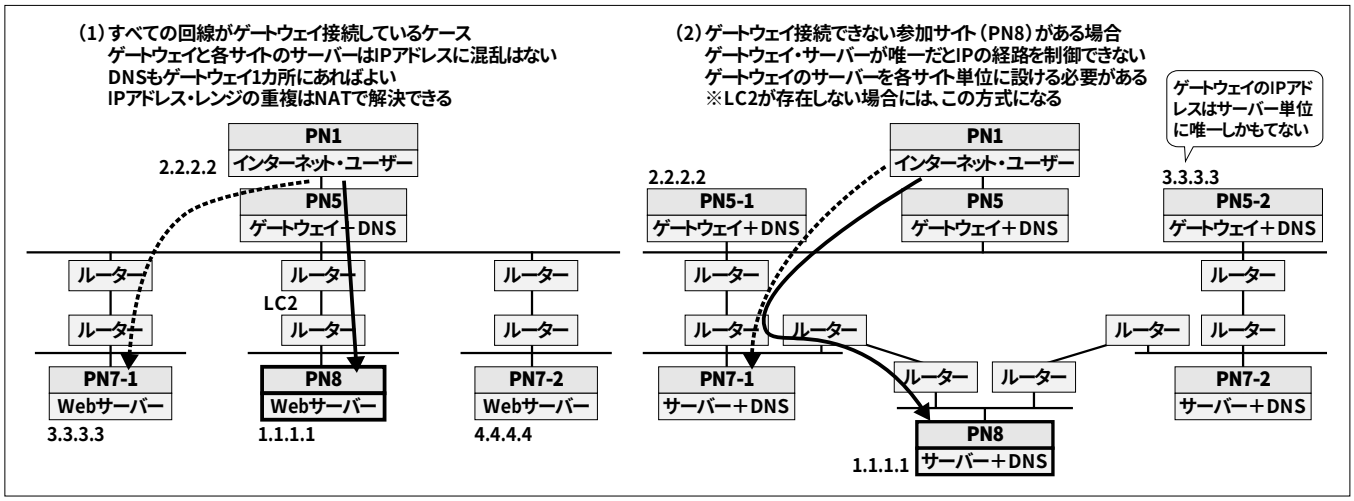


図6. IPネットワーク設計

の段階で技術的に検証すべき対象です。

2.3.2. シングル・サインオンのメカニズム

シングル・サインオン（SSO）は本システムの与件であり、特長の一つとしてアピールされている重要な機能です。

SSOの方式は基本的には2通りしかありません。一括認証方式と分散認証方式です。前者は全員一律のルールに統制されている必要があります。例外があると、後者の方式の併用を考えるという面倒な事態になります。後者は、ユーザーIDの読み替えと属性のトランスファーを受け、自社システムで再度認証することになります。当システムでも、2.3.1節の話題とペアでこの問題を持っていましたが、幸い前者に落ち着く結果になりました。

シングル・サインオンの認証機能は、ゲートウェイが一括して行います。接続されるシステムはURL変換により接続され、自社設定のユーザーIDと属性情報をゲートウェイのHTTPヘッダー情報から受け取り、環境変数としてアプリケーションで扱うことができます。これにより、アクセス制御にユーザーIDを用いることが可能となります。

ゲートウェイがセキュリティーのフォーカル・ポイントとなるため、ネットワーク上、ゲートウェイを経由して他社システムに入ることは許されません。IPネットワーク・レベル、プロトコル・レベル（HTTPS）、認証レベルのセキュリティーが確保されています。

2.3.3. Webシステム間のシームレスなリンク

各社サーバー間のWebリンクは、URLのリンクにて実装されます。リンク情報はゲートウェイでURL変換され、ユーザーIDも自社用にマッピングされて渡されています。自社Webサイトでは別ブラウザーを立ち上げ、メニュー画面を表示します。その間、再認証はしませんが、アクセス制御のためのセキュリティー・ランク情報を自社データベースより読み込んでおきます。基本的にはシンプルかつ一般的な方法ですが、想定できる課題を以下に示します。

(1) 2重ログインの課題

リンクで移った先から、再度のリンクで自社Webシステムにユーザーが戻ってきた場合、最初のブラウザーは画面に残っているので、別ブラウザーが立ち上がることになります。各社システムで2重ログインが許されていないと、2番目の画面は表示されません。

(2) タイムアウトの課題

Webサーバー上のリソースを開放する仕組みとして、タイムアウト処理は必要です。画面上、背景に隠れたブラウザー上のアプリケーションにタイムアウトが発生します。Webデザイン上は、タイムアウトになっているものを、次回のアクセスから内部的に同一ユーザーの再ログインとして、ユーザー操作上シームレスに見える扱いにします。

2.3.4. Webシステムのフレームワーク

Webアプリケーションを開発する上で、フレームワークを設計することは特に重要な作業です（図7）。

昨今では一から設計するのではなく、WebSphere®用の既存の基盤を採用することが多いと考えられています。基盤は、製品またはSIコアとして提供されています。

基盤とは、WebSphere製品をアプリケーションで使う上での、共通機能や部品を提供するものです。基盤によって違いがありますが、例えば、MVCモデルでの画面制御、セッション管理、データベース・アクセス、印刷（PDF生成）、ホスト連携といった実行機能や、画面の開発支援機能との連動で開発保守の生産性を高める機能やログのような問題判別機能を提供しています。

WebSphereに限らず、アプリケーション・サーバー製品は、J2EEに準拠する機能を提供しています、共通フレームワークを提供する機能までは立ち入っていません。W3Cなどの規格の

変化に製品を追随させるためには賢明な住み分けですが、アプリケーションを開発する側に立つと、裸の製品だけではスピーディーな開発はできませんので、アプリケーションに共通な部分を基盤と称するミドルウェアがカバーしている状況にあります。

ユーザーとしては、基盤が提供しているフレームワークの不足部分をさらに補う必要があります。例えば、アプリケーションのサービスの停止と再開を制御したいという要件があれば、サービス制御のための初期化処理をサーブレットのinit()処理に組み込む必要があります。また、トランザクションごとに稼働可否を判定するクラスを作成し、そのメソッドを呼ぶことにもなります。さらに、監視通知や問題判別のためのログの取得も、製品では手薄な部分です。これらは、インフラストラクチャーとして個別開発が必要な項目であり、外部設計が終わると、フレームワークの仕様が決定されます。

要件定義の段階でも、採用基盤とフレームワークを意識して要件を固めていかないと、できないことを約束してしまいます。結果、設計を歪めてしまい、インプリメンテーションに必要以上の工数増を招く可能性があります。意識してフレームワークの研究をしておくべきです。

2.3.5. サーバー負荷分散のCBR機能

本システムのような構成で注意が必要なのは、ゲートウェイがプロキシー・サーバーであるため、HTTPのトランザクションは、すべてゲートウェイの同一IPアドレスを持っていることにあります。つまり、クライアントのIPアドレスはすべて同じに見えるため、IPのスティッキー・ビットを使ったアフィニティー制御は、NDサーバーと複数のWASサーバー間での負荷分散には使えないのです。

負荷分散の代替手段としては、クッキー・アフィニティーを使用したCBR(Content Base Routing)の機能が着目されています。当機能は最新のEdge ServerのWTEのコンポーネントに含まれています。現時点では機能検証に至っていないので詳細は書けませんが、次の2点が課題と考えられます。

- 処理負荷が重くなると考えられるWTEのパフォーマンス評価
- WTE製品ではサポートのないサーバー切り替え、HA機能の代替機能の設計

3. プロジェクトの性質を把握してアウトプットを作成する

3.1. プロジェクトの鳥瞰図を作る

技術論と並ぶ2本柱の一方である、プロジェクトの進め方について全体感を把握してみましょう。特にポイントをデフォルメした表現は、自分自身の理解と関係者とのコミュニケーションのために心掛けましょう。そのためには、以下のような作業が役立ちます。

(1) インフラストラクチャー要件の洗い出し(何をつくるのか:What)

- 処理パターンをつくって、お客様と共通認識に立つ。

(2) インフラストラクチャー要件の洗い出し(どうつくるのか:How)

- インフラストラクチャーを整理する(四つに区分)。
  - #1 Webオンライン:ゲートウェイと接続し、オンライン・アプリケーションが稼働
  - #2 データベース・バッチ:データベースを更新する名寄せバッチ・アプリケーションが稼働
  - #3 ファイル転送:ゲートウェイやVANとの他社接続
  - #4 運用監視:サーバーの監視やバックアップ、変更管理

(3) 開発期間の理解

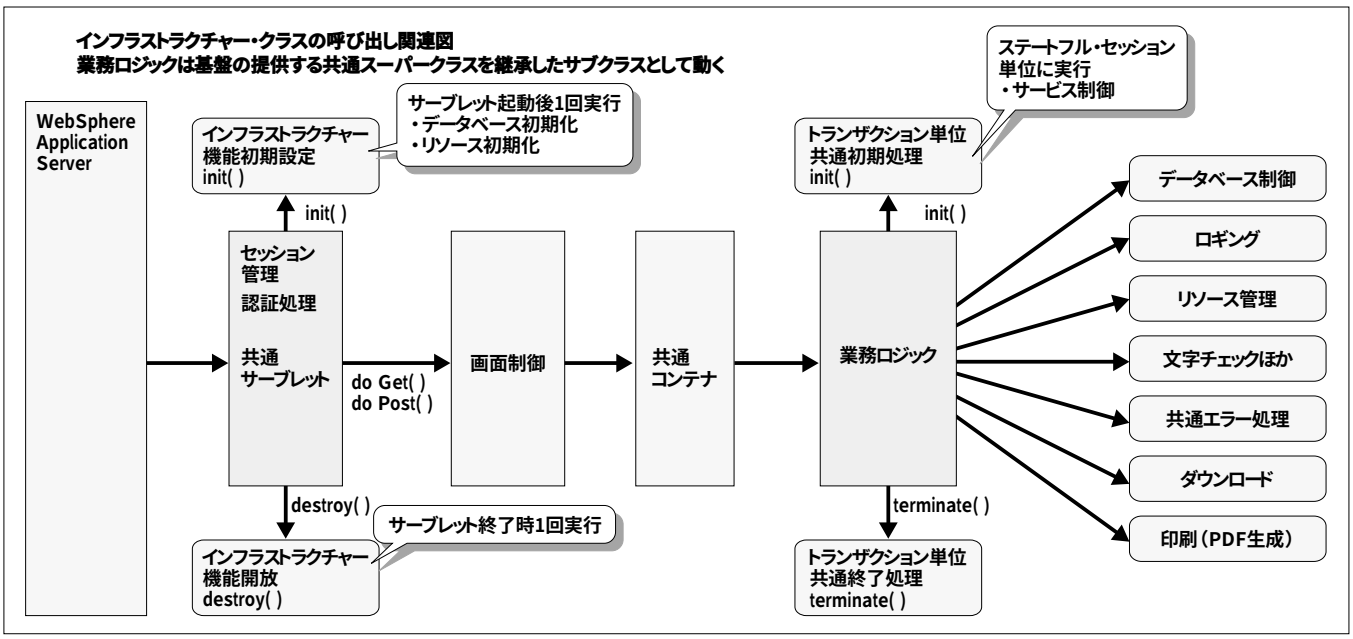


図7. Webアプリケーションのフレームワーク

- 9カ月の短さ→設計フェーズで遅れると取り戻す時間がなくなる。
- 期間を3:3:3に分ける（設計・製作・テストをおのおの3カ月に割る）。
- 対外接続テストの時期がインフラストラクチャー完了のマイルストーンになる。
- インフラストラクチャーの4区分（サブシステム）ごとに平行して動ける体制をつくる。

3.1.1. 要件を網羅する処理パターン図の作成

図8と図9が、処理パターン図のサンプルの一部で、データ・フロー・ダイアグラムの粗いモデルのようなものです。主要なシステム単位に、ディプロイメント・ユニット（ユーザー）とパーシスタンス（データ・ストア）とノード（サーバー）を配置した絵を描いています。処理のパターンを洗い出して、システム・イメージを把握するための一覧表を作る作業です。業務アプリケーションは、

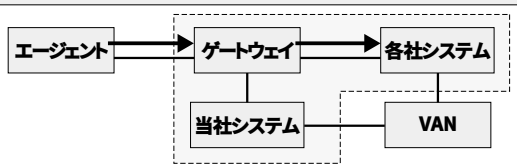
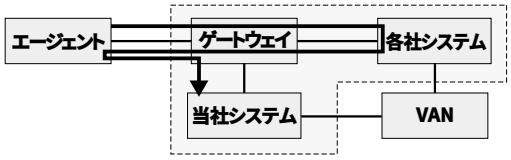
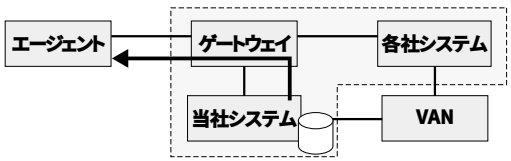
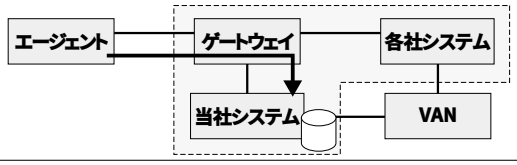
No.	処理パターン	データと処理の流れ	処理内容
1.	ログイン ・各社Webページへ		1.エージェントは、各社Webページにログインするためにゲートウェイにて基本認証を受ける。 ・個人単位のユーザーIDの入力 2.認証を受けると、リバース・プロキシー型の認証サーバーを経由して各社システムへのアクセスが可能となる。 ・セキュアな環境に入る ・唯一のユーザーIDでゲートウェイ配下のWebでアクセス制御を受ける
2.	Webリンク ・各社Webページより当社のWebページへリンク		1.各社のWebページより顧客または契約照会を選択すると、内部的に当社のHPにリンクされる。 2.当社システムは認証機能は持たない。 3.引き継がれたユーザーIDを元に、当社システムではアクセス制御を行う。
3.	オンライン照会 ・検索		1.当社システムの顧客と契約データベースより情報を検索して、表示を行う。 ・絞り込んだデータを印刷処理へ渡す ・絞り込んだデータをCSV形式のファイルに加工する
4.	オンライン入力 ・更新		1.エージェントの操作にて、顧客データの入力や名寄せの更新を行う。

図8. 処理パターン1

注：図中のVAN部分はオフライン・テープ交換による代替法を含む。

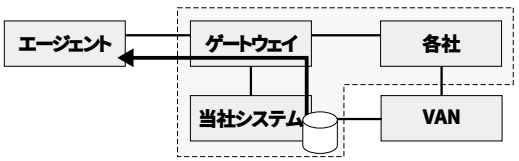
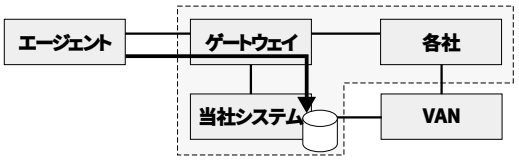
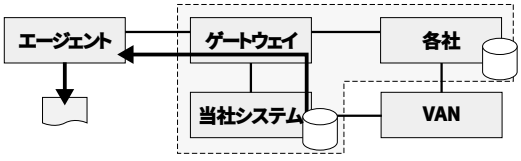
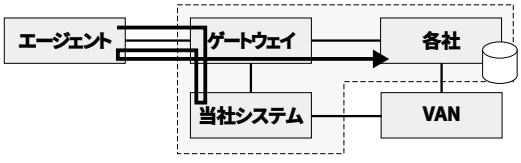
No.	処理パターン	データと処理の流れ	処理内容
5.	ダウンロード		1.エージェントの起動にて、CSV形式のファイルを、HTTP経由でPCにダウンロードする。
6.	アップロード		1.エージェントの起動にて、CSV形式のファイルを、HTTP経由でPCにアップロードする。 ・ウィルス・チェック機能について検討する。
7.	印刷		1.当社システムの印刷は、PDFファイルを介して行う。 ・各社の印刷機能との標準化は行わない。 2.印刷用のPDFファイルを当社サーバーで作成し、エージェントにダウンロードの後、Acrobat Readerにてローカル・プリンターに印刷する。 ・PDFファイルはサーバー上には残さないようにする。 3.ダウンロードされたCSV形式のファイルを元に、エージェントにてオフライン印刷も可能。
8.	他社リンク ・当社Webページから他の元受けWebページへ		1.当社システムでの照会結果より、各社へのWebページへリンクする。 ・リンク元ではないほかの各社Webページへリンク

図9. 処理パターン2

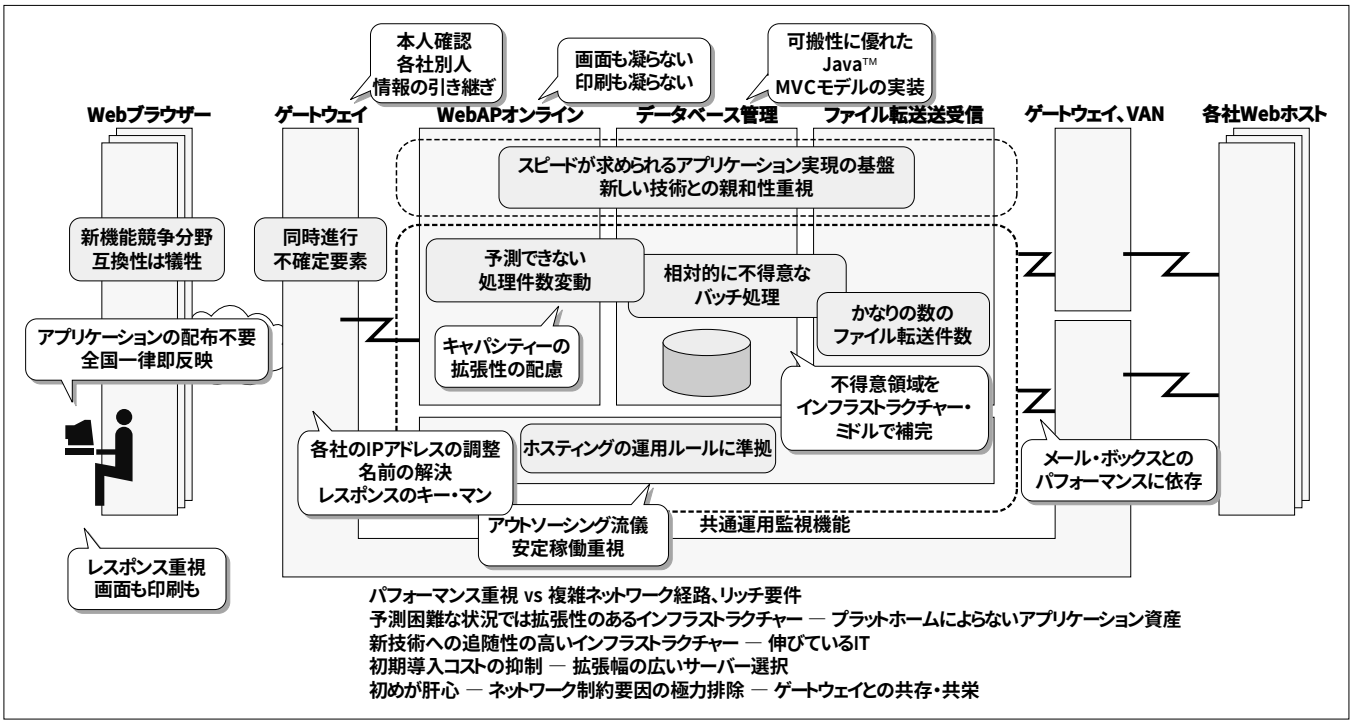


図10. インフラストラクチャー4サブシステムと要件定義時のメッセージ

このパターンのいずれかに乗ることが検証できれば、インフラストラクチャー要件定義としては網羅性のある資料となります。

3.1.2. インフラストラクチャー領域を分かりやすく分割し分担する

ここでは図10の要件定義時の説明会で使用した図を掲載します。

ここでは課題を管理しつつ、前へ行くことを重視しています。例では、現状の課題の評価について、構成要素ごとに主なポイントを伝えようとしています。人は一度に多くのことは記憶できません。そこで、プロジェクトを進める上で、相手に「何を安心し」「何を心配してもらうか」を、その時々のシナリオとして表現することが大切です。

3.2. 長期的なリスクを考える

以下は、技術的側面として、インフラストラクチャーの責任者は常に意識しておく項目です。

(1) お客様が最終的にこだわるもの

プロジェクトのどの局面でも、性能・品質を意識すべきです。その細目は各システムで異なりますが、今回のサンプル・ケースでは、例として(2)(3)のようなリスクを挙げてみます。それぞれを、リーダーが課題として書き留め、有効なアクションの手が打てる最終期限以前に、評価と判断の時期を設定することが必要です。

(2) インフラストラクチャーのリスク要素

- 他社システム接続
- 短期開発のスキル確保(スキルの育成)
- 実績の少ない製品機能の品質

(3) 技術的なリスク要素

- 複数企業をまたがるシングル・サインオンとアクセス制御
- ゲートウェイのリバース・プロキシー仕様決定
- 各社とのWebリンク方式
- ファイル転送方式の決定と大量データの伝送時間
- データベース設計と名寄せバッチの所要時間

コーチングの手法として、常に質問を繰り返すと、おのずと考えるようになっていきます。長丁場の課題は、ToDoのアクション・プランを立て、繰り返し考えてみることです。

3.3. 差し迫ったリスクへの対処

プロジェクトの立ち上げ時期の最初の2カ月は、準備もないところから始める人のための方法論が必要となります。社内の知的資産データベースや標準化ガイド、身近な事例を参考にしつつも、お客様への提出アウトプットになると、コピーや修正ができるほどの素材は見当たりません。自ら作るとしても、時間との勝負になってしまいます。自分の能力と性能そのものが、プロジェクト・リスクになってしまっていることに気付くはずです。

最初にすることは、活動のシナリオを作ることです。具体的には、成果物を作るためには、何をどの順番で作っていくべきかを考えます。自分の活動指針を持つことと、お客様の気掛かりを解決して安心感を与えることが、活動を円滑に行うコツです。そうすることで余分な活動も発生せず、専念できるようになります。

活動には表裏一体の関係があり、片方だけでは長続きしま

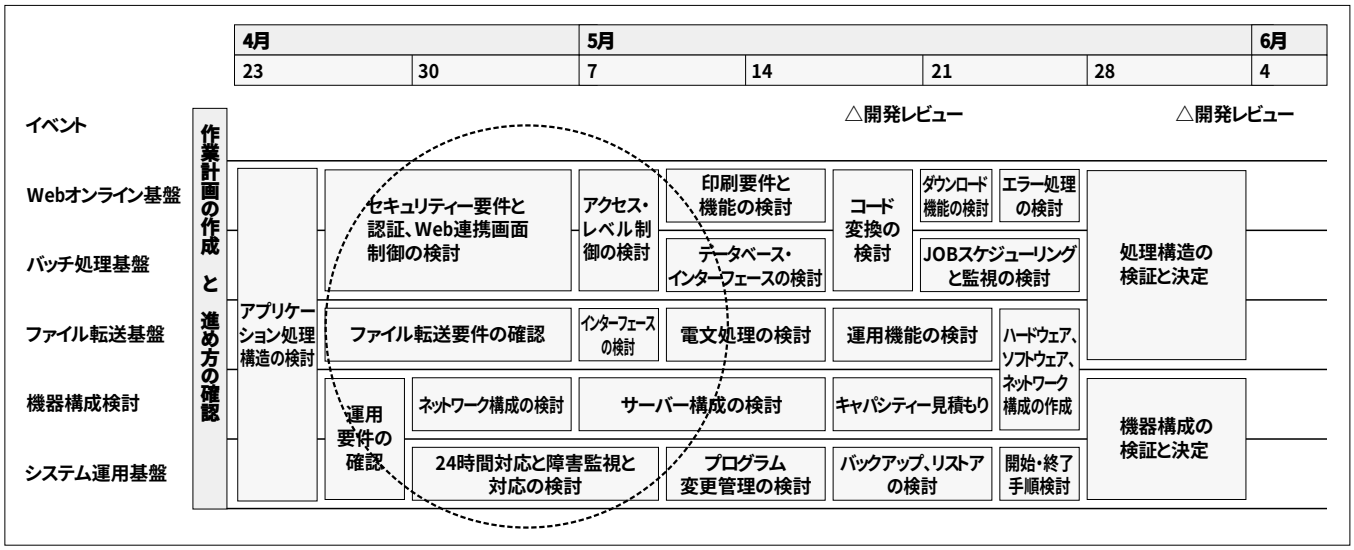


図11. インフラストラクチャー検討スケジュール例

せん。リーダーシップを取り続けるためには、個人の実力を発揮すると同時に安定性も必要です。

(1)表の活動

- 要件定義書の目次を作ること
- 日程計画をつくること
- 課題管理シートを用意する
- 議事録のフォームを用意する

(2)裏番組の活動

- 開発体制のチームづくり
- 契約と社内手続きの完了

図11では2カ月間の活動スケジュール例を載せておきます。四つのサブシステムと装備計画について、おのおのに活動の流れをつくります。議論が必要な問題含みのテーマは、前半で検討するようにしましょう。

日々のアウトプット作成は重要な作業です。お客様の要件の記述や技術の説明が正しく書かれていることは大前提ですが、その上で、自分の意図を伝えるための資料になっているかということも重要です。しかし、アウトプットを作ることに専念してしまい、何のために作っているのかを見失っている場合もあります。以下は、資料作成時に振り返ってもらいたい注意点です。

- 個々の資料では、何を伝えたいのか目的をはっきりさせる
- 技術評価では、3案は比較検討する(代替案を作っておく)
- 比較では、評価軸を決め、メリットとデメリットを評価する
- 図表をうまく使う(絵が描けないのは、理解してない証拠)

3.4. 装備計画をつくる

SIで開発したものを動かすための、ホスティングのハードウェアとソフトウェアの製品を決める活動があります。インフラストラクチャー設計を担当している立場の人は、少なくとも装備構成とキャパシティー計画を示す必要があります。装備を決める要素は以下の点です。

- インフラストラクチャーの基本設計を満たしている
- 重要なリソースには、障害時対応の冗長構成が組まれている
- 初期導入時に満たすべきキャパシティー計画が立てられている
- 装備の拡張性と拡張方法について合意されている
- 運用計画には、定期保守作業の枠がとられている

4. おわりに

本論文のテーマを選んだ理由は、今春、新しいお客様の短期間プロジェクトのインフラストラクチャー開発に携わることになったとき、アウトプット作成の活動指針が見つからずに苦しんだ経験からです。そこで何らかの手順をつくりたいと考えました。

エンジニアである自分にとって、危機に直面したときに使えるパターンをいかに多く持っているかということこそが、自分の値打ちであり、技術面の裏付けが自分のスキルであると考えています。これまでの苦しい経験をバネにして、次のレベル・アップを図りたいと考えています。

(ページ数および表記上の観点から、著者の了解を得て編集部にて手を入れてあります)

[参考文献]
[1] 佐藤 浩司「アーキテクチャーの考え方についての一考察」
2000年度IBMプロフェッショナル論文
[2] Ed Kahan ほか『GSMethod Architectural Thinking』IBM Corp.、2000年
[3] オージス総研『かんたんUML』翔泳社、1999年
[4] 小泉 修『Web技術のすべて』日本実業出版社、2001年