分散システム環境の変化とハイレベル・デザインの必要性に関する一考察

A study on changes in decentralized system environments and the need for high-level design

日本アイ・ビー・エム システムズ・エンジニアリング株式会社
第一システム・センター
ネットワーク&システム・マネジメント・システム部
ICPーコンサルティングITスペシャリスト
宮本 良麿

Yoshimaro Miyamoto
Consulting IT Specialist
Network & System Management Systems
System Center No.1
IBM Japan Systems Engineering Co.,Ltd.

本論文は、最近の分散システム環境における質的・量的な変化と運用形態の変化を整理してみました。今後拡大が予想される分散システムを全社的な観点から、統一的に運用管理する上での原則（ハイレベル・デザイン）の必要性と設計上の考慮点を整理し、さらにはサービス・ガイドの作成など具体的な実施例を紹介していきます。
分散システムの量的・質的な拡大を予定され、より効果的・効率的で柔軟な運用管理システムの必要性を感じている方々に、「そもそも運用管理システムとは何か」といった基本設計を実施するためのヒントになり、設計・構築の参考になれば幸いです。

In this paper I have attempted to summarize the qualitative and quantitative changes that have taken place recently in decentralized system environments and changes in modes of implementation. I examine the need for principles (high-level design) governing the integrated implementation and management of decentralized system, which are likely to expand yet further in the future, from the standpoint of companies as a whole and summarize the points that need to be taken into consideration in connection with design, and I then go on to introduce actual examples of implementation involving the preparation of service guides, etc. I hope to have been able to provide hints to people who feel the need for more effective, efficient and flexible operational control systems and who expect to see a quantitative and qualitative expansion of decentralized systems in the future, and I hope that these hints will prove useful in connection with implementation of basic design that attempts to confront the topic of what exactly operational control systems are all about. I would be happy if my considerations prove to be of assistance to such people in their own design and construction activities.

1. はじめに

ここ数年、汎用機中心のホスト・セントリックなシステムは、いわゆるクライアント/サーバー・タイプのシステムやe-businessタイプに変化しています。本論文では、それを分散システムと総称します。とはいえ、その一語では総称できないくらいに千差万別な形態でもあります。

ホスト・セントリックなシステムから分散システムへの移行を実施した当初は、パイロット的なアプリケーションであり、限定されたユーザーを対象に、限定されたアプリケーションで試行した企業が大部分を占めていました。しかし、その後、システムの利用対象のユーザーが拡大され、処理件数も飛躍的に増加しています。さらにサービス時間帯も拡大され、24時間365日のサービス提供が必要となり処理量の増大をもたらしました。その結果、サーバーの増強やサーバー台数の増加が必要とされています。これらは量的な拡大です。

また、検索を中心とした情報提供型のアプリケーションから、情報の更新を伴う基幹系の業務やWebを利用して直接的に社外のお客様との接点となる業務など、分散系のシステムで実行される業務の種類も増加しています。いわゆるミッション・クリティカルなアプリケーションが分散システムで実施され、可用性の向上やユーザー・サービスの向上が必要となってきているのです。これらは、質的な拡大です。

さらに、今日の環境変化の速さに対応するために、分散システムでは短期間でのアプリケーションの開発やサービス・インが期待されています。また、分散システムの質的、量的な拡大に加えて、最近の分散システムを運用管理の観点から見ると、運用形態にも大きな変化が現れています。さらに、複数のバージョンの管理も含めて、マルチプラットフォーム、マルチミドルウェア、マルチベンダー環境では、運用部門に必要となるスキルやノウハウは拡大の一途となるでしょう。このような運用上の課題に対応するために、分散システムの運用管理でもアウトソースの利用が有効な選択肢といえます。

2. 分散システム環境の変化と対応

2.1. 分散システムの質的な変化と対応

分散アプリケーションでは、次のような質的な変化、アプリケーション属性の変化が見られます。

2.1.1. 対象業務の変化

分散システムで提供されるアプリケーションは、当初はパイロット的で、情報提供型のアプリケーションが中心でしたが、最近ではトランザクション処理型のアプリケーションになりました。つまり直接的に業務連携するような処理が組み込まれているのです。いわゆる「ミッション・クリティカル」なアプリケーションが分散システム環境で実行されています。その結果、「応答時間の悪さが直接的にビジネスの機会損失につながる」とか「セキュリティー上の問題が直接的に社会的な評価の低下につながる」など、影響範囲の大きさ、深刻さは、従来とは比較になりません。

これらの変化に対応するために、アプリケーションには、高い可用性や迅速な応答時間、セキュリティーの確保が重要となります。具体的には、サーバーの2重化や代替経路の確保などの各種の冗長構成の実現や障害の検知と切り替えの自動化などにより、可用性を確保することができます。また、応答時間の劣化を迅速に、かつ事前に検知するために、しきい値監視による性能状況のリアルタイム監視を行います。さらに、しきい値超えなどの性能異常が発生した場合には、操作員や管理者に対して自動通知も行います。もちろん、発生した性能異常に対応するための施策の準備も必要です。例えば、拡張性のあるハードウェアを選択しておくことや定期的で定常業務化されたキャパシティー計画の実施や不要不急なプロセスの停止、そしてバックアップ・サーバーの利用などの一時的な対応策を考慮しておくことも必要です。

また、セキュリティーの確保のためには、ファイアウォールやコンピューター・ウィルス対策、侵入検知（IDS：Intrusion Detection System）などのネットワーク・セキュリティーの整備とユーザー管理、認証、アクセス制御などのサーバー・セキュリティーの整備も必要です。

2.1.2. サービス時間帯の変化

対象の業務の変化に伴い、サービス時間帯も変化してきています。直接的に業務に関連するシステムであればあるほど、関連する部門や企業、お客様も多様になり、どうしてもサービス時間帯は拡大されることになります。海外との取引に関連するような場合には、各国の時差にも影響され、従来の日中だけのサービス時間帯では対応できません。現実に、ある種の一般消費者向けのWebサービスでは、23時以降にトランザクション量がピークになるような現象も発生しています。Webベースでのアプリケーションやコンビニエンス・ストアなどからの各種サービスを例に出すまでもなく、24時間365日のサービス提供への必要性は今後ますます高まっていくでしょう。

このような要求に対応するために、分散システムはオンライン保守の機能や冗長構成を利用した保守作業の実施が必要と

なります。日常的に実施される保守作業の一つにバックアップの取得があります。データベースのバックアップでは、バックアップ・データの整合性を確保するために、データベースを停止した状態でバックアップを取得することが一般的です。この方法ではデータベースの停止中は、このデータベースを利用するアプリケーションはサービスを提供することができません。ここで必要とされるものは、データベースの停止時間を最少化し、データベースの停止がアプリケーションやユーザーからは見えなくする工夫や、データベースの停止を必要としないような「オンライン・バックアップ」の機能などです。

ソフトウェアのバージョン・アップやパッチの適用も日常運用として必要な業務の一つです。ソフトウェアのバージョン・アップやパッチの適用に際しては、システムのリスタートが必要となる場合も少なくありません。このような場合でも、サーバーの2重化などの冗長構成により、停止時間を最小限に抑えることができます。冗長構成の片側ずつのバージョン・アップやパッチの適用を行うわけです。さらに、分散システムのプラットフォームによっては、定期的なシステムのリスタートが推奨されている場合もあります。メモリー・リークなどの現象で、次第にメモリーが圧迫され、システム停止に陥ることを防ぐためにです。本質的には、プラットフォームなどの機能が改善され、解決されるべき問題ですが、現実にはある程度の定期的なシステム・リスタートを実施することになります。これも冗長構成を採用することにより、サービスの停止時間を最少限に抑えることができます。

2.1.3. 対象ユーザーの変化

対象ユーザーの変化を、量的な面ではなく、質的な面からとらえると、対象ユーザーのスキルやノウハウ、使用環境、企業からの働き掛けの有効性などに変化があります。

分散システムのユーザーは、企業内の特定の部門から企業内の全部門へ拡大され、さらに取引先などのほかの企業へと広がっています。また最近のWebベースのアプリケーションでは、一般消費者がシステムのユーザーとなっているので、一般消費者がユーザーであると想定すると質的な変化は理解しやすいと思います。

当然、ユーザーが行いたいことは、情報の取得や注文などのアプリケーションが提供するサービスの利用です。分散システムの利用が目的ではありません。従って、ユーザーにはシステムの利用に関するスキルやノウハウは期待してはいけません。ユーザーが特定できれば、研修などの実施により、システムの利用方法をガイドし、「標準的な使い方」や「利用上の常識」などを紹介できますが、一般消費者を対象とした場合は、これらの研修やガイドは不可能です。つまり、アプリケーションを設計する段階では想定されていないような操作や処理を実行してしまう可能性もあるのです。例えば、サーバー処理の途中で端末を電源オフするとか、不用意にキーボードやマウスを操作するなどです。アプリケーションとしては、これらの状況にも対応できる必要があるでしょう。また、親切なヘルプの提供や分かりやすい画面構成と画面展開も必須の要件です。しかし、同時に親切なヘルプや画面展開が、冗長でくどい「おせっかいな」ヘルプや画面展開にならないように注意することも重要です。また、各種の問い合わせ窓口を一本化し、分かりやすくすることも重要でしょう。

ユーザーがサービスを利用する端末などの環境もさまざまです。同時に稼働しているソフトウェア環境やサーバーへの接続の形態や経路も多種多様です。また、対ハッカー用に、セキュリティーの確保は重要な項目の一つです。さらに、一般消費者がユーザーである場合は、緊急のサービス変更や障害発生時に、企業から有効な連絡方法を持っていないこともあります。Webベースのアプリケーションでは、「Sorry Server」を利用して、接続したユーザーに状況を連絡するような仕組みを構築できますが、企業側から能動的に働き掛ける手段が乏しいのが現実です。

以上のような対象ユーザーの質的な変化に対しては、アプリケーション上の対応策が幾つか考えられ、運用管理としてはセキュリティーの確保や組織の整備などが考えられます。

2.2. 分散システムの量的な変化と対応

分散システムは、質的にも量的にも変化しています。量的には次のような変化が見られます。

2.2.1. 対象ユーザーの拡大

前述したように、対象ユーザーは特定ユーザーから一般消費者へと質的にも大きく変化しています。これは同時に量的にも増大していることを意味します。従来のように特定の部門や特定の企業が対象であれば、ユーザー数の想定や限定が可能でした。従って、ユーザー数の増加も比較的容易に予測できました。その結果、最大ユーザー数を想定した資源の確保やユーザーの登録・保守などの運用も可能でした。しかし、最近のWebアプリケーションでは、これらの予測や限定は極めて困難であり、最大数の想定などは不可能です。

同時に利用しようとするユーザー数に対応できるだけの資源や運用体制が十分でなければ、応答時間の悪化などのサービス・レベルが低下してしまいます。場合によっては、システム資源の不足によるサービス全体の停止にもつながりかねません。これはビジネスの機会損失です。しかし、アプリケーションのサービス・イン当初から、世界中の人口を対象とするようなユーザー数を想定した資源を確保することは現実的ではなく、運用しな

がらチューニングするというアプローチにならざるを得ません。

対応策としては、アプリケーション的には「Sorry Server」を設置し、過負荷が原因でサービス全体が停止することを避け、かつ、ある程度のお客様への情報提供ができるような仕組みが必要です。また、一時的なキャパシティーの増強ができるような構成や、将来的なキャパシティーの増強が実施しやすい拡張性のあるシステム構成を採用することが重要です。運用管理としては、しきい値を利用したリアルタイムの性能監視を行い、プロアクティブな対応を実施しましょう。また、性能データの取得と蓄積を行い、比較的短期の周期で定期的に傾向分析を実行し、将来的なキャパシティー計画を立案・実施することも必要です。

2.2.2. アプリケーションの処理量の拡大

いったんサービス・インされた分散アプリケーションは、ユーザーの拡大や提供するサービスの内容や使い勝手の良さなどから飛躍的に処理量が増大する場合があります。それらは処理量の増大に応じて、サーバーの増強や追加、運用の変更などを行う必要があります。資源の不足などによる処理の不具合や応答時間の悪化もビジネス機会の損失につながります。処理量の増加傾向をいかに迅速に、かつ正確に把握できるか、また、いかに対応できるかが重要な点でしょう。

対応策としては、負荷の増大への対応という観点では、前項目の対象ユーザーの拡大と同様です。ただし、アプリケーション的にはサーバーの分割を想定した機能を持たせることが望ましいといえます。すなわち、複数のサーバーでサービスを提供しても、サービス全体としては整合性が確保されるような設計です。ユーザー情報の整合性、データベースやトランザクション処理の整合性などアプリケーションとしての考慮点は少なくありません。システムの構成としては、一時的、および将来的なキャパシティーの増強が実施しやすい拡張性のあるシステム構成を採用することが重要です。運用管理としては、しきい値を利用したリアルタイムの性能監視とプロアクティブな対応を実施し、性能データの取得と蓄積による比較的短期の周期でのキャパシティー計画を立案・実施することが必要です。

2.2.3. アプリケーションの種類の拡大

分散アプリケーションの特長の一つに、その開発の速さがあります。以前のような数年の期間をかけたアプリケーション開発は少なくなり、着想から設計・開発・本番までが数カ月という分散システムも多くなっています。その身軽さから、企業内の複数の部門がそれぞれ独自の計画に従って、システムの設計・開発を行う場合もあります。結果、企業全体としては、多くの分散システムが並行して設計・開発され、運用されることになります。1年間で数十の分散システムがサービス・インされるケースもあります。

対象ユーザーの拡大やアプリケーションの処理量の拡大とは異なり、アプリケーションの種類の拡大は、運用管理上の直接的な課題となる可能性があります。分散システムは、短期間で開発するために、開発段階では、その後の運用管理が十分に考慮されていない場合も少なくありません。考慮されている場合でも、選択したプラットフォームやミドルウェアが標準として提供している管理機能を利用し、その範囲内での管理機能（の一部）を実現している場合が多いのです。

このような状況で分散システムの種類が増加すると使用するプラットフォームもまちまちで、管理の仕組みや管理業務の内容、使用するツールも異なってしまいます。つまり、運用負荷の増大をもたらすことになるのです。

対応策としては、分散システムのアプリケーション設計の段階で運用管理を考慮した設計をすることが挙げられます。しかし、現実には運用管理上の要件がアプリケーション側から提示されることは、まれです。従って、運用管理側でひな型となるベスト・プラクティスを作成し、アプリケーションを設計する側に提示しましょう。そのためには、運用管理の業務を標準化し、マルチプラットフォーム、マルチミドルウェア、マルチベンダー環境で使用できるものにする必要があります。

ここで、システム管理のハイレベル・デザインの必要性が再認識されることになるのです。これらの変化のうち量的な変化の傾向を早く正確に把握することが重要です。また、アプリケーションの種類の拡大に対しては、どこまで標準化が実現できるかがキーです。

2.3. 運用形態の変化と対応

さらに最近では運用形態そのものへの変化が見受けられます。

初期の分散システムは、開発部門がそのまま運用管理をしている場合がありました。このような運用形態を取らざるを得なかった理由は幾つか考えられます。例えば、パイロット的なアプリケーションであり、アプリケーション要件の実現のために短期間で設計・開発したため、運用管理上の考慮が少なく既存の運用の仕組みに引き継ぐことが困難であったこと。さらに、使用者も限定されており、障害発生時にはパーソナル・チャネル経由での問い合わせやサポートが中心であったことなどです。一方、既存の運用管理組織は、汎用機中心の運用を想定しており、分散系のスキルや運用ノウハウの蓄積が十分でありませんでした。また、分散システムを意識した運用引き継ぎや移管の仕組みがなかったのです。

これらの状況は、分散システムが質的にも量的にも変化してきた今日でも同様です。例えば分散システムは個別に開発して

きました。そのため、運用管理も個別となり、全社的な運用組織への引き継ぎができません。また、このような状況を想定した運用引き継ぎや移管の仕組みができていないのです。さらには分散システムのマルチプラットフォーム、マルチミドルウェア、マルチベンダーに対して、運用組織がスキルやノウハウなどで対応できていませんでした。短期間の開発であり、手順書などが整備できていないため、開発者にしか分からない内容が多いなどの理由もあります。

これでは、今後の分散システムの増大に対しては、効率的な運用管理は実現できません。開発者が既存の分散システムの運用管理にワークロードを費やすことになり、新規のアプリケーションの設計・開発を行うことが困難になります。また、汎用機から分散システムにアプリケーションが移行されるにつれて、従来の運用組織の役割が減少してくることになります。企業としては、効率的な組織の運営という観点からも、従来のような開発部門による分散システムの運用管理から、運用組織による分散システムの運用管理へと移行する必要があります。

このような状況から、開発組織による運用管理から、運用組織による運用管理へと運用形態も変化する傾向にあります。ここでの運用組織としては、自社の既存の運用組織、自社の新規の運用組織、アウトソースの利用などの形態があります。

2.3.1. 自社の既存の運用組織での運用

汎用機と分散システムでは、その運用方法や考え方に相違があります。しかし、分散システムの運用標準を定め、既存の運用引き継ぎや移管の仕組みに分散システムを持たせることにより、既存の運用組織での対応が可能となります。その際に、分散アプリケーションの運用の標準化を行うことが必須です。

同時に、分散系のスキルを運用組織が蓄積する必要があります。開発組織と運用組織間での協業や人材交流、ローテーションなどの全社的な取り組みも必要です。汎用機と分散システムでは、システム的な「文化」の違いもあり、「汎用機を運用する感覚で分散システムを運用する」ことには困難な場合が多く、汎用機の運用管理に慣れた組織では、分散システムの運用管理への対応には時間がかかるかもしれません。

一方、最近の分散システムの質的な変化のように、従来の汎用機並みの運用管理の質が要求される場合もあり、このような場合には、汎用機での運用ノウハウが有効利用できます。分散システムでの運用が標準化されれば、蓄積すべき分散システムのスキルも限定され、自社の既存の運用組織での運用はより実際的・容易になります。

2.3.2. 自社の新規の運用組織での運用

汎用機の運用を詳細に標準化し、組織としての業務フローやそれに伴う処理形態・連絡体制・役割分担などがきめ細かく規定されていると、汎用機の運用と分散システムの運用とのギャップが大きく、既存の汎用機を想定した運用組織では、分散システムの運用が困難と思われる場合も考えられます。このような場合には、分散システムの運用管理のために、新規の運用組織を構成することも一つの現実的な解決策です。

新規の組織を構成する方法の利点は、既存の汎用機を想定した仕組みや業務フローにとらわれずに、分散システムだけを意識した処理を構成することができる点にあります。分散システムのプラットフォームやミドルウェア、運用管理ツールが提供する運用管理機能は、汎用機に比較すると、まだ成熟していません。従って、運用管理の業務としては、ツールで実行できる部分の設計と運用上の人間系で対応すべき部分の設計も重要です。分散システムでは、人間系による対応の占める位置付けは、決して小さくはありません。従って、運用担当者には、汎用機の運用以上に、分散システムのスキルや運用ノウハウが必要となる可能性があります。

これらの状況を踏まえ、開発段階からの開発担当者と運用担当者との打ち合わせやフェーズごとのチェック・ポイントの実施、運用手順書や引き継ぎの手順書の内容やレベルの規定、運用担当者への各種の研修の実施などの項目を、分散システムを前提とした内容として、開発から運用への引き継ぎ条件として規定する必要があります。

新規の運用組織を利用する場合でも、分散システムの運用を標準化することの必要性は変わりません。個別の分散システムごとの個別管理であれば、分散システムの拡大に伴い、新規の運用組織が肥大化し、要員がいくらいても運用が回らないことになるからです。

2.3.3. アウトソースの利用

最近の「本業回帰」の流れは、分散システムの運用管理においても同様です。運用管理の業務を社外に委託（アウトソース）することも少なくありません。アプリケーションの立案・設計を自社で実施し、開発・運用は社外に委託する方式です。この場合でも、分散システムごとの個別運用では、運用をアウトソースする際にも個別運用としてのアウトソースとなり、割高です。効率的な運用にはならないでしょう。分散システムの運用業務が標準化されていれば、全社的に標準的な運用業務のアウトソースであり、スケール・メリットも期待できます。

今日では、汎用機関連の運用に関しては、アウトソースが検討対象となっている場合が多々あります。一方、分散システムは、自

表1. 分散システムの変化の整理

変化の分類	変化の内容	要件・課題	実現方法
質的な変化	対象業務	• 高可用性 • 迅速な応答時間 • セキュリティー	• 冗長構成・2重化 • 性能監視／分析 • 各種のセキュリティー機能
	サービス時間帯	• 24時間365日	• 冗長構成・2重化 • オンライン・バックアップ • オンライン保守
	対象ユーザー	• 使い勝手の良さ • セキュリティー • 情報連絡方法	• ヘルプ機能・画面設計 • 各種のセキュリティー機能 • サービス窓口組織
量的な変化	対象ユーザーの拡大	• 過負荷への対応 • 運用しながらのチューニング	• 「Sorry Server」 • 拡張性のあるシステム構成 • 性能監視 • キャパシティー計画
	アプリケーション処理量の拡大	同上	同上 • 複数サーバーの利用
	アプリケーション種類の拡大	• 運用の効率化	• 運用の標準化 • 設計・開発段階からの運用への考慮 • ハイレベル・デザイン
運用形態の変化	既存の運用組織	• 開発/運用組織の分離 • 運用の効率化	• 運用の標準化 • 開発/運用組織の協業 • ハイレベル・デザイン
	新規の運用組織	同上	同上
	アウトソース	• 開発/運用組織の分離 • 運用の効率化 • 本業への回帰	• 運用の標準化 • 管理業務の位置付けの明確化 • 管理業務評価基準の作成 • ハイレベル・デザイン

社の運用組織に運用スキルやノウハウの蓄積が少なく、今後のテクノロジーやスキル変化が激しいと想定されます。従って、分散システムについては、汎用機以上にアウトソースの有効利用が、今後ますます必要となってきます。

運用管理業務には、社内的な調整やアプリケーションの設計・開発に関連する部分もあります。また、セキュリティーの観点からの考慮も必要です。アウトソースを有効に利用するためには、自社内で実施すべき業務とアウトソースすべき業務の明確化が必要です。

その際の考慮点としては、運用管理業務の標準化を推進することです。業務の標準化により、個別の分散システム単位ではなく、全社的な分散システムとしての運用管理業務が評価でき、アウトソースの対象としての適正を統一的に判断できます。さらに管理業務の評価内容も標準化してみます。アウトソースを利用する際には、サービスの質を評価するためにサービス・レベルを締結します。具体的なサービス・レベルの内容や目標値は、アウトソース提供者との交渉で決められることになりますが、その際にそれぞれの管理業務ごとの評価項目／基準があることが望ましいでしょう。

2.4. 分散システムの変化の整理

これまでの議論を表1に整理してみました。

3. ハイレベル・デザインの概要

本論文では、ハイレベル・デザインを実施する方法論は、SMFD-TE（Systems Management Framework Design-Template Edition）をベースとしたハイレベル・デザインを想定しています。SMFD-TEをベースとした具体的なサービスとして、Tivoli®サービス方法論があります。

3.1. 成果物の一覧

SMFD-TEをベースとしたハイレベル・デザインの成果物は以下のものが想定されています（表2）。以下に分散システムの変化への対応として、ハイレベル・デザインを実施する際に、重要となる成果物を選択して紹介します。

3.2. 基本方針（Guiding Principles）

基本方針は、より上位の方針を受けて、運用管理の考え方の原則を規定していきます。たたき台とすべきひな型があり、このひな型に基づいてお客様の要件に合うものを採用し、当てはまらないものを落としていきます。また、不足するものがあれば追加します。ひな型となる基本方針は、サービスや組織、評価、セキュリティーなどのカテゴリーに分かれていて、実際の設計ではカテゴリーごとの検討を行うことにより、議論の焦点を集中させることができます。これで比較的短期間で基本方針を作成することが可能となります。

表2. 成果物の一覧

成果物	内　容
基本方針（Guiding Principles）	ハイレベル・デザイン全体の基本方針となる原則。基本方針は複数のカテゴリーに分類され、それぞれの基本方針ごとに選択すべき理由、選択した場合の影響・効果と受容度の評価が行われる
管理モデル	企業全体での運用管理システムの構成モデル。運用管理に関する制御の場所、実行の場所、および情報の流れから判断し、ひな型となる四つのモデルから選択する
サービス・ガイド	運用管理業務をユーザーの観点からサービスとして整理した記述。サービスの定義、サービスを実現するためのアクティビティーの流れ、役割、およびツールを記述
プロセス・ガイド	運用管理業務をシステムの観点からプロセスとして整理した記述。プロセスを構成するアクティビティーやアクティビティーを構成するタスクのレベルでの詳細な記述
組織ガイド	アクティビティーを実行する際に必要となる役割とその役割を実行する上で必要となるスキルに関する記述
ツール・ガイド	アクティビティーを実行する際に必要となるツールとそのツールが提供すべき機能に関する記述

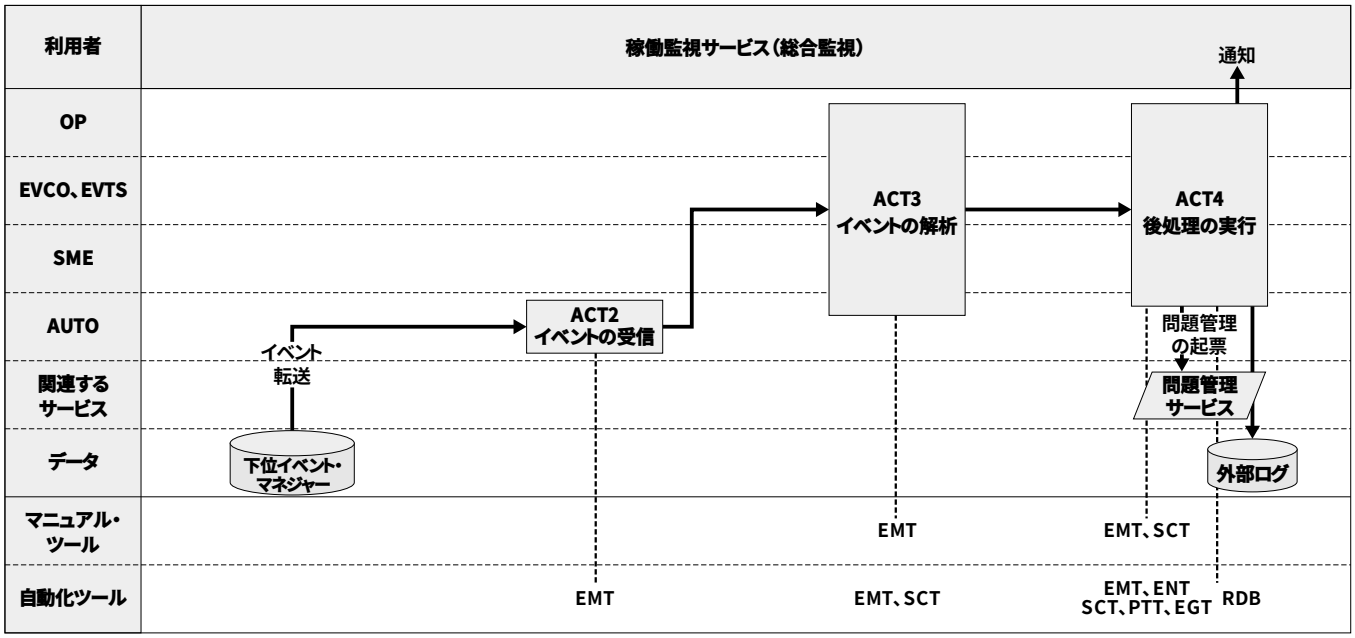

図1. サービス・ガイド

3.3. サービス・ガイドと組織ガイド

サービス・ガイドの中心的な記述内容は、管理業務のフローです。このフローは、LOVEM（Line Of Visibility - Enterprise Model）の手法にのっとった形式で記述されています。大きくは四つの観点からの記述です（図1）。

まずは、LOVEMチャートの中心であるフロー図の部分です。これは、一つのサービスを提供するために必要となる一連のアクティビティーとそれらの相互連携を示しています。

2番目の観点は、利用者とそれ以下のシステムとのインターフェースです。上記の例では、利用者と「OP」の間の線をLOV（Line Of Visibility）と呼び、利用者から見て、どのような接点があるのかを示しています。システムや運用組織としては、多くのアクティビティーを実行している場合でも、ユーザーは見えていません。そこで運用組織のVisibilityを向上させたい場合の基礎資料としても利用できます。

3番目の観点は、フロー図の左側に記述されている「OP」などの役割です。これは、それぞれのアクティビティーを実行する際の役割を表しています。

4番目の観点は、フロー図の下側に記述されているツールです。これは、それぞれのアクティビティーを実行する際に必要となるツール、もしくはツール機能を表しています。

従って、上記の例では、例えば「ACT2 イベントの受信」というアクティビティーは、AUTOという役割の人が（実際にはAUTOはシステムによる自動処理であるが）、EMT（Event Management Tool）というツールを利用して実行し、その結果は「ACT3 イベントの解析」というアクティビティーに連携します。この間では、直接利用者との接点はないということが分かります。

また、組織ガイドは、サービス・ガイドで定義された役割の整理と現状の組織への対応、今後の組織への提言があります。現状の組織への対応では、本来の組織のミッションと実際に行われている役割との関連付けや整合性を吟味していきます。本来のミッションとは異なる役割を実施しているような場合には（このような場合が少なくないのですが）、その役割は本来的にはどの組織で担うべきであるかを議論し、決定します。既存の組織のあり方や新しい組織の形態を提言することになります。

4. ハイレベル・デザインの必要性と適用上の考慮点

4.1. ハイレベル・デザインの必要性

2章で記述した分散システムの変化のうちで、ハイレベル・デザインの必要性や意義に直接関係するのは、アプリケーションの種類の拡大や運用形態の変化です。ほかには、運用管理業務の内容やきめ細かさへの影響はありますが、直接的には関係しません。極端な場合、きめ細かなサービスが要求されるミッション・クリティカルなアプリケーションを分散システムで稼働させる場合でも、それが運用管理すべき唯一のシステムであれば、技術的な困難さは考慮すべきですが、個別対応でも運用はでき、ハイレベル・デザインを実施する必要性は少ないのです。これに反して、ミッション・クリティカルではないアプリケーションが稼働する分散システムであっても、分散システムの個数が多いと、分散システムごとに個別の管理をするような運用管理では、運用負荷が過大となり、運用に支障を来します。従って、ハ

イレベル・デザインを実施して、標準的な運用管理を規定する必要があります。すなわち、ハイレベル・デザインとは、運用管理業務の基本方針、業務フロー、役割、ツール機能などをハイレベルで規定するものといえます。その結果、マルチ環境での複数の分散システムに共通の規定をすることで、分散システムの種類の増加に対して、統一的なデザインを提供します。また、管理業務や役割の分析によりアウトソース化が必要な管理業務の洗い出しが可能となり、より効率的で柔軟なシステムの運用管理が可能です。

例えば、具体的な製品への依存性が高い詳細設計や設定手順のレベルでは、使用するプラットフォームや製品が異なってしまうと適用できません。極端な場合、同一製品であっても、バージョンが異なると適用できなくなる可能性もあります。ハイレベル・デザインであれば、これらのプラットフォームや製品への依存性がなく、全社的な標準として設計することができます。

以上のようにハイレベル・デザインを実施することにより、その後の分散システムの拡張や運用形態の変化に柔軟に対応することが可能です。

4.2. ハイレベル・デザインの実施上の考慮点

ハイレベル・デザインを実施する上で、一般的に指摘されている考慮点は、お客様を含めた体制やメンバーの選択、プロジェクトの中での役割分担を明確化することです。ここでは実際のプロジェクトを通じて筆者が感じたことを追加します。

まずは、できた成果物が実用に耐えるものでなければなりません。ハイレベル・デザインというと「非常に整合性の取れた美しい概念図を記述するが、現実のシステムに実装することは困難、または不可能」というようなイメージを持つ人もいると思います。それぞれが独自の特徴を持つような分散システムでは、統一的なハイレベル・デザインは不可能であるという意識です。このような人たちには、複数の分散システムに対して統一的なハイレベル・デザインは可能である、と理解してもらいましょう。少ないながらも国内での実績もあり、海外ではより多くの実績があるのです。

しかし、ハイレベル・デザインを実施する際には、上記のような意識のメンバーがあるかもしれないことを想定し、実用に耐える設計を意識する必要があります。さらに、マルチプラットフォーム、マルチミドルウェア、マルチベンダー環境を想定し、十分な抽象性を持つようなデザインを実施する必要もあります。もちろん、分散システムの属性が大きく異なり、複数のモデルを含むようなハイレベル・デザインを設計することもあるでしょう。それらも含めて、抽象化されたデザインであり、分散システム個別の設計とは目的が異なるのです。一つの典型的な分散システムをパイロット的に想定し、ハイレベル・デザインを実施することも実際的なアプローチですが、その場合、重要なことは、そのパイロット的な分散システムだけの運用管理を目的とした設計をしてはいないことです。全社的なハイレベル・デザインであるとの位置付けであり、成果物は全社的な標準としてその後の分散システム管理の指針とならなければ意味がありません。このためにも方法論が提供するひな型や過去のプロジェクトでの成果物が有効に利用できます。

4.3. 基本方針作成上の考慮点

基本方針を作成する上での考慮点の一つに、個数への考慮があります。ひな型として提供されている基本方針は、8個のカテゴリーに分かれ、40個以上あります。これらの基本方針から取捨選択もしくは追加するわけですが、筆者の経験では10個程度を目安として議論し、基本方針を採用した方がよいでしょう。また、ひな型から取捨選択する場合には該当しませんが、新たに基本方針を追加する場合には、基本方針の記述内容は、設計への影響を与えるだけの明確な記述が必要です。

採用を決めた基本方針に対しては、それぞれの効果と社内的な受容度（アクセプタンス）を評価します。この際の効果や受容度の多くは、参加メンバーの所属や立場、経験、背景により異なります。これらを調整するわけですが、一般的には効果の大きいものは受容度が小さく、実現へ向けての障害も大きいといえます。効果が大きく、受容度も高いものは既に採用されている場合が多いからです。

個別の障害を取り除くための具体的なアクション・プランの作成には、深入りしてはなりません。これらの議論のための提言やたたき台の提示を行うことは有意義ですが、具体的な解決策の検討や決定はハイレベル・デザインの範囲を超えています。独立した一つのプロジェクトとする必要があるほどの内容を含むこともあります。ハイレベル・デザインとは、基本方針を策定し、効果と受容度を評価した上で、分散システムの運用管理の標準的なデザインを行うことなのです。限られた時間やリソースの中で、プロジェクトの目的やスケジュールを維持することは重要なプロジェクトマネジメントの項目です。

4.4. サービス・ガイド作成上の考慮点

サービス・ガイドは、ひな型を参考にして、お客様の環境や要件に応じて作成できます。この際、ハイレベル・デザインの実施上の考慮点で述べたように、十分な抽象性と十分な具体性・実現可能性を持つ必要があります。

まず、役割は組織ではなく、また、具体的な特定の個人を示してはいないということが挙げられます。この役割は、複数の要

員が実行する場合もあれば、ある要員が複数の役割を兼任する場合もあります。また、役割をアウトソースし、実行する要員が社内的には「見えない」場合も考えられます。その利点は、現実の組織のしがらみや先入観から離れて役割やアクティビティーの分析ができることにあります。

組織の変更に対して柔軟に役割のアサインも可能になります。将来的には、役割単位でのアウトソースも想定できるでしょう。ついで、ツールの記述での考慮点は、ツールの記述は必要な機能の記述であって、具体的な特定の製品を記述するものではないことにあります。例えば、EMT（Event Management Tool）は、イベント管理ツールであり、TEC（Tivoli Enterprise Console®）などの特定の製品を前提とするものではありません。従って、ツール機能としては、イベントの受信機能とかイベントの分析機能という記述であり、一般ツール（Generic Tool）と呼ばれています。具体的な製品ではなく、一般ツールによる記述をしているために、マルチプラットフォーム環境や異なるツール群を利用する環境でも同一のサービス・ガイドでのツール記述が利用できることになります。上記の記述は、サービス・ガイドが十分な抽象性を持つための考慮点です。

では、十分な抽象性と十分な具体性・実現可能性を、同時に持たせるための考慮点は何でしょうか。ハイレベル・デザインといえども、具体的な運用イメージと、少なくとも一つの製品群を利用して実装できる実現イメージを持つことが必要です。サービス・ガイドにおける役割とは、実際の運用の業務フローが想定でき、役割を担う要員により運用が実行できることであり、そのためには、各役割に想定されているスキル項目やスキル・レベルが実際に習得可能なものであることにあります。できれば、類似した環境での実績があるハイレベル・デザインであることなどを確認することが望ましいでしょう。

また、ツールに対して、具体的な製品群を利用した場合の実装が可能であることも重要です。一般ツールに対して定義されたツール機能が、具体的な製品群の標準機能またはカスタマイズなどにより実現できることや、ツール機能の要件によっては、製品群の標準機能とカスタマイズでの実現が困難な場合には、実績のある作り込みなどの検証ができているかなどを確認すべきでしょう。さらには運用の実現性を確認するためには、管理業務から見た整理と役割から見た整理も必要です。通常は、設計対象の管理業務を選択し、管理業務ごとに必要となる役割を洗い出します。そして個々の役割に対して、必要とされるスキル項目とスキル・レベルを定義します。サービス・ガイドは、このような手順で作成され、管理業務ごとにサービスのフローと関連する役割、ツールが定義されていきます。実現イメージも描きやすく、理解もしやすいでしょう。

ただし、複数の管理業務を設計する場合には、同一の役割が複数の管理業務で必要となることもあります。コーディネーター的な役割や操作員、窓口対応の要員などが、その典型的な例です。この場合、それぞれの管理業務ごとに考えると十分に現実的なスキル要件が、複数の管理業務を通してみると、現実的ではない場合があります。「スーパーマン」ともいえるコーディネーターの存在が前提になってはいないか、結果として、あらゆるプラットフォームのあらゆるミドルウェアやアプリケーションのすべてのスキルが必要とされたり、過度に高いスキル・レベルを期待されたりしてはいないかなどの観点から整理と見直しを行います。つまり、複数の管理業務に登場する役割については、実施すべきアクティビティー、取得すべきスキル項目とスキル・レベル、作業の実施場所などを整理し、現実的な内容であるか否かを吟味することが必要なのです。

4.5. 組織ガイド作成上の考慮点

組織に対するとらえ方は、お客様の文化や参加メンバーの所属や立場で異なります。組織ガイドを作成するためには、どのような内容のガイドを作成するかをお客様と確実に合意する必要があります。

表3. ハイレベル・デザインの考慮点

項目	考慮点	内容、備考など
ハイレベル・デザイン全般	体制・メンバーの選択、役割分担の明確化など	プロジェクト実行上の一般的な考慮点でもあり、ここでは議論しない
	十分な抽象性の確保	• マルチ環境での適用が可能なこと • 将来の環境の拡張に対応できること • ひな型や実績の有効な利用
	十分な具体性・実現性の確保	• 実績の調査、確認 • パイロット的なシステムの想定
基本方針作成	基本方針の個数	• 多過ぎない、少な過ぎない • 10個程度(経験則)
	新規の基本方針の明確な記述	• 玉虫色の記述にしない • 設計への影響のある内容とする
	効果と受容度の評価	• 議論に深入りし過ぎない
サービス・ガイドの作成	役割の記述 （十分な抽象性と十分な具体性・実現性の確保）	• 役割と組織・個人の区別 • 将来的な組織変更やアウトソースも視野に入れる • 具体的な運用イメージの確認 • 必要なスキル項目/レベルの現実性の検証 • 管理業務横断的な整理
	ツールの記述 （十分な抽象性と十分な具体性・実現性の確保）	• 特定の製品群を前提としないツール機能の記述 • 具体的な製品群を想定した実装イメージの確認 • 作り込みなどの現実性の検証
組織ガイドの作成	成果物イメージの確認	• お客様との合意 • 参加メンバーの所属や立場の理解

まずは、組織ガイドの項目と記述レベルの確認を行います。例えば、一般的な役割の分析と必要なスキル項目、スキル・レベルを記述し、現状の組織と役割との対応を分析し、本来の組織のミッションとの兼ね合いを評価していきます。さらには、将来的な組織像を提示し、そこでの組織のミッションと対応する役割を整理します。その際の組織像としては、アウトソースも想定しておきます。

これらの将来像を実現するための計画を提言としてまとめておきます。これらの内容を想定し、お客様における必要性と、参加メンバーのスキルや期待できるワークロードなどを勘案し、期間内に実現できる範囲とお客様と要望とのすり合わせを行います。そこでお客様の要望と必要性とを判断し、プロジェクトとしての現実に作成できるガイドを考えていきます。

4.6. ハイレベル・デザインの考慮点の整理

4章で説明したハイレベル・デザインを実施する上での各種の考慮点を整理してみます(表3)。

5. ハイレベル・デザインの必要性の認識

個別の分散システムに依存しないハイレベル・デザインの必要性は、概念的には議論の余地はありませんし、理解されていると思います。ただし、実践となると目前のシステムの運用管理に焦点が当てられてしまい、ハイレベル・デザインが実施されていない場合があります。これは2章で指摘したような分散システムの環境変化が、まだこれから発生する状況であったり、当面は運用管理すべき分散システムが数個にとどまり、従来の運用管理の延長線上で対応が可能であったりするからでしょう。

差し当たり運用管理すべき具体的な分散システムに対して、その運用管理を設計する場合でも、今後の分散システムの拡張を念頭に置いた設計は必要です。特にアプリケーションに依存したマルチプラットフォーム環境では、全社的に統一された運用管理業務の標準化は必要不可欠です。

ハイレベル・デザインの実践には、複数のアプローチや方法論が考えられます。運用コンサルテーションに依頼する場合や、既存の運用標準を、分散システムを想定して改訂する場合もあります。最近では、アウトソース・ベンダーから運用管理業務の提案がなされる場合もあります。MSP (Management Service Provider) と称されるシステムの運用管理サービスの提供を目指したサービス提供ベンダーを利用することも選択肢の一つです。

いずれのアプローチでもハイレベル・デザインの実践に対しては、十分な抽象性と同時に十分な具体性を確保する必要があります。その結果、将来のシステムの拡張を視野に入れたハイレベル・デザインを実施できます。さらに今後のアプリケーションの拡張にも対応できる運用管理システムを実装することもできるでしょう。

分散システムの将来的な拡張、質的な変化、量的な変化、さらに運用形態の変化にも対応できるような柔軟な運用管理システムを構築するためにも、ハイレベル・デザインの意義を再認識することが重要です。

6. おわりに

本論文では、分散システムの最近の変化に焦点を当て、それらの変化へのシステムとしての対応を列挙しました。運用管理に関連する課題点の幾つかは、ハイレベル・デザインを実施することにより対応できるものです。ハイレベル・デザインの国内での実績は必ずしも多いとはいえません。しかし、今後の分散システムの質的、かつ量的な拡大や運用形態の変化を想定すると、分散システムごとの個別の運用管理では限界が来ることは明らかです。そうした状況を踏まえ、本論文が、ハイレベル・デザインの位置付けや必要性の再確認と実践へのヒントになり、また、今後の分散システムの運用管理を設計・構築する方々の参考になれば幸いです。

[参考文献]
[1] 仲田 聡 編『システムマネジメント』(株) リックテレコム、1998年
[2] 各種の方法論に関しては、http://www.ibm.com から検索可能。
[3] Tivoli関連の方法論に関しては、http://www.tivoli.com から検索可能

(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)