Sun Java Enterprise System 2005Q4 配備計画ガイド

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054
U.S.A.

Part No: 819–3449
2005 年 10 月

原典: Sun Java Enterprise System 2005Q4 Deployment Planning Guide

Part No: 819-2326

Revision A

Adobe PostScript

051218@13215

目次

表目次

図目次

はじめに

『Sun Java Enterprise System 2005Q4 配備計画ガイド』では、Sun Java™ Enterprise System に基づく企業向け配備ソリューションの計画と設計について説明します。このマニュアルでは、配備計画と設計の基本概念と原則について、また配備設計プロジェクトの各フェーズとタスクで構成されるソリューションライフサイクルについて説明するとともに、Java Enterprise System (Java ES) に基づく企業全体用の配備ソリューションを計画する上で役立つ高レベルの例と戦略を示します。

対象読者

このマニュアルは主に、企業向け配備の分析と設計を担当する配備設計者およびビジネスプランナーを対象としています。このマニュアルは、企業アプリケーションのさまざまな要素の設計および実装を担当するシステムインテグレータなどにとっても有用です。

お読みになる前に

このマニュアルは、読者が企業レベルのアプリケーションの設計とインストールについて深く理解していること、および『Sun Java Enterprise System 2005Q4 Technical Overview』をすでに読んでいることを前提としています。

内容の紹介

このマニュアルは、配備計画の各フェーズを説明するソリューションライフサイクルに基づいています。ソリューションライフサイクルについては、第 1 章で説明しています。

Java ES のマニュアルセット

Java ES のマニュアルセットには、配備計画とシステムのインストールに関する情報が記載されています。システムマニュアルの URL は `http://docs.sun.com/coll/1286.1` です。Java ES の概要については、次の表に示された順序で、各マニュアルを参照してください。

表 **P–1** Java Enterprise System マニュアル

マニュアル	内容
『Sun Java Enterprise System 2005Q4 リリースノート』	既知の問題を含む、Java ES の最新情報を記載しています。これに加えて、各コンポーネントには、個別のリリースノートもあります。
『Sun Java Enterprise System 2005Q4 Documentation Roadmap』	Java ES のシステムおよび各コンポーネントの両方の関連マニュアルすべてについて説明しています。
『Sun Java Enterprise System 2005Q4 Technical Overview』	Java ES の技術基盤および概念基盤の概要を説明しています。コンポーネント、アーキテクチャー、プロセス、および機能について説明しています。
『Sun Java Enterprise System 2005Q4 配備計画ガイド』	Java ES に基づく企業向け配備ソリューションの計画と設計について説明しています。配備計画と設計についての基本概念と原則およびソリューションライフサイクルについて説明するとともに、Java ES に基づくソリューションを計画する上で役立つ高レベルの例と戦略を示しています。
『Sun Java Enterprise System 2005Q4 Installation Planning Guide』	Java ES 配備のハードウェア、オペレーティングシステム、およびネットワークの特徴に合わせて実装仕様を作成するために役立ちます。コンポーネントの依存関係など、インストールと設定の計画で対処すべき問題について説明しています。

表 **P–1** Java Enterprise System マニュアル　(続き)

マニュアル	内容
『Sun Java Enterprise System 2005Q4 インストールガイド(UNIX 版)』	Solaris オペレーティングシステムまたは Linux オペレーティングシステム用に Java ES をインストールする方法を説明しています。インストール後のコンポーネントの設定、および設定したコンポーネントが正しく機能することを検証する方法も示しています。
『Sun Java Enterprise System 2005Q4 Installation Reference』	設定パラメータに関する追加情報、設定の計画に使用するワークシート、およびデフォルトディレクトリやポート番号などの参考資料を記載しています。
『Sun Java Enterprise System 2005Q1 Deployment Example Series: Evaluation Scenario』	1 つのシステムに Java ES をインストールし、中核の共有ネットワークサービスのセットを確立し、確立したサービスにアクセスできるユーザーアカウントを設定する方法について説明しています。
『Sun Java Enterprise System 2005Q4 Upgrade Guide』	Solaris オペレーティングシステムまたは Linux オペレーティング環境で Java ES をアップグレードする手順について説明しています。
『Sun Java Enterprise System Glossary』	Java ES のマニュアルで使用されている用語を定義しています。

表記上の規則

このマニュアルでは、次のような字体や記号を特別な意味を持つものとして使用します。

表 **P–2** 表記上の規則

字体または記号	意味	例
`AaBbCc123`	コマンド名、ファイル名、ディレクトリ名、画面上のコンピュータ出力、コード例を示します。	`.login` ファイルを編集します。 `ls -a` を使用してすべてのファイルを表示します。 `machine_name% you have mail.`
`AaBbCc123`	ユーザーが入力する文字を、画面上のコンピュータ出力と区別して示します。	`machine_name%` **`su`** `Password:`
aabbcc123	変数を示します。実際に使用する特定の名前または値で置き換えます。	ファイルを削除するには、`rm` *filename* と入力します。

表 **P–2** 表記上の規則　　(続き)

字体または記号	意味	例
『 』	参照する書名を示します。	『コードマネージャ・ユーザーズガイド』を参照してください。
「 」	参照する章、節、ボタンやメニュー名、強調する単語を示します。	第 5 章「衝突の回避」を参照してください。 この操作ができるのは、「スーパーユーザー」だけです。
\	枠で囲まれたコード例で、テキストがページ行幅を超える場合に、継続を示します。	`sun% grep '^#define \` `  XV_VERSION_STRING'`

コード例は次のように表示されます。

- C シェル

  `machine_name% command y|n [filename]`

- C シェルのスーパーユーザー

  `machine_name# command y|n [filename]`

- Bourne シェルおよび Korn シェル

  `$ command y|n [filename]`

- Bourne シェルおよび Korn シェルのスーパーユーザー

  `# command y|n [filename]`

[ ] は省略可能な項目を示します。上記の例は、*filename* は省略してもよいことを示しています。

| は区切り文字 (セパレータ) です。この文字で分割されている引数のうち 1 つだけを指定します。

キーボードのキー名は英文で、頭文字を大文字で示します (例: Shift キーを押します)。ただし、キーボードによっては Enter キーが Return キーの動作をします。

ダッシュ (-) は 2 つのキーを同時に押すことを示します。たとえば、Ctrl-D は Control キーを押したまま D キーを押すことを意味します。

コマンド例のシェルプロンプト

次の表は、デフォルトのシステムプロンプトとスーパーユーザープロンプトを示しています。

表 **P–3** シェルプロンプト

シェル	プロンプト
UNIX システムおよび Linux システムの C シェル	`machine_name%`
UNIX システムおよび Linux システムの C シェルのスーパーユーザー	`machine_name#`
UNIX システムおよび Linux システムの Bourne シェルおよび Korn シェル	`$`
UNIX システムおよび Linux システムの Bourne シェルおよび Korn シェルのスーパーユーザー	`#`
Microsoft Windows のコマンド行	`C:\`

記号の表記規則

次の表では、このマニュアルで使用されている記号について説明します。

表 **P–4** 記号の表記規則

記号	説明	例	意味
`[ ]`	省略可能な引数およびコマンドオプションが含まれます。	`ls [-l]`	`-l` オプションは必須ではありません。
`{ | }`	必須のコマンドオプションの選択肢のセットが含まれます。	`-d {y|n}`	`-d` オプションには、`y` 引数または `n` 引数のどちらかを使用する必要があります。
`${ }`	変数の参照を示します。	`${com.sun.javaRoot}`	`com.sun.javaRoot` 変数の値を参照します。
-	同時に実行する複数のキーストロークを結び付けます。	Control-A	コントロールキーを押しながら A キーを押します。

表 **P–4** 記号の表記規則　*(続き)*

記号	説明	例	意味
+	連続する複数のキーストロークを結び付けます。	Ctrl+A+N	コントロールキーを押して放したあと、以降のキーを押します。
→	グラフィカルユーザーインタフェースのメニュー項目の選択を示します。	「ファイル」→「新規」→「テンプレート」	「ファイル」メニューから「新規」を選択します。「新規」サブメニューから「テンプレート」を選択します。

マニュアル、サポート、およびトレーニング

Sun のサービス	**URL**	内容
マニュアル	`http://jp.sun.com/documentation/`	PDF 文書および HTML 文書をダウンロードできます。
サポートおよびトレーニング	`http://jp.sun.com/supportraining/`	技術サポート、パッチのダウンロード、および Sun のトレーニングコース情報を提供します。

第 1 章

配備計画について

この章では、Sun Java™ Enterprise System (Java ES) の概要、配備計画の概念、およびソリューションライフサイクルについて説明します。 ソリューションライフサイクルの説明では、企業のソフトウェアシステムの計画および設計を行うためのさまざまなステップの概要を示します。この章で説明する内容は、次のとおりです。

Java Enterprise System について

Java Enterprise System は、ネットワークまたはインターネット環境に分散された、企業のアプリケーションをサポートするためのミドルウェアサービスをすべて提供するソフトウェアインフラストラクチャーです。サービスを提供する Java Enterprise System のコンポーネントは、共通のインストーラを使用してインストールされ、共通の共有ライブラリセットに同期化され、ユーザーアイデンティティーとセキュリティー管理の統合システムを共有します。

システムサービス

Java Enterprise System のコンポーネントによって提供される主なインフラストラクチャーサービスは、次のように分類されます。

- ポータルサービス: これらのサービスは、モバイル従業員、在宅勤務者、ナレッジワーカー、ビジネスパートナー、取引先、および顧客が企業ネットワーク外のどこからでもインターネットを経由して、個人用カスタマイズされた企業ポータルに安全にアクセスすることを可能にします。これらのサービスは、いつでも、どこからでもユーザーのコミュニティーにアクセスすることを可能にするとともに、統合、集約、個人用カスタマイズ、セキュリティー、モバイルアクセス、および検索を実現します。

- 通信および共同作業サービス**:** これらのサービスは、多様なユーザーコミュニティー間で情報を安全に交換することを可能にします。具体的な機能としては、ユーザーのビジネス環境に即したメッセージング、リアルタイム共同作業、カレンダスケジューリングなどがあります。
- ネットワークアイデンティティーおよびセキュリティーサービス**:** これらのサービスは、適切なアクセス制御ポリシーをすべてのコミュニティー、アプリケーション、およびサービスに対してグローバルベースで実施することで、企業の重要な情報資産のセキュリティーと保護を強化します。これらのサービスは、アイデンティティープロファイル、アクセス権、およびアプリケーションとネットワークのリソース情報の格納と管理に使用されるリポジトリと連動します。
- **Web** およびアプリケーションサービス**:** これらのサービスは、分散されたコンポーネントが相互に通信することを可能にするとともに、さまざまなサーバー、クライアント、およびデバイス用のアプリケーションの開発、配備、および管理をサポートします。これらのサービスは、Java 2 Platform, Enterprise Edition (J2EE™) テクノロジに基づいています。
- 可用性サービス**:** これらのサービスは、アプリケーションと Web サービスのほぼ連続的な可用性とスケーラビリティーを提供します。

次の表は、上記のサービスカテゴリと、各カテゴリのサービスを提供する Java Enterprise System コンポーネントを示しています。

表 **1–1** Java Enterprise System のサービスカテゴリ

サービスカテゴリ	**Java Enterprise System** コンポーネント
ポータルサービス	Portal Server
	Portal Server Secure Remote Access
	Access Manager
	Directory Server
	Application Server
	Web Server
通信および共同作業サービス	Messaging Server
	Calendar Server
	Instant Messaging
	Access Manager
	Directory Server
	Application Server
	Web Server

表 1–1 Java Enterprise System のサービスカテゴリ　(続き)

サービスカテゴリ	**Java Enterprise System** コンポーネント
ネットワークアイデンティティーサービス	Access Manager
	Directory Server
	Web Server
Web およびアプリケーションサービス	Application Server
	Message Queue
	Web Server
可用性サービス	Sun Cluster
	Sun Cluster エージェント

Java Enterprise System のサービス、コンポーネント、および Java Enterprise System のアーキテクチャー概念の詳細については、『Sun Java Enterprise System 2005Q4 Technical Overview』を参照してください。

組み込みサービスとカスタム開発されたサービス

Java Enterprise System に基づく配備ソリューションは一般に、次の 2 つのカテゴリに分類されます。

- **80:20** 配備**:** これらのソリューションは、主に Java Enterprise System によって提供されるサービスで構成されます。サービスの 80% 以上が Java Enterprise System によって提供されます。
- **20:80** 配備**:** これらのソリューションは、多数のカスタム開発されたサービスおよびサードパーティー製アプリケーションで構成されます。

80:20 および 20:80 の分類は、大まかな一般化です。提供されるサービスのタイプの正確な割合は重要ではありません。ただし、割合は、ソリューションに含まれるカスタマイズの量を示します。

Java Enterprise System は、特に 80:20 配備に適しています。Java ES によって豊富なサービスセットが提供されるためです。たとえば、Java Enterprise System によって提供されるサービスを使用すると、企業全体用の通信システムや企業全体用のポータルシステムの配備は比較的容易に行えます。

カスタム開発を必要とする配備については、Java Enterprise System は、カスタム開発によるサービスとアプリケーションの作成と統合をサポートします。

17 ページの「システムサービス」に示されているサービスカテゴリのほとんどは、80:20 配備を提供するために使用できます。たとえば、通信および共同作業サービスは、電子メール、カレンダ、およびインスタントメッセージングの各サービスをエンドユーザーに提供し、エンドユーザーがコンテンツを集約および個人用カスタマイズすることを可能にします。同様に、ネットワークアイデンティティーサービスとエンタープライズポータルのサービスカテゴリは、カスタムサービスの開発や統合なしに、企業全体用のアプリケーションのインストールと設定を行うことを可能にします。

J2EE プラットフォームサービスのカスタム開発が必要な企業ソリューションは、Application Server、Message Queue、または Web Server を活用できます。これらには、Java Enterprise System の Web およびアプリケーションサービスが提供されます。

企業の配備では、必要とされるカスタム開発されたサービスの数は大きく異なります。Java Enterprise System の各種サービスは相互運用性が確保されているので、企業の特定のニーズに合わせてサービスを独自に組み合わせることができます。

Java Enterprise System への移行

Java Enterprise System を使用する企業ソリューションの計画、設計、および実装は、主に現行の配備戦略によって決まります。初めての配備ソリューションを計画している企業の場合、計画、設計、および実装は、主に企業の特定のニーズによって決まります。ただし、初めての配備ソリューションは一般的ではありません。Java Enterprise System を使用して、企業の既存のソリューションを強化したり、古いバージョンの Java Enterprise System コンポーネントをアップグレードしたりするソリューションのほうが一般的です。

既存のソリューションを交換またはアップグレードする場合、計画、設計、および実装において追加の手順を実行し、既存のデータが保持されることおよびソフトウェアが最新のバージョンに適切にアップグレードされることを確認する必要があります。このマニュアルで概要が説明されている分析および設計を行う際は、既存のソフトウェアシステムの交換およびアップグレードに必要な準備と計画に留意してください。

最新のバージョンの Java Enterprise System にアップグレードする方法およびほかのアプリケーションから移行する戦略の詳細については、『Java Enterprise System アップグレードと移行』を参照してください。

配備計画の概要

配備計画は、Java Enterprise System ソリューションの実装を成功させるにあたって不可欠なステップです。企業はそれぞれ独自の目標、要件、および優先順位を考慮する必要があります。計画を成功させるには、企業の目標を分析し、それらの目標を達成するためのビジネス要件を特定することから始めます。その後、ビジネス要件を、企業の目標を達成するシステムの設計と実装のベースとして使用される技術要件に変換する必要があります。

配備計画の成功は、慎重な準備、分析、および設計によってもたらされます。計画プロセス中に誤りや失敗があると、いろいろな形でシステムが失敗する恐れがあります。重大な問題は、システムの計画が不完全であることから生じる可能性があります。たとえば、パフォーマンスが予定を下回る、保守が難しい、運用コストが高すぎる、リソースを浪費する、増大するニーズに対応できないといった問題がシステムに生じる恐れがあります。

ソリューションライフサイクル

次の図に示すソリューションライフサイクルは、Java Enterprise System をベースにした企業のソフトウェアソリューションの計画、設計、および実装におけるステップを説明しています。ライフサイクルは、配備プロジェクトを進める上で有用なツールです。

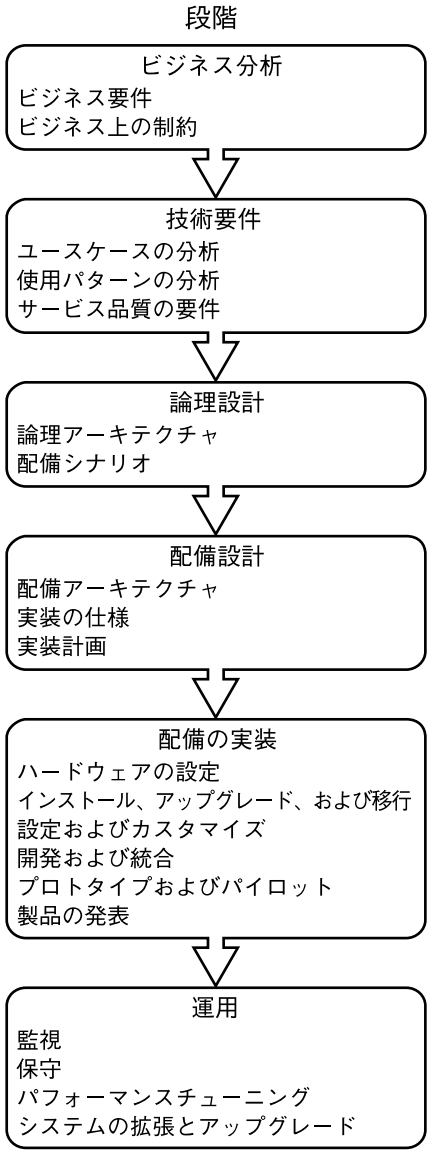


図 **1–1** ソリューションライフサイクル

ライフサイクルは、順序性のあるフェーズで構成されています。各フェーズは、後続フェーズへのインプットとして渡されるアウトプットを生成する関連性のあるタスクで構成されています。各フェーズ内のタスクは、そのフェーズのアウトプットを生成する前に徹底的な分析と設計を必要とするもので、繰り返し実行されます。初期のいくつかのフェーズも、繰り返し実行されることがあります。たとえば、配備設計フェーズでは、それまでのフェーズでの分析が不十分で、さらに分析する必要があることに気が付く場合があります。

以降の各節では、各ライフサイクルフェーズの概要を説明しています。

ビジネス分析フェーズ

ビジネス分析フェーズでは、配備プロジェクトのビジネス上の目的を定義し、その目的を果たすために必要となるビジネス要件を明確にします。ビジネス要件を明確化するときは、ビジネス上の目的を果たす上で障害となりうるビジネス上の制約を考慮します。ソリューションライフサイクル全体において配備計画の成功度を評価し、最終的には、ビジネス分析フェーズで行なった分析に基づいて、配備ソリューションの成功度を評価します。

ビジネス分析フェーズでは、ビジネス要件ドキュメントを作成します。 このドキュメントは、あとで技術要件フェーズへのインプットとして使用します。

ビジネス分析フェーズの詳細については、第 2 章を参照してください。

技術要件フェーズ

技術要件フェーズでは、ビジネス分析フェーズで定義したビジネス要件とビジネス制約に基づいて、これらの要件を技術仕様に変換します。これらの仕様は、あとで行う配備アーキテクチャーの設計に使用されます。技術要件には、パフォーマンス、可用性、セキュリティーなど、サービス品質 (QoS: Quality of Service) の特徴を特定します。

技術要件フェーズでは、次の情報についてドキュメントを作成します。

- ユーザータスクと使用パターンの分析
- 計画中のシステムに対するユーザー操作のモデルとなるユースケースの適用
- 場合によってはユーザータスクと使用パターンの分析に基づく、ビジネス要件から派生するサービス品質要件

使用パターンの分析結果、ユースケース、および QoS 要件ドキュメントは、ソリューションライフサイクルの論理設計フェーズへのインプットとなります。使用パターン分析は、配備設計フェーズでも重要な役割を果たします。

技術要件フェーズでは、サービスレベル要件も特定する場合があります。 その要件は、あとでサービスレベル契約 (SLA: Service Level Agreement) を作成する際のベースになります。サービスレベル契約には、システムを維持するために提供されるカスタマーサポートの条件を明示します。 これらは、配備設計フェーズにおけるプロジェクト承認の一環として署名されます。

技術要件の詳細については、第 3 章を参照してください。

論理設計フェーズ

論理設計では、技術要件フェーズで得られたユースケースをインプットとして使用して、ソリューションの実装に必要な Java Enterprise System コンポーネントを特定します。それらの Java ES コンポーネントをサポートするコンポートを特定するとともに、ビジネス要件を満たすために必要な、追加のカスタム開発されたコンポーネントをすべて特定します。その後、コンポーネント間の相互関係を示す論理アーキテクチャー内のコンポーネントをマッピングします。論理アーキテクチャーには、ソリューションの実装に必要なハードウェアは特定しません。

論理設計フェーズのアウトプットは、論理アーキテクチャーです。論理アーキテクチャーだけでは、配備設計を開始するには不十分です。技術要件フェーズからの QoS 要件も必要です。技術要件フェーズからの論理アーキテクチャーと QoS 要件により、配備シナリオが形成されます。この配備シナリオが、配備設計フェーズへのインプットとなります。

論理設計の詳細については、第 4 章を参照してください。

配備設計フェーズ

配備設計では、論理アーキテクチャーに特定されているコンポーネントを物理環境にマッピングし、上位レベルの配備アーキテクチャーを作成します。また、実装仕様も作成します。 これには、配備アーキテクチャーの構築方法を指定する、下位レベルの詳細を示します。さらに、ソフトウェアソリューションの実装におけるさまざまな側面を詳細に説明する一連の計画と仕様も作成します。

プロジェクト承認は、配備設計フェーズで行われます。プロジェクト承認時は、配備のコストが見積もられます。承認された場合、配備の実装に関する契約に署名がされ、プロジェクトを構築するためのリソースが確保されます。多くの場合、プロジェクトの承認は、実装仕様が詳細化されたあとで行われます。ただし、配備アーキテクチャーの完成後に承認が行われる場合もあります。

配備設計フェーズのアウトプットには、次のものがあります。

- 配備アーキテクチャー**:** ネットワークハードウェアおよびソフトウェアに対するコンポーネントのマッピングを表す、上位レベルの設計ドキュメントです。
- 実装仕様**:** 配備を構築するための青写真として使用される詳細な仕様です。
- 実装計画**:** 企業ソフトウェアソリューションの実装におけるさまざまな側面をカバーする、一連の計画と仕様です。実装計画には、移行計画、インストール計画、ユーザー管理計画、テスト計画などが含まれます。

配備設計の詳細については、第 5 章を参照してください。

実装フェーズ

実装フェーズでは、配備設計で作成された仕様と計画に基づいて、配備アーキテクチャーを構築し、ソリューションを実装します。配備プロジェクトの性質に応じて、このフェーズには次のタスクの一部またはすべてが含まれます。

- ハードウェアインフラストラクチャーのインストールと設定をする
- ソフトウェアのインストールと設定をする
- LDAP ディレクトリ設計でユーザーおよびリソースをモデル化する
- ユーザー管理計画に基づいて、既存のディレクトリおよびデータベースからデータを移行する
- パイロット配備およびプロトタイプ配備を作成して、テスト環境に配備する
- システム要件を満たすことを確認するために、機能テストを設計、実行する
- ピーク負荷時のパフォーマンスを確認するために、ストレステストを設計、実行する
- 企業のカスタムアプリケーションを開発および統合する
- 本稼働配備を作成する (段階的に行われる場合がある)

  配備が本稼働に入ったら、ソリューションライフサイクルの運用フェーズに進みます。

実装フェーズの詳細については、第 6 章を参照してください。

運用フェーズ

運用フェーズには、配備の実装を円滑に運用するために必要なタスクが含まれます。このフェーズには、次のタスクがあります。

- 配備を監視して、システムが計画どおりに稼働していることを確認する
- パフォーマンスを調整して、配備されたソフトウェアが最適なレベルで実行されることを確認する
- 円滑な運用のための定期的保守、および必要に応じた臨時保守を提供する
- 必要に応じてソフトウェアおよびハードウェアをアップグレードする

運用フェーズの詳細については、このマニュアルの対象範囲に含まれません。

第 2 章

ビジネス分析

ソリューションライフサイクルのビジネス分析フェーズでは、ビジネス上の問題点を分析することでビジネス上の目的を定義し、その目的を達成するにあたってのビジネス要件とビジネス制約を明確にします。

この章で説明する内容は、次のとおりです。

ビジネス分析について

ビジネス分析は、ビジネス上の目的を明確にすることから始めます。次に、解決が必要なビジネス上の問題点を分析するとともに、ビジネス上の目的の達成に必要なビジネス要件を特定します。また、目的を果たす上で障害となりうるビジネス上の制約についても考慮します。ビジネスの要件と制約を分析した結果から、一連のビジネス要件ドキュメントを作成します。

作成した一連のビジネス要件ドキュメントは、技術要件フェーズで技術要件を導き出すためのベースとして使用します。ソリューションライフサイクル全体において配備計画の成功度を評価し、最終的には、ビジネス分析フェーズで行なった分析に基づいて、ソリューションの成功度を評価します。

ビジネス要件の定義

すべてのビジネス要件を特定できる単純な公式は存在しません。ソフトウェアソリューションを必要とする利害関係者との共同作業、ビジネス分野に関する各自の知識、およびクリエイティブな思考からビジネス要件を決定します。

ここでは、ビジネス要件を定義する際に考慮するべき要因について説明します。

ビジネス上の目的の設定

ビジネス分析では、配備プロジェクトの目的を明確にする必要があります。目的を明確にすることで、設計上の意思決定に集中しやすくなり、プロジェクトが道を外れることを回避できます。ビジネス上の目的と現状を比較することで、設計上の意思決定も行いやすくなります。

範囲

ビジネス要件には、配備プロジェクトの範囲を定義する必要があります。解決する領域を特定するとともに、目的が不明瞭になったり、達成不能になったりしないように、オープンエンドな要件は避ける必要があります。定義した範囲が不明瞭であると、ビジネスニーズを十分に満たせない配備設計やリソースが浪費される配備設計につながる恐れがあります。

優先順位

最も重要な配備内容が最初に達成されるように、目的に優先順位を付けます。リソースが限定されている場合、いくつかの目標を後回しにしたり、修正したりする必要も生じます。たとえば、大規模で複雑な配備では一般に、ソリューションの段階的な実装が必要とされます。優先順位を明確にすることで、利害関係者に受け入れられる配備設計を作成する上で必要な、意思決定のガイドラインが得られます。

不可欠な品質

成功に不可欠な領域を特定し、利害関係者および設計者が最も重要な基準に集中できるようにします。

成長要因

ビジネス上の目的を設定する際は、組織の現在のニーズを考慮するだけでなく、そのニーズの長期にわたる変化および増大を予測します。途中で役に立たなくなるソリューションとならないようにするためです。

安全マージン

ソリューションの設計は、このビジネス分析フェーズで確立された前提に基づきます。不十分なデータ、判断上の誤り、予期しない外部のイベントなど、さまざまな理由から、これらの前提は正確でない場合があります。設計するソリューションが予期しないイベントに対処できるように、ビジネス上の目的だけでなく、計画全体において、安全マージンを考慮する必要があります。

ユーザーニーズの把握

ソリューションの対象ユーザーのタイプ、ユーザーのニーズ、および予期されるユーザー利益を把握するために必要なリサーチを行います。次のリストは、ユーザーを分類する方法の一例です。

- 現在の従業員のみ
- 現在と過去の従業員
- 管理者
- 現在の顧客
- すべての顧客
- メンバーシップサイト
- 一般大衆
- 制限付きアクセス

予期されるユーザー利益の明確化は、設計上の意思決定に役立ちます。次に示すのは、ソリューションがユーザーにもたらす利益の例です。

- 企業リソースへのリモートアクセス
- 組織内での共同作業
- 日次業務の単純化
- 遠隔地のチームとのリソースの共有
- 生産の向上
- エンドユーザー自身による管理

動作要件の定義

動作要件を、明確な目標を持つ一連の機能要件として表します。一般的には、次のような領域について動作仕様を作成します。

- エンドユーザー用の機能
- 応答時間の短縮
- 可用性と稼働時間
- エラーレートの低減
- 情報のアーカイブと保持

動作要件を、すべての利害関係者が理解できる測定可能な表現で表します。「適切なエンドユーザー応答時間」などのあいまいな表現を避けます。次に示すのは、動作要件の例です。

- 障害発生後 10 分以内にサービスを再開できる
- 48 時間以内に受信したメッセージを再受信できる
- ピーク時のオンライントランザクションが 60 秒以内に完了する
- ピーク時のエンドユーザー認証が 4 秒以内に完了する

既存の使用パターンに対するサポート

既存の使用パターンを測定可能な目標として明確に表します。次に示すのは、そのような目標を決定する上で役立つ質問です。

- 現在のサービスはどのように利用されているか。
- どのような使用パターンがあるか (たとえば、散発的、頻繁、または非常に頻繁な使用)。
- ユーザーが一般にアクセスするサイトはどれか。
- ユーザーが一般に送信するメッセージのサイズはどのくらいか。
- ユーザーが一般に 1 日または 1 時間に完了するトランザクションはいくつか。

サービスにアクセスするユーザーについて調査します。ユーザーが既存のサービスにアクセスする時刻やその長さなどの要因は、目標を特定する上で重要です。組織の経験からこれらのパターンが得られない場合は、類似組織の経験を調査します。

企業文化の理解

要件分析では、企業の文化および政策のさまざまな側面を考慮する必要があります。企業文化に対する考慮が欠けていると、受け入れられないソリューションや実装しにくいソリューションとなる恐れがあります。

利害関係者

提案したソリューションの成功によって恩恵を受ける個人および組織を特定します。ビジネス上の目的と要件の定義には、すべての利害関係者が活発に参加する必要があります。利害関係者が参加しない場合、または変更計画を知らされない場合、計画が重大な欠点を持つ可能性があります。そのような利害関係者は、配備の実装を妨げる可能性さえもあります。

標準と方針

ソリューションを必要としている組織の標準と方針を明確に理解する必要があります。これらの標準と方針が、設計の技術面、製品の選択、配備方法に影響する場合があります。

一例として、人事組織や部門責任者によって所有および管理される個人データの機密性が挙げられます。別の例としては、企業による変更管理の手順が挙げられます。変更管理方針は、ソリューションの受け入れに大きく影響するとともに、実装方法やスケジュールに大きく影響する可能性があります。

規制による要件

規制による要件は、ビジネスの性質によって大きく異なります。配備に影響する可能性のある規制による要件をすべて調査し、理解します。多くの企業および政府機関は、アクセシビリティー標準への準拠を必要とします。グローバルなソリューションを配備する際は、外国の法規制を考慮します。たとえば、多くのヨーロッパ諸国は、個人情報の保管について厳しく規制しています。

セキュリティー

特定した目標の中には、重視する必要のある、潜在的なセキュリティー上の問題がある場合があります。ソリューションに不可欠な、具体的なセキュリティー上の目標を明確にします。例:

- 占有情報へのアクセスは、認証されたユーザーに限定する
- 機密情報へのアクセスは、ロールベースのアクセスに限定する
- 遠隔地との通信をセキュリティー保護する
- ローカルシステム上でリモートアプリケーションを利用できるようにする
- サードパーティーの企業および組織とのトランザクションをセキュリティー保護する
- セキュリティーポリシーを実施する

サイトの分散

サイトの地理的分散およびサイト間の帯域幅は、設計上の意思決定に影響する可能性があります。また、ローカル管理が必要なサイトがある場合もあります。

このような地理的な考慮事項により、プロジェクトの研修費や複雑さなどが増大する可能性があります。サイトの地理的分散による要件を明確にします。設計の成功に不可欠なサイトを特定します。

段階的な配備の採用

通常、ソフトウェアソリューションは、全体が包括的なシステムとして見なされます。しかし、多くの場合は、慎重なステップを踏むことで、完全なシステムの配備に段階的に到達します。

段階的な配備を採用する場合、通常は、最終的に包括的なソリューションに至るまでの段階的な目標を示すロードマップを作成します。また、包括的なソリューションのうち実装が延期された部分に関する短期の計画を立てる必要がある場合もあります。

段階的な配備には、次のような利点があります。

- ビジネスの成長による要件の変化に適応できる。
- 最終的な配備の実装までの過渡期に既存のインフラストラクチャーを活用できる。

- 資本支出要件に対応できる。
- 少人数の人的資源を活用できる。
- 提携の可能性を考慮することができる。

サービスレベル契約について

サービスレベル契約 (SLA: Service Level Agreement) には、最小パフォーマンス要件と障害時にこれらの要件を満たすために提供する必要のあるカスタマーサポートのレベルと範囲を規定します。サービスレベル契約は、ビジネス分析で定義されたビジネス要件に基づきます。 これはあとで、技術要件フェーズにおいてサービスレベル要件として示されます。SLA は、配備設計フェーズのプロジェクト承認時に署名がなされます。

稼働時間、応答時間、メッセージ配信時間、障害回復などの領域について SLA を作成する必要があります。SLA では、システムの概要、サポート組織の役割と責任、サービスレベルの評価方法、変更要求などの項目を説明する必要があります。システムの可用性に関する自組織の期待を特定することは、SLA の範囲を決定する上で重要です。

ビジネス上の制約の定義

ビジネス上の制約は、配備プロジェクトの本質に関わる重要な問題です。成功する配備設計の鍵の 1 つは、ビジネス上の制約の範囲内でビジネス要件を満たす最適な方法を見つけることにあります。ビジネス上の制約としては、財務的制約、物理的制約 (たとえば、ネットワーク容量)、時間的制約 (たとえば、次回の年次会議などの重要なイベントの前に完了する) など、ビジネス上の目的の達成に影響するあらゆる要因が考えられます。

ここでは、ビジネス上の制約を定義する際に考慮するべき要因について説明します。

移行問題

通常、既存のソフトウェアインフラストラクチャーおよびデータは、配備プロジェクトで交換または補完されます。新しいソリューションは、既存のインフラストラクチャーから新しいソリューションにデータおよび手順を移行できる必要があり、多くの場合は、既存のアプリケーションとの相互運用性も保持できる必要があります。現行のインフラストラクチャーの分析は、提案ソリューションの移行問題を特定するために必要です。

スケジュール制約

ソリューションの実装スケジュールは、設計上の意思決定に影響する可能性があります。厳しいスケジュールは、目標の規模縮小、優先順位の変更、段階的なソリューションの採用などにつながることがあります。スケジュール内にも、注意を必要とする重要な達成期限が定められることがあります。達成期限は、スケジュールされたサービスのロールアウトなどの内部イベント、または学期の開始日などの外部イベントによって設定される場合があります。

予算制約

ほとんどの配備プロジェクトは、予算内で達成する必要があります。次の点を含めて、提案ソリューションを構築する費用とソリューションを特定期間維持する上で必要なリソースを検討します。

- 既存のハードウェアとネットワークインフラストラクチャー**:** 既存インフラストラクチャーの活用は、システム設計に影響する可能性があります。
- ソリューションの実装に必要な開発リソース**:** ハードウェア、ソフトウェア、人的資源などの開発リソースが限られている場合、段階的な配備が必要となる可能性も生じます。実装の各フェーズで、同じリソースや開発チームを再利用しなければならない場合があります。
- 保守、管理、およびサポート**:** システムユーザーの管理、保守、およびサポートに利用できるリソースを分析します。リソースが限られていると、設計上の意思決定に影響する場合があります。

保有コスト

保守、管理、サポートのほかに、保有コストに影響するその他の要因を分析します。必要となるハードウェアとソフトウェアのアップグレード、電力網に対するソリューションの影響、通信費、および出費を要するその他の要因があります。ソリューションの可用性レベルを規定するサービスレベル契約も、より多くの冗長性を必要とすることで、保有コストに影響します。

ソリューションの実装は、ソリューションに対する投資の回収を実現するものである必要があります。投資回収率の分析では、通常は、投資費用から得られた経済的効果が算出されます。

ソリューションの経済的効果を見積もるときは、ビジネス上の目標を達成した場合と、それ以外の方法で同じ目標を達成した場合、または、なにも行わなかった場合を比較して慎重に分析する必要があります。

第 3 章

技術要件

ソリューションライフサイクルの技術要件フェーズでは、使用パターンの分析を行い、ユースケースを特定し、提案する配備ソリューションのサービス品質要件を決定します。

この章で説明する内容は、次のとおりです。

技術要件について

技術要件の分析は、ソリューションライフサイクルのビジネス分析フェーズで作成されたビジネス要件ドキュメントに基づいて行います。ビジネス分析に基づいて、次の手順を実行します。

- 想定される負荷状況を特定するために、使用パターンを分析します。
- システムに対するユーザー操作のモデルとなるユースケースを作成します。
- 配備されたソリューションに必要とされる、応答時間、可用性、セキュリティーなどの領域におけるパフォーマンスを定義する一連のサービス品質 (QoS) 要件を作成します。

  サービス品質要件は、それまでに特定したビジネス要件とビジネス制約を考慮しながら、使用パターン分析およびユースケースから導き出します。

  後の論理設計フェーズでは、サービス品質要件を論理アーキテクチャーと対応付けて、配備シナリオを作成します。配備シナリオは、ソリューションライフサイクルの配備設計フェーズへの主要なインプットです。

ビジネス分析の場合と同様に、使用パターン分析、ユースケース、およびシステム要件を導く、技術要件分析のための単純な公式は存在しません。技術要件分析では、ビジネス領域、ビジネス上の目的、基本となるシステム技術の理解が必要となります。

使用パターンの分析

使用パターンの分析では、設計しているソリューションを利用するさまざまなユーザーを特定し、それらのユーザーの使用パターンを特定します。収集した情報は、システムの負荷状況を見積もるための基礎となります。使用パターンの分析結果は、38 ページの「ユースケース」で説明するユースケースに重みを割り当てる上でも役立ちます。

使用パターンを分析するときは、できるだけ多く機会を設けてユーザーにインタビューし、使用パターンに関する既存のデータを収集する必要があります。また、従来のシステムの構築者や管理者にもインタビューします。次の表は、使用パターンの分析時に考慮すべき要因を示しています。

表 **3–1** 使用パターンの分析で考慮すべき要因

項目	説明
ユーザーの人数と種類	ソリューションがサポートするユーザーの人数を特定し、必要に応じてユーザーを分類します。 例: ■ B2C (企業対顧客) ソリューションでは、訪問者は多いが、登録し、ビジネストランザクションに参加する人数は少ない可能性があります。 ■ B2E (企業対従業員) ソリューションでは、通常は各従業員に対応します。 ただし、一部の従業員は社内ネットワークの外からのアクセスを必要とします。B2E ソリューションでは、管理職の従業員は一般従業員がアクセスできない領域へのアクセス権を必要とする場合があります。
アクティブユーザーと非アクティブユーザー	アクティブユーザーと非アクティブユーザーの使用パターンと使用率を特定します。 アクティブユーザーは、システムにログインし、システムサービスと対話しているユーザーです。非アクティブユーザーは、ログインしていないユーザー、ログインはしていてもシステムコンポーネントと対話していないユーザー、データベースには存在するものの一度もログインしていないユーザーなどです。

表 3–1 使用パターンの分析で考慮すべき要因　(続き)

項目	説明
管理ユーザー	配備を監視、更新、サポートするために、配備されたシステムにアクセスするユーザーを特定します。 技術要件に影響する可能性のある管理上の使用パターンをすべて特定します。 たとえば、ファイアウォール外からの配備の管理などです。
使用パターン	各種ユーザーがシステムにどのようにアクセスするかを特定し、想定される用途の対象を特定します。 例: ■ 使用状況が急上昇するピーク時刻は存在するか。 ■ 通常の業務時間帯はいつか。 ■ ユーザーはグローバル規模で分散しているか。 ■ ユーザーの想定接続時間はどれくらいか。
ユーザー数の成長	ユーザーベースのサイズは固定されているか、ユーザー数の増加を想定した配備が必要となるかを調べます。 ユーザーベースの成長が予想される場合は、妥当性のある成長の見込みを立てます。
ユーザートランザクション	サポートする必要のあるユーザートランザクションの種類を特定します。ユーザートランザクションは、ユースケースに変換することができます。 例: ■ ユーザーはどのようなタスクを実行するか。 ■ ログインしたユーザーのログイン状態は維持されるか。一般的には、いくつかのタスクを実行したらログアウトするのか。 ■ ユーザー間の重要な共同作業において、共通のカレンダ、Web 会議室、内部 Web ページの配備を必要とするか。
ユーザーに関する調査と統計データ	既存のユーザー調査などの情報に基づいて、ユーザーの行動パターンを調べます。 多くの企業や組織にはリサーチ結果が用意されており、ユーザーに関する有用な情報をそこから抽出できます。既存のアプリケーションのログファイルにも、システムの概数の算出に使用できる統計データが含まれている可能性があります。

ユースケース

ユースケースは、設計しているソリューションとユーザーとの一般的なインタラクションをモデル化し、エンドユーザーの観点から完全な操作フローを説明します。すべてのユースケースについて設計の優先順位を付けることで、想定される機能の提供に継続して注力できます。ユースケースは、論理設計への主要なインプットです。

ユースケースには重みを割り当てます。 最も重いユースケースが、ユーザーの最も一般的なタスクを表します。ユースケースに重みを割り当てることで、最も使用されるシステムサービスに集中して設計上の意思決定を行うことができます。

ユースケースは、2 つのレベルで説明されます。

- ユースケースレポート**:** 各ユースケースの説明。 イベントの一次フローと代替フローの説明が含まれます。
- ユースケースダイアグラム**:** 関連要素とユースケースの関係を示す図。 より正式な構造でイベントフローが示されます。ユースケースダイアグラムは、長いユースケースまたは複雑なユースケースをモデル化する上で有用です。通常、ユースケースダイアグラムを作成するには、統一モデル言語 (UML: Unified Modeling Language) を使用します。

サービス品質要件

サービス品質 (QoS) 要件は、パフォーマンス、可用性、スケーラビリティーなどについてシステム品質を特定する技術仕様です。QoS 要件は、ビジネス要件で指定されたビジネスニーズによって導き出されます。たとえば、サービスが 24 時間 365 日利用可能であることが必要とされる場合、可用性要件でこのビジネス要件に対応する必要があります。

次の表は、一般的に QoS 要件の基礎となるシステム品質を示しています。

表 3–2 QoS 要件に影響するシステム品質

システム品質	説明
パフォーマンス	ユーザー負荷状況に対応する応答時間とスループット。
可用性	エンドユーザーがシステムのリソースとサービスにアクセスできる頻度。 通常は、システムの稼働時間として表されます。

表 **3–2** QoS 要件に影響するシステム品質　(続き)

システム品質	説明
スケーラビリティー	時間の経過とともに、配備されるシステムに容量 (およびユーザー) を追加する能力。通常、スケーラビリティーにはシステムへのリソースの追加が関連しますが、配備アーキテクチャーの変更は含まれません。
セキュリティー	システムとユーザーの整合性を説明する、要因の複雑な組み合わせ。セキュリティーには、ユーザーの認証と承認、データの安全性、配備されたシステムへのセキュリティー保護されたアクセスなどが含まれます。
潜在処理能力	例外的なピーク負荷が生じた場合に、追加リソースなしでシステムがそれを処理する能力。潜在処理能力は、可用性、パフォーマンス、およびスケーラビリティーの要因です。
保守性	システムの監視、発生した問題の解決、ハードウェアおよびソフトウェアコンポーネントのアップグレードなどを含めた、配備されたシステムの保守の容易さ。

各システム品質は、密接に関連しています。あるシステム品質の必要性が、その他のシステム品質の必要性と設計に影響することがあります。たとえば、セキュリティーのレベルを高めることは、パフォーマンスに影響し、それが可用性に影響する可能性があります。可用性の問題を解決するためにサーバーを追加導入することは、保守性 (保守コスト) に影響します。

ビジネス要件とビジネス上の制約の両方を適切に満たすシステムを設計する鍵は、システム品質がどのように連携するかを把握し、得失評価が必要であることを理解することにあります。

以降の各節では、配備設計に影響するシステム品質を詳細に説明し、QoS 要件を具体化する上で考慮すべき要因についてガイドラインを示します。サービスレベル要件についての節もあります。 この要件は、サービスレベル契約の基礎となります。

パフォーマンス

通常、ビジネス要件では、技術用語を用いずに応答時間を指定することでパフォーマンスを表します。たとえば、Web ベースのアクセスに関するビジネス要件は、次のように指定されます。

ユーザーは、ログイン時に妥当な応答時間、通常は 4 秒以内を想定している。

このビジネス要件に基づき、すべてのユースケースを調べ、この要件をシステムレベルで表す方法を決定します。場合によっては、使用パターン分析で特定したユーザーの負荷状況を含める必要があります。各ユースケースのパフォーマンス要件は、特定の負荷状況における応答時間、または応答時間とスループットの併記で表します。また、許容可能なエラー数も指定します。

次に、パフォーマンス上のシステム要件を指定する例を 2 つ示します。

- Web ページ更新に対する応答時間は、終日 4 秒以内である必要がある。 サンプリングは 15 分間隔で実行されるものとし、エラーの数は 100 万トランザクションあたり 3.4 件未満とする。
- 定義されたピーク時において、システムは 1 秒あたり 25 件のセキュリティー保護されたログインが可能である必要がある。 すべてのユーザーに対する応答時間は 12 秒以内とし、エラーの数は 100 万トランザクションあたり 3.4 件未満とする。

パフォーマンス要件は、可用性要件 (フェイルオーバーがパフォーマンスにどの程度影響するか) および潜在処理能力 (例外的なピーク負荷の処理に使用できる能力がどの程度あるか) に密接に関連します。

可用性

可用性は、システムの稼働時間を特定する基準で、通常はユーザーがシステムにアクセス可能な時間の割合で測定されます。システムにアクセスできなくなる時間 (停止時間) は、ハードウェア、ソフトウェア、ネットワークの障害、またはシステムをダウンさせるその他の要因 (停電など) によって生じます。サービスの予定された停止時間 (保守およびアップグレード) は、停止時間とみなしません。稼働時間の割合でシステムの可用性を計算する基本的な方程式は、次のとおりです。

```
可用性 = 稼働時間 / (稼働時間 + 停止時間) * 100%
```

一般に、可用性は達成できる「ナイン」の数で表現されます。たとえば、99% の可用性であれば、2 ナインとなります。ナインの追加は、配備設計に大きく影響します。次の表では、24 時間 365 日、つまり合計 8,760 時間稼働するシステムに、可用性のナインを追加した場合の予定外の停止時間を数値化しています。

表 **3–3** 1 年間無休で (8,760 時間) 稼働するシステムの予定外の停止時間

ナインの数	使用可能割合	予定外の停止時間
2	99%	88 時間
3	99.9%	9 時間
4	99.99%	45 分
5	99.999%	5 分

耐障害システム

4 つまたは 5 つのナインが要求される可用性を実現するには、通常は耐障害のシステムが必要となります。耐障害システムは、ハードウェアまたはソフトウェアに障害が発生した場合にも、継続してサービスを提供できる必要があります。一般に、耐障害性は、主要サービスを提供するハードウェア (CPU、メモリ、ネットワークデバイスなど) とソフトウェアの両方に冗長性を持たせることで実現されます。

シングルポイント障害は、クリティカルパスの一部でありながら冗長コンポーネントによってバックアップされていないハードウェアまたはソフトウェアコンポーネントです。このコンポーネントで障害が発生した場合、システムはサービスを提供できなくなります。耐障害システムを設計するときは、潜在的なシングルポイント障害を特定し、それを取り除きます。

耐障害システムの実装と維持は、高額になる可能性があります。可用性に関するビジネス要件の本質を理解し、それらの要件に見合った可用性ソリューションの戦略とコストを考慮する必要があります。

サービスの可用性の優先順位付け

ユーザー側から見ると、多くの場合、システム全体の可用性よりも、サービス単位の可用性のほうが重要です。たとえば、インスタントメッセージングサービスが利用できなくなっても、通常、その他のサービスの可用性には影響がまったくないか、あってもごく軽微なものです。しかし、その他のサービスの多くが依存するサービス (Directory Server など) が利用できなくなると、はるかに広範な影響をもたらします。高い可用性の仕様には、可用性の向上を必要とするユースケースと使用パターン分析が明確に参照されている必要があります。

可用性の必要性を優先度の順にリストしておくと便利です。次の表は、各種サービスの可用性の優先順位を示しています。

表 **3–4** 優先度順のサービスの可用性

優先順位	サービスの種類	説明
1	ミッションクリティカル	常時使用可能である必要のあるサービス。たとえば、アプリケーションに対するデータベースサービス (LDAP ディレクトリなど) です。
2	使用可能である必要あり	使用可能である必要はあるが、パフォーマンスが低下しても問題はないサービス。たとえば、ビジネス環境によっては、メッセージングサービスの可用性は重要視されない場合があります。
3	延期可能	特定期間内に使用可能であることが必要となるサービス。たとえば、ビジネス環境によっては、カレンダサービスの可用性は重要視されない場合があります。
4	省略可能	恒久的に延期しても問題のないサービス。たとえば、ビジネス環境によっては、インスタントメッセージングサービスは有用ではあっても必要ではないと考えられる場合があります。

サービスの損失

可用性設計では、可用性が低下した場合や、コンポーネントが失われた場合の状況も考慮します。これには、接続していたユーザーがセッションを再起動する必要があるかどうか、また 1 つの領域で発生した障害がシステムのほかの領域にどのような影響を与えるかについて考慮することも含まれます。QoS 要件には、これらのシナリオが考慮されていて、配備によるこれらの状況への対処が示されている必要があります。

スケーラビリティー

スケーラビリティーは、既存のユーザーまたは増大したユーザーベースからの追加負荷にシステムが対応できるように、システムに容量を追加する能力です。通常、スケーラビリティーは追加リソースを必要としますが、配備アーキテクチャーの設計変更や、追加リソースの追加に要する時間によるサービスの喪失は考慮の対象となりません。

可用性と同様に、スケーラビリティーはシステム全体よりも、システムが提供する各サービスで重要視されます。ただし、他のサービスが依存する Directory Server のようなサービスのスケーラビリティーは、システム全体に影響する可能性があります。

配備の成長の予定がビジネス要件に明確に示されていない限り、スケーラビリティー要件を QoS 要件として指定する必要は必ずしもありません。ただし、ソリューションライフサイクルの配備設計フェーズでは、システムのスケーラビリティーを確保するために何らかの許容範囲を配備アーキテクチャーに必ず追加する必要があります。 これは、スケーラビリティーについての QoS 要件が指定されていない場合にも当てはまります。

成長予測

スケーラビリティー要件を決定するためのシステムの成長予測には、実行できない可能性のある予測、見積もり、推測なども含まれます。スケーラブルなシステムの要件を決定する際の鍵は、次の 3 つです。

- 高パフォーマンス設計の戦略**:** パフォーマンス要件を指定する際に、時間の経過とともに増す可能性のある負荷を処理できるだけの潜在処理能力を盛り込みます。また、予算の制約の範囲内で最大の可用性を設計します。この戦略を採用することで、成長を吸収し、システムの規模を柔軟に拡張するための効果的なマイルストーンを設定できます。
- 段階的な配備**:** 段階的な配備は、リソースの追加予定を立てる上で役立ちます。システムの規模拡大について明確な目標を設定します。通常、達成期限は、スケーラビリティーの評価を行う特定の日を考慮した、負荷に基づいた要件です。
- 広範なパフォーマンス監視**:** パフォーマンスの監視は、システムへのリソースの追加時期を決定する上で役立ちます。パフォーマンス監視要件により、保守やアップグレードを担当するオペレータおよび管理者にガイドラインを示すことができます。

次の表は、スケーラビリティー要件を決定する上で考慮が必要な要因を示しています。

表 **3–5** スケーラビリティー要因

項目	説明
使用パターンの分析	既存のデータを分析し、現在の (または予定される) ユーザーベースの使用パターンを把握します。現在のデータを利用できない場合は、業界のデータまたは市場予測を分析します。
妥当な最大スケールの設計	既知の需要と潜在的な需要の両方について、必要となる最大スケールの目標を設定します。 多くの場合、これは既存のユーザー負荷のパフォーマンス予想と、将来の負荷に関する妥当な予想に基づく、24 か月間の見積もりです。見積もりの対象期間は、予想の信頼性によって大きく異なります。
適切な達成目標の設定	配備設計を段階的に実装することで、予期せぬ成長に対応するための短期要件を満たすことができます。システムリソースの追加に関する達成目標を設定します。 例: ■ 予算獲得 (四半期ベース、年次ベースなど) ■ ハードウェアおよびソフトウェアを購入する際のリードタイム (1 〜 6 週間など) ■ バッファー (成長予測に応じて 10 〜 100%)
新興技術の採用	より高速なプロセッサや Web サーバーなどの新しい技術について知り、これらの新しい技術が、基本となるアーキテクチャーのパフォーマンスにどの程度影響するかを把握します。

セキュリティー要件

セキュリティーは、配備されたシステムの全レベルが関わる複雑な事項です。セキュリティー要件の作成は、セキュリティー脅威の特定およびそれらに対抗する戦略の策定を中心に進めます。このセキュリティー分析には、次のステップがあります。

1. 重大な資産の特定
2. それらの資産に対する脅威の特定
3. 組織にリスクをもたらす脅威を顕在化する脆弱性の特定
4. 組織に対するリスクを低減するセキュリティー計画の作成

セキュリティー要件の分析には、管理職、ビジネスアナリスト、情報技術担当者など、組織の幅広い利害関係者が関与する必要があります。多くの場合、セキュリティー設計者がセキュリティー対策の設計と実装におけるリーダーに指名されます。

次の節では、セキュリティー計画の対象となる領域の一部を説明します。

セキュリティー計画の要素

システムのセキュリティーを計画することは、配備設計の一部で、実装の成功に不可欠な部分です。セキュリティーを計画する際は、次の要素を考慮します。

- 物理的なセキュリティー**:** 物理的なセキュリティーは、ルーター、サーバー、サーバールーム、データセンター、またはインフラストラクチャーを構成するその他の部分に対する物理的なアクセスに対応します。権限のない人物がサーバールームに入ってルーターのプラグを抜くことができる場合、その他のセキュリティー対策の効力が脅かされます。
- ネットワークセキュリティー**:** ネットワークセキュリティーは、ファイアウォール、セキュアアクセスゾーン、アクセス制御リスト、およびポートアクセスを使用した、ネットワークへのアクセスに対応します。ネットワークセキュリティー対策のために、不正なアクセス、改ざん、およびサービス拒否 (DoS) 攻撃に対する戦略を策定します。
- アプリケーションおよびアプリケーションデータセキュリティー**:** アプリケーションおよびアプリケーションデータセキュリティーは、認証と承認の手順とポリシーを使用した、ユーザーアカウント、企業のデータ、および企業向けアプリケーションへのアクセスに対応しています。この領域には、次のポリシーの定義が含まれます。
  - パスワードポリシー
  - 管理者アクセスではなく、ユーザーに委任された委任管理などのアクセス権限
  - アカウントの無効化
  - アクセス制御
  - セキュリティー保護されたデータ転送やデータに署名する際の証明書の使用などの暗号化ポリシー
- 個人のセキュリティー順守**:** 組織全体のセキュリティーポリシーでは、作業環境と対策が定められます。 これらは、その他のセキュリティー対策が確実に設計どおり実施されるために、すべてのユーザーが順守する必要があります。通常、セキュリティーについてのハンドブックやマニュアルを作成し、ユーザーにセキュリティー順守についてのトレーニングを提供します。セキュリティーポリシー全体が効果的であるためには、適切なセキュリティーの順守を組織文化の一部にする必要があります。

潜在処理能力

潜在処理能力は、使用状況に例外的なピーク負荷が生じた場合に、追加リソースなしで配備がそれを処理する能力です。通常は、潜在処理能力について QoS 要件を直接指定することはありませんが、このシステム品質は、システムの可用性、パフォーマンス、スケーラビリティーの要因となります。

保守性要件

保守性は、システムの監視、発生した問題の解決、システムに対するユーザーの追加および削除、ハードウェアおよびソフトウェアコンポーネントのアップグレードなどのタスクを含め、配備されたシステムの保守の容易さを意味します。

保守性の要件を計画するときは、次の表に示される事項を考慮します。

表 **3–6** 保守性要件について考慮すべき事項

項目	説明
停止時間の計画	特定のサービスが利用できなくなる、または部分的に利用できなくなる保守作業を特定します。 ユーザーに対してシームレスに行われる保守とアップグレードもありますが、サービスの中断を必要とするものもあります。ユーザーが事前に停止時間に備えることができるように、可能であれば、停止時間を必要とする保守作業についてユーザーと共同で予定を立てます。
使用パターン	使用パターンを特定して、保守を予定するのに最も適した時間帯を決定します。 たとえば、通常のピーク利用が一般的な業務時間帯であるシステムの場合、夜または週末に保守を予定します。地理的に分散したシステムの場合、この時間帯の特定はより困難になります。
可用性	多くの場合、可用性の設計は保守性に反映されます。保守とアップグレードのための停止時間を最小化する戦略は、可用性戦略を中心に展開されます。高度な可用性を必要とするシステムでは、保守、アップグレード、修復のための機会は限られています。 可用性要件に対処するための戦略は、保守やアップグレードの処理に影響します。たとえば、地理的に分散したシステムでは、サービスの可用性は、保守期間中のリモートサーバーへの作業負荷のルーティング機能に依存します。 また、高度な可用性を必要とするシステムは、人的な介入をほとんど必要とせずに、システムを自動的に再起動する、より洗練されたソリューションを必要とします。
診断と監視	診断ツールと監視ツールを定期的に実行して問題領域を特定することで、システムの安定性を向上できます。 定期的なシステムの監視により、発生前に問題を回避したり、可用性戦略に基づく作業負荷のバランスに役立てたりすることができます。 また、保守と停止時間の計画も改善されます。

サービスレベル要件

サービスレベル契約 (SLA: Service Level Agreement) には、最小パフォーマンス要件と障害時にこれらの要件を満たすために提供する必要のあるカスタマーサポートのレベルと範囲を規定します。サービスレベル要件は、SLA が基づく条件を特定するシステム要件です。

サービスレベル要件は、QoS 要件と同様にビジネス要件に基づいて特定され、配備されたシステムが満たす必要のあるシステム全体の品質についての保証を示します。サービスレベル契約は契約の一環とみなされるので、サービスレベル要件の仕様は明白である必要があります。サービスレベル要件は、どのような条件のもとで要件がテストされるのかを具体的に定義し、何をもって要件の不適合とするかを厳密に定義する必要があります。

第 4 章

論理設計

ソリューションライフサイクルの論理設計フェーズでは、ソリューションの論理コンポーネント間の相互関係を示す論理アーキテクチャーを設計します。論理アーキテクチャーと技術要件フェーズの使用パターン分析から、配備シナリオが形成されます。これは、配備設計フェーズへのインプットとなります。

この章で説明する内容は、次のとおりです。

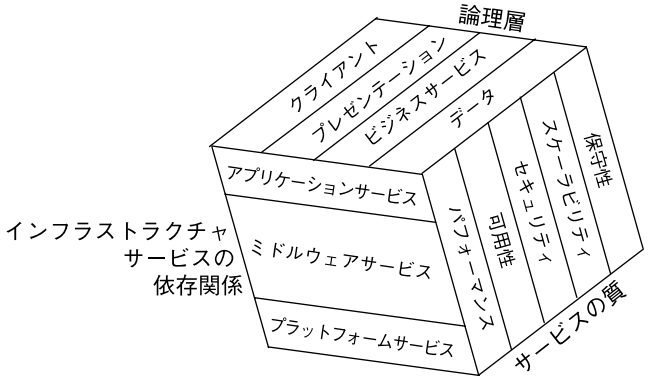
論理アーキテクチャーについて

論理アーキテクチャーは、ソリューションの実装に必要とされるソフトウェアコンポーネントを特定し、コンポーネント間の相互関係を示します。論理アーキテクチャーと、技術要件フェーズで決定されたサービス品質要件により、配備シナリオが形成されます。配備シナリオは、次のフェーズである配備設計で行なわれる配備アーキテクチャーの設計の基礎となります。

論理アーキテクチャーを設計する際は、ユーザーにサービスを提供するコンポーネントだけでなく、必要なミドルウェアサービスおよびプラットフォームサービスを提供するその他のコンポーネントも特定する必要があります。インフラストラクチャーサービスの依存関係と論理層は、この分析を実行するための 2 つの相補的な手段となります。

インフラストラクチャーサービスの依存関係と論理層は、Sun Java™ Enterprise System が基づくソリューションアーキテクチャーの 3 つの次元のうちの 2 つです。次に示す 3 つの次元は、47 ページの「論理アーキテクチャーについて」にも示されています。

- インフラストラクチャーサービスの依存関係**:** エンタープライズサービスを提供する、対話型のソフトウェアコンポーネント。これらのソフトウェアコンポーネントは、分散型のコンポーネントが相互に通信したり相互動作したりすることを可能にする基本のインフラストラクチャーサービスのセットを必要とします。
- 論理層**:** ソフトウェアコンポーネントの論理的構成を表す層。 提供するサービスの性質に基づいて、ソフトウェアコンポーネントの論理的および物理的な独立性を表しています。
- サービス品質**:** システムのサービス品質。 パフォーマンス、可用性、スケーラビリティーなど、ソフトウェアソリューションの設計上および運用上の特定の側面を表します。



注 – Java Enterprise System アーキテクチャーの概念に関する詳細については、『Sun Java Enterprise System 2005Q4 Technical Overview』の「Java Enterprise System アーキテクチャー」の章を参照してください。

論理アーキテクチャーでは、必要なコンポーネントとそれらの依存関係を示すことで、インフラストラクチャーサービスのレベルを表します。また、論理アーキテクチャーでは、プレゼンテーション、ビジネス、およびデータの各サービスを表す論理層にコンポーネントを割り当てます。これらの層には、最終的にクライアント層からアクセス可能です。サービス品質要件は論理アーキテクチャーでモデル化されませんが、配備シナリオにおいて、論理アーキテクチャーと対応付けられます。

論理アーキテクチャーの設計

論理アーキテクチャーを設計する際は、技術要件フェーズで特定したユースケースを使用して、ソリューションに必要なサービスを提供する Java Enterprise System コンポーネントを決定します。また、最初に特定したコンポーネントにサービスを提供するコンポーネントもすべて識別する必要があります。

Java Enterprise System コンポーネントを、それらが提供するサービスの種類に応じて多層アーキテクチャー内に配置します。多層アーキテクチャーの一部としてコンポーネントを理解することは、あとで、コンポーネントによって提供されるサービスをどのように分散するか決定する上で役立ちます。 また、スケーラビリティーや可用性などのサービス品質を実装するための戦略を決定する上でも役立ちます。

さらに、セキュアアクセスゾーン内に配置された論理コンポーネントの、別のビューも提供できます。セキュアアクセスゾーンの例は、65 ページの「アクセスゾーン」の節を参照してください。

Java Enterprise System コンポーネント

Java Enterprise System は、エンタープライズサービスを提供する、対話型のソフトウェアコンポーネントで構成されています。 これらを使用して、企業のソリューションを構築できます。次の図は、Java Enterprise System が提供する主要なソフトウェアコンポーネントを示しています。Java Enterprise System コンポーネントとそれらが提供するサービスの詳細については、『Sun Java Enterprise System 2005Q4 Technical Overview』を参照してください。

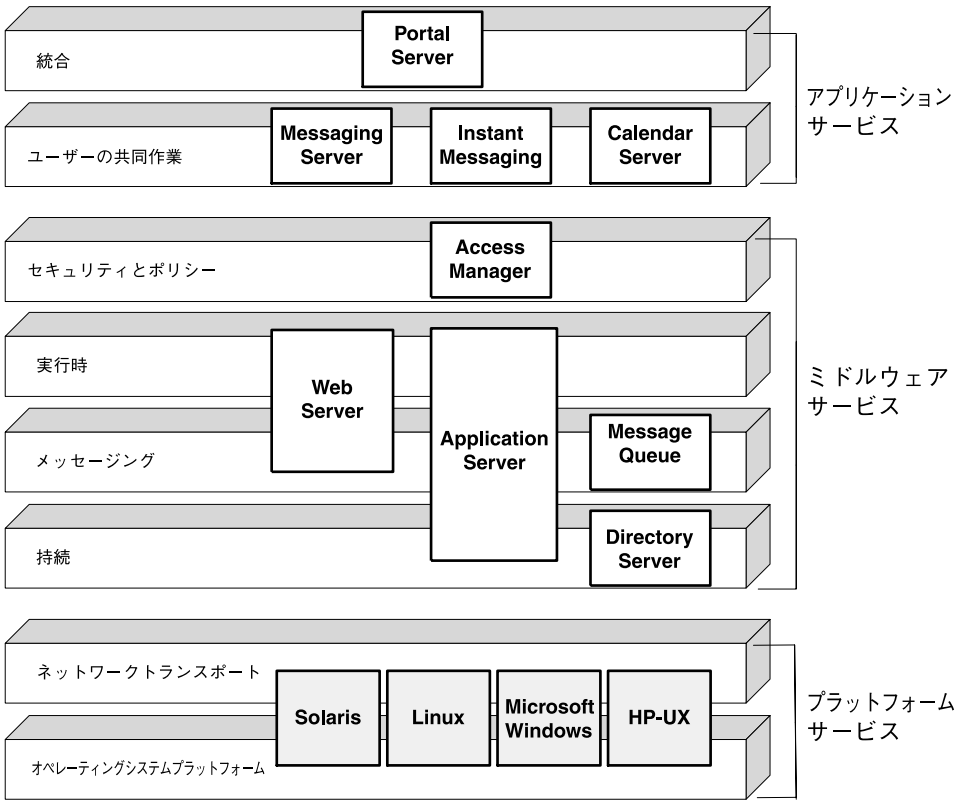


図 **4–1** Java Enterprise System コンポーネント

コンポーネントの依存関係

論理アーキテクチャー用に Java Enterprise System コンポーネントを特定する際は、サポートするコンポーネントも識別する必要があります。たとえば、Messaging Server を論理アーキテクチャーに必要なコンポーネントとして特定した場合、Directory Server と、場合によっては Access Manager も論理アーキテクチャーに含める必要があります。Messaging Server は、ディレクトリサービスに関して Directory Server に依存し、シングルサインオンを必要とするソリューションに関して Access Manager に依存します。

次の表は、Java Enterprise System コンポーネントの依存関係を示しています。主要コンポーネント間の依存関係を視覚的に表現したものについては、50 ページの「コンポーネントの依存関係」を参照してください。論理アーキテクチャーを設計する際は、この表とそれに付随する図を参照して、設計における依存コンポーネントを決定します。

表 **4–1** Java Enterprise System コンポーネントの依存関係

Java Enterprise System コンポーネント	依存するコンポーネント
Application Server	Message Queue
	Directory Server (省略可能)
Calendar Server	Messaging Server (電子メール通知サービス用)
	Access Manager (シングルサインオン用)
	Web Server (Web インタフェース用)
	Directory Server
Communications Express	Access Manager (シングルサインオン用)
	Calendar Server
	Messaging Server
	Instant Messaging
	Web Server (Web インタフェース用)
	Directory Server
Directory Proxy Server	Directory Server
Directory Server	なし
Access Manager	Application Server
	Web Server
	Directory Server
Instant Messaging	Access Manager (シングルサインオン用)
	Directory Server
Message Queue	Directory Server (省略可能)
Messaging Server	Access Manager (シングルサインオン用)
	Web Server (Web インタフェース用)
	Directory Server

表 **4–1** Java Enterprise System コンポーネントの依存関係　(続き)

Java Enterprise System コンポーネント	依存するコンポーネント
Portal Server	Portal Server チャネルの使用を設定する場合に必要なコンポーネント:
	Calendar Server
	Messaging Server
	Instant Messaging
	Access Manager (シングルサインオン用)
	Application Server
	Web Server
	Directory Server
Portal Server Secure Remote Access	Portal Server
Web Server	Access Manager (省略可能、アクセス制御用)

注 – 50 ページの「コンポーネントの依存関係」に示されている Java Enterprise System コンポーネント間の依存関係には、コンポーネントのすべての依存関係が一覧表示されているわけではありません。50 ページの「コンポーネントの依存関係」には、インストール計画時に考慮する必要のある依存関係は一覧表示されていません。Java Enterprise System の依存関係の全リストについては、『Sun Java Enterprise System 2005Q4 インストールガイド(UNIX 版)』を参照してください。

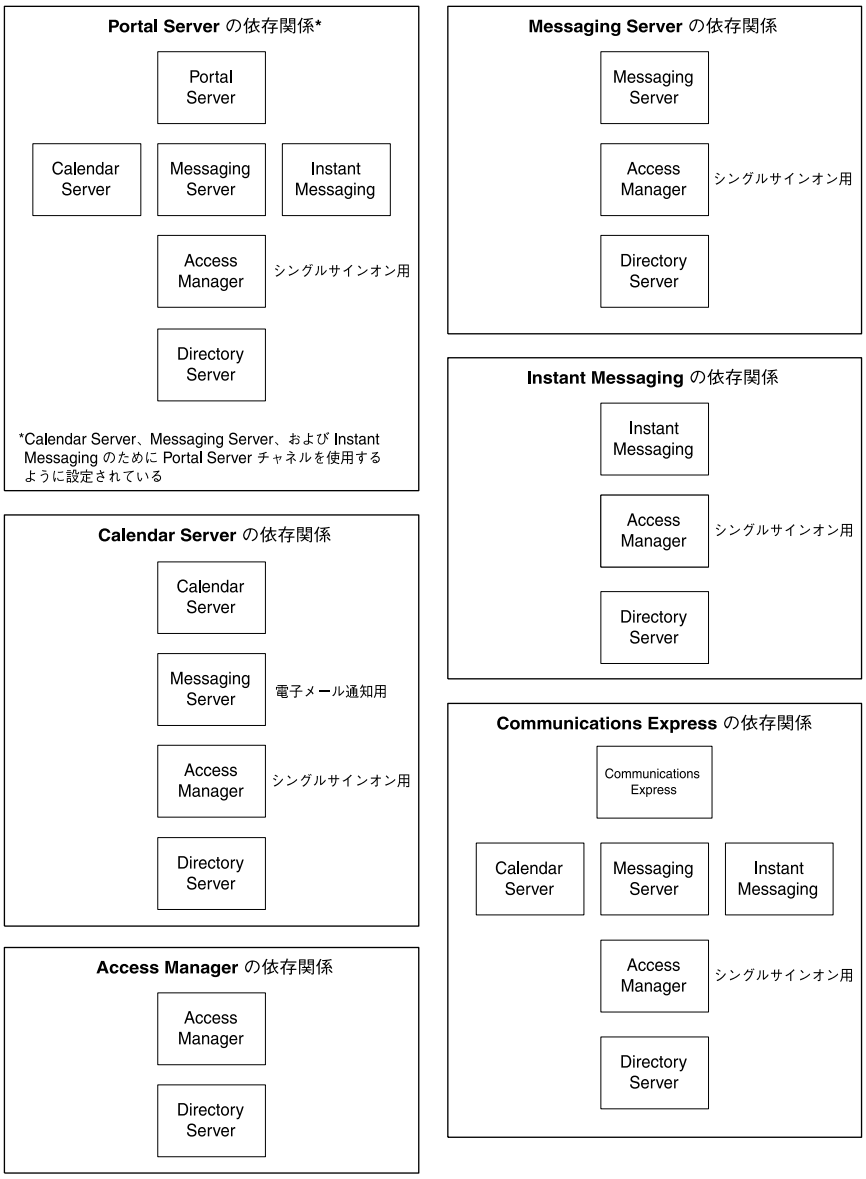


図 **4–2** Java Enterprise System コンポーネントの依存関係

Web コンテナサポート

前節の50 ページの「コンポーネントの依存関係」では、Portal Server および Access Manager が稼働する Web コンテナについては説明されていません。この Web コンテナは、Application Server、Web Server、またはサードパーティー製品によって提供されます。Portal Server または Access Manager を含んだ論理アーキテクチャーを設計する場合は、それらのコンポーネントに必要な Web コンテナを必ず考慮してください。

Messaging Server によって提供される論理的に区別されるサービス

論理的に区別される次のサービスを提供する別個のインスタンスを提供するように Java Enterprise System Messaging Server を設定できます。

- メッセージ転送エージェント
- メッセージマルチプレクサ
- メッセージエクスプレスマルチプレクサ
- メッセージストア

Messaging Server のこれらの各種設定は、別個の物理サーバーに配備可能で、論理アーキテクチャーの別個の層に表すことができる機能を提供します。Messaging Server のこれらの設定は、異なる層にある論理的に区別されるサービスであるため、論理アーキテクチャーを設計する際は、それらを論理的に区別されるコンポーネントと考えます。論理的に区別されるコンポーネントの例は、57 ページの「論理アーキテクチャーの例」の節を参照してください。

次の表は、Messaging Server の論理的に区別される設定について説明しています。

表 **4–2** Messaging Server の設定

サブコンポーネント	説明
メッセージ転送エージェント (MTA)	SMTP 接続の処理、電子メールのルーティング、および適切なメッセージストアへのメッセージの配信を行なうことで、電子メールの送信をサポートします。MTA コンポーネントは、企業外部から送信された着信電子メールまたは企業内部から送信された発信電子メールをサポートするように設定できます。
メッセージストア (STR)	電子メールメッセージの検索と保存を提供します。
メッセージマルチプレクサ (MMP)	IMAP または POP のいずれかのプロトコルを使用して電子メールクライアントのメッセージストアにアクセスすることで、電子メールの検索をサポートします。
メッセージエクスプレスマルチプレクサ (MEM)	Web ベース (HTTP) クライアントの代わりにメッセージストアにアクセスすることで、電子メールの検索をサポートします。

アクセスコンポーネント

Java Enterprise System には、システムサービスへのアクセスを提供するコンポーネントもあります。 このアクセスは多くの場合、企業のファイアウォール外から行なわれるものです。メッセージマルチプレクサ用に設定された Messaging Server など、Messaging Server の一部の設定では、ネットワークアクセスも提供されます。次の表は、システムサービスへのリモートアクセスを提供する Java Enterprise System コンポーネントについて説明しています。

表 **4–3** リモートアクセスを提供する Java Enterprise System コンポーネント

コンポーネント	説明
Directory Proxy Server	拡張されたディレクトリアクセス、スキーマ互換性、ルーティング、および複数の Directory Server インスタンス間でのロードバランスを提供します。
Portal Server, Portal Server Secure Remote Access	内部ポータルやインターネットアプリケーションなど、Portal Server のコンテンツやサービスへの、企業ファイアウォール外からのセキュリティー保護されたインターネットアクセスを提供します。
Portal Server, Portal Server Mobile Access	Portal Server に対する、モバイルデバイスからのワイヤレスアクセスおよび音声アクセスを提供します。
Messaging Server メッセージマルチプレクサ (MMP)	Web ベース (HTTP) クライアントの代わりにメッセージストアにアクセスすることで、電子メールの検索をサポートします。

65 ページの「アクセスゾーン」の節の例で示されているように、リモートアクセスを提供するコンポーネントは、通常、セキュアアクセスゾーンに配備されます。

多層アーキテクチャー設計

Java Enterprise System は、提供する機能に応じてサービスが各層に配置される多層アーキテクチャー設計に特に適しています。各サービスは論理上独立していて、同じ層のサービスまたは異なる層のサービスのいずれからもアクセス可能です。次の図は、企業アプリケーションの多層アーキテクチャーモデルです。 クライアント、プレゼンテーション、ビジネスサービス、データの各層が示されています。

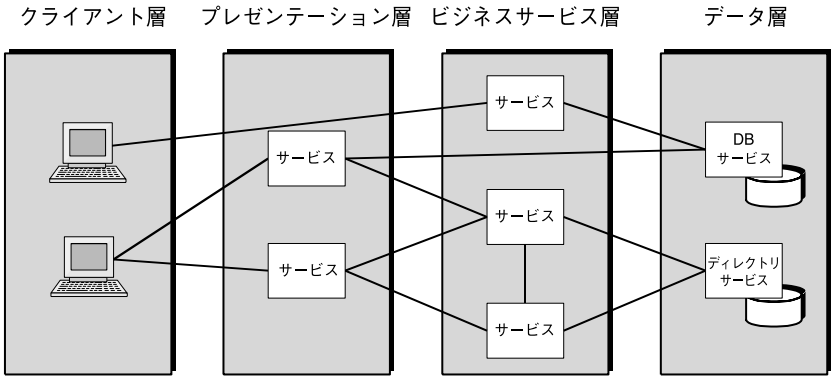


図 **4–3** 多層アーキテクチャーモデル

次の表は、55 ページの「多層アーキテクチャー設計」に示されている論理層について説明しています。

表 **4–4** 多層アーキテクチャーの論理層

層	説明
クライアント層	情報をエンドユーザーに提示するクライアントアプリケーションが配置されます。Java Enterprise System の場合、これらのアプリケーションは、通常、メールクライアント、Web ブラウザ、またはモバイルアクセスクライアントとなります。
プレゼンテーション層	エンドユーザーにデータを表示するサービスを提供し、ユーザーがプレゼンテーションを処理および操作できるようにします。たとえば、Web メールクライアントまたは Portal Server コンポーネントの場合、ユーザーは、受信した情報の表示内容に変更を加えることができます。
ビジネスサービス層	通常、プレゼンテーション層またはビジネスサービス層のほかのサービスに提供するデータ、またはクライアント層のクライアントに直接提供するデータをデータ層から取り出す、バックエンドサービスを提供します。たとえば、Access Manager は、ほかの Java Enterprise System コンポーネントにアイデンティティーサービスを提供します。
データ層	プレゼンテーション層またはビジネスサービス層のサービスによってアクセスされるデータベースサービスを提供します。たとえば、Directory Server は、ほかのサービスに LDAP ディレクトリアクセスを提供します。

多層アーキテクチャー設計には、いくつかの利点があります。多層アーキテクチャーでの機能に応じたサービスの配置は、配備設計フェーズで、ネットワークにどのようにサービスを分散するか決定する上で役立ちます。また、アーキテクチャー内のコンポーネントがほかのコンポーネントのサービスにどのようにアクセスするかについても確認できます。この視覚化により、可用性、スケーラビリティー、セキュリティーなどのサービス品質ソリューションの計画が行ないやすくなります。

論理アーキテクチャーの例

ここでは、Java Enterprise System ソリューションの論理アーキテクチャーの例をいくつか示します。これらの例は、論理コンポーネントを多層アーキテクチャーの適切な層に配置する方法を示します。 その後、ユースケースを調べることで、コンポーネント間の関係を分析します。ここで示される論理アーキテクチャーの例を、Java Enterprise System ソリューションの論理アーキテクチャー設計を理解する基礎として使用してください。

最初の例は、Messaging Server の論理的に区別されるコンポーネントがほかのコンポーネントとどのように対話するかを示す、基本的な Messaging Server ソリューションです。2 番目の例は、約 1,000 ～ 5,000 人の従業員を持つ中規模企業に適した、アイデンティティーベースの配備ソリューションの論理アーキテクチャーを示しています。

Messaging Server の例

次の図は、Messaging Server 配備の基本的な論理アーキテクチャーを示しています。この論理アーキテクチャーは、Messaging Server に必要な、論理的に区別されるコンポーネントのみを示しています。この後の各図は、これらのコンポーネント間の関係を示しています。

注 – 62 ページの「アイデンティティーベースの通信の例」で示されているように、通常、Messaging Server の配備は、ほかの Java Enterprise System コンポーネントを含む企業ソリューションの一部です。

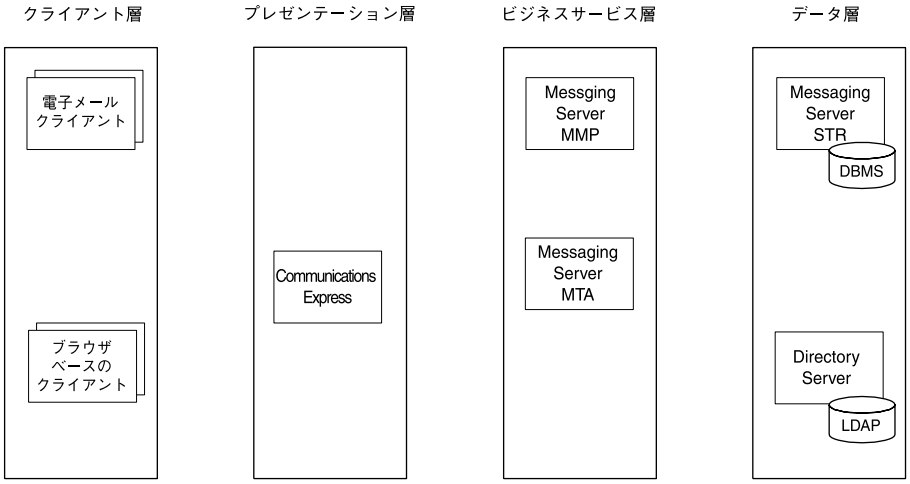


図 **4–4** Messaging Server 配備の論理アーキテクチャー

次の表は、57 ページの「Messaging Server の例」に示されているコンポーネントについて説明しています。

表 **4–5** 論理アーキテクチャー上の Messaging Server コンポーネント

コンポーネント	説明
電子メールクライアント	電子メールの参照および送信に使用するクライアントアプリケーション。
Messaging Server MTA	電子メールメッセージを受信、ルーティング、転送、および配信するために、メッセージ転送エージェント (MTA) として設定された Messaging Server。
Messaging Server MMP	検索および保存用の適切なメッセージストアに接続をルーティングするために、メッセージマルチプレクサ (MMP) として設定された Messaging Server。MMP は Directory Server にアクセスしてディレクトリ情報を検索し、適切なメッセージストアを決定します。
Messaging Server STR	電子メールメッセージの検索および保存用のメッセージストアとして設定された Messaging Server。
Directory Server	LDAP ディレクトリデータに対するアクセスを提供します。

論理アーキテクチャーでは、Messaging Server コンポーネント用のサービスのレプリケーションを指定しません。たとえば、企業の配備では、通常、着信と発信で異なる MTA インスタンスが作成されますが、57 ページの「Messaging Server の例」は 1 つのMTA コンポーネントだけを示しています。論理コンポーネントの複数インスタンスへのレプリケーションは、設計上の意思決定として、配備設計フェーズで行います。

Messaging Server ユースケース

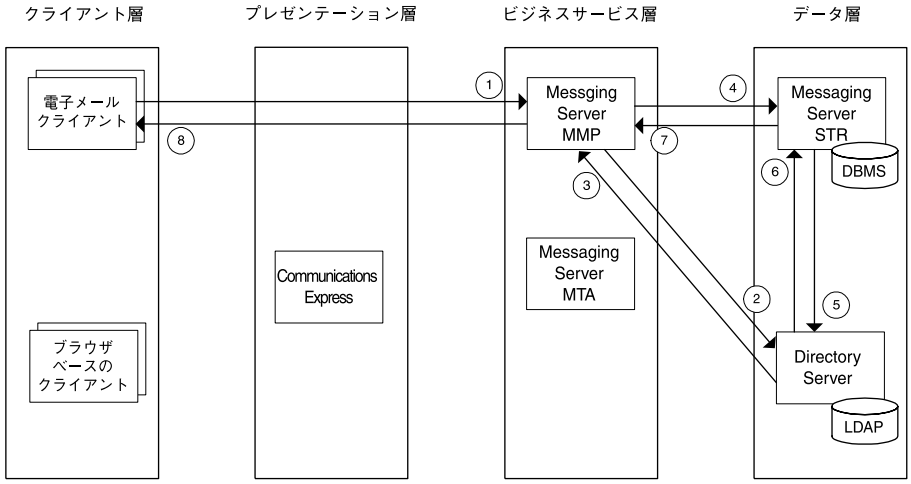
ユースケースは、アーキテクチャーの論理コンポーネント間の関係を識別する上で役立ちます。ユースケースに基づいてコンポーネント間のインタラクションをマップすることで、コンポーネントのインタラクションを図で表現できます。 その図は配備設計で役立ちます。

通常、配備設計の前に、各ユースケースを分析してコンポーネントのインタラクションを決定します。次の 3 つのユースケースは、Messaging Server の場合に一般的なケースで、論理コンポーネント間のインタラクションを示しています。

▼ ユースケース 1: ユーザーが正常に Messaging Server にログインする

手順

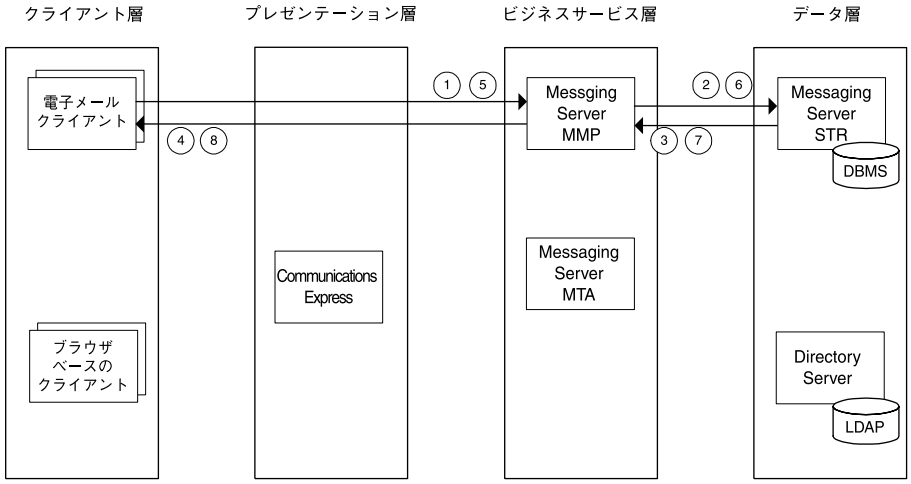
1. 電子メールクライアントは、ログイン情報を **Messaging Server** マルチプレクサ **(MMP)** に送信する。
2. **MMP** は、ユーザー **ID** とパスワードの検証を **Directory Server** に要求する。
3. **Directory Server** は **MMP** に検証結果を返す。
4. **MMP** は、**Messaging Server** メッセージストア **(STR)** にメッセージリストを要求する。
5. **STR** は、**Directory Server** にユーザーの **LDAP** レコードを要求する。
6. **Directory Server** は、ユーザーの **LDAP** レコードを **STR** に返す。
7. **STR** は、**MMP** にメッセージリストを返す。
8. **MMP** は、電子メールクライアントにメッセージリストを転送する。



▼ ユースケース 2: ログインしたユーザーがメールを参照し、削除する

手順

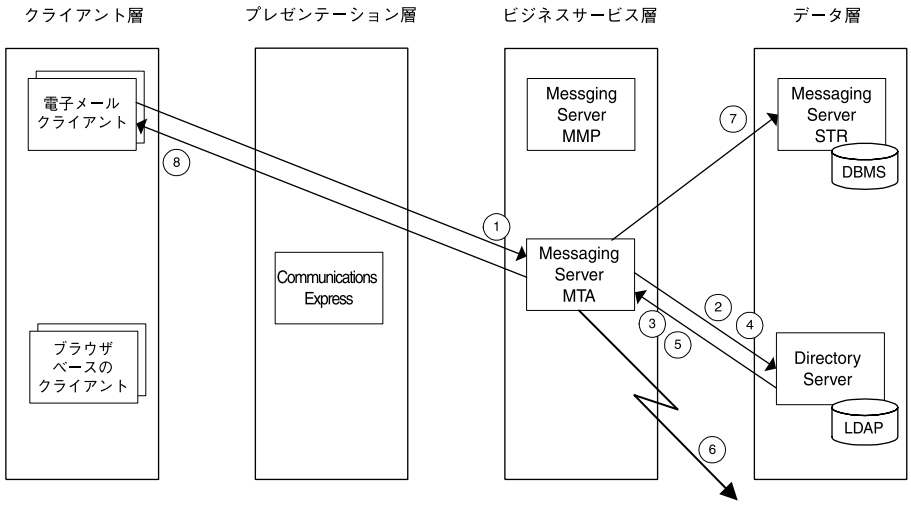
1. 電子メールクライアントは、参照するメッセージを **Messaging Server** マルチプレクサ **(MMP)** に要求する。
2. **MMP** は、**Messaging Server** メッセージストア **(STR)** にメッセージを要求する。
3. **STR** は、**MMP** にメッセージを返す。
4. **MMP** は、電子メールクライアントにメッセージを転送する。
5. 電子メールクライアントは、メッセージを削除するアクションを **MMP** に送信する。
6. **MMP** は、メッセージを削除するアクションを **STR** に転送する。
7. **STR** は、データベースからメッセージを削除し、**MMP** に確認を送信する。
8. **MMP** は、電子メールクライアントに削除の確認を転送する。



▼ ユースケース 3: ログインしたユーザーが電子メールメッセージを送信する

手順

1. 電子メールクライアントは、クライアントで作成したメッセージを **Messaging Server** メッセージ転送エージェント **(MTA)** に送信する。
2. **MTA** は、ユーザー **ID** とパスワードの検証を **Directory Server** に要求する。
3. **Directory Server** は **MTA** に検証結果を返す。
4. **MTA** は、**Directory Server** で、各受取人の宛先ドメインをチェックする。
5. **Directory Server** は、各受取人の宛先ドメインを **MTA** に返す。
6. **MTA** は、各受取人にメッセージを転送する。
7. **MTA** は、**Messaging Server** メッセージストア **(STR)** にメッセージを転送し、送信トレイにメッセージを保存する。
8. **MTA** は、電子メールクライアントに確認を送信する。



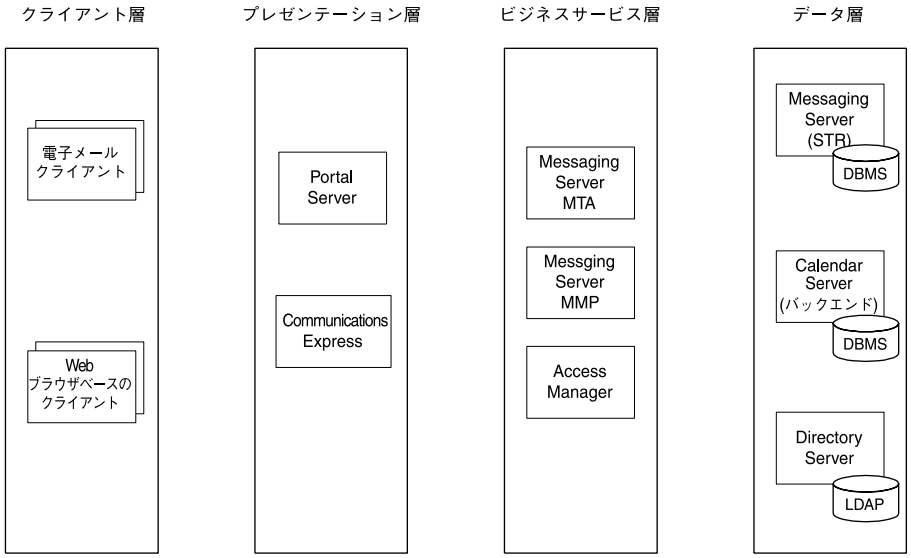
アイデンティティーベースの通信の例

この例では、約 1,000 ～ 5,000 人の従業員を持つ中規模企業用のアイデンティティーベースの通信を示します。通常、論理アーキテクチャーの設計を行なうためには、徹底的なビジネス分析の後に詳細な技術要件分析を行なう必要があります。ただし、これは理論上の例であるので、次のビジネス要件が決定されたものと仮定します。

- 企業の従業員は、内部の Web サイト、通信サービス、カレンダサービス、およびその他のリソースに対する個人用カスタマイズされたアクセスを必要としている。
- 企業全体の認証と承認により、内部の Web サイトなどのサービスに対するアクセスが許可される。
- 1 つのアイデンティティーは、すべてのエンタープライズサービスにおいて追跡され、内部の Web サイトなどのサービスへのアクセスを許可するシングルサインオンが有効になる。

この例のユースケースでは、ログイン手順、電子メールの参照、電子メールの送信、ポータルの個人用カスタマイズ、カレンダの同期、およびその他の類似のユーザーアクティビティーについて詳細化されています。

次の図は、このタイプのアイデンティティーベースの通信ソリューションに対する論理アーキテクチャーを示します。



アイデンティティーベースの通信シナリオ例のユースケース

通常、この種の配備ソリューションの場合、ソリューションが提供するサービスに対するユーザーのインタラクションを説明する多数の詳細なユースケースがあります。この例では、ユーザーが Web ブラウザクライアントからポータルにログインするときの、コンポーネント間のインタラクションを取り上げます。例では、ログインシナリオが次の 2 つのユースケースに分割されています。

- ユーザーがログインし、認証されると、Portal Server はユーザーのポータル設定を検索する。
- Portal Server は、電子メールおよびカレンダ情報を検索して、Web クライアントに表示する。

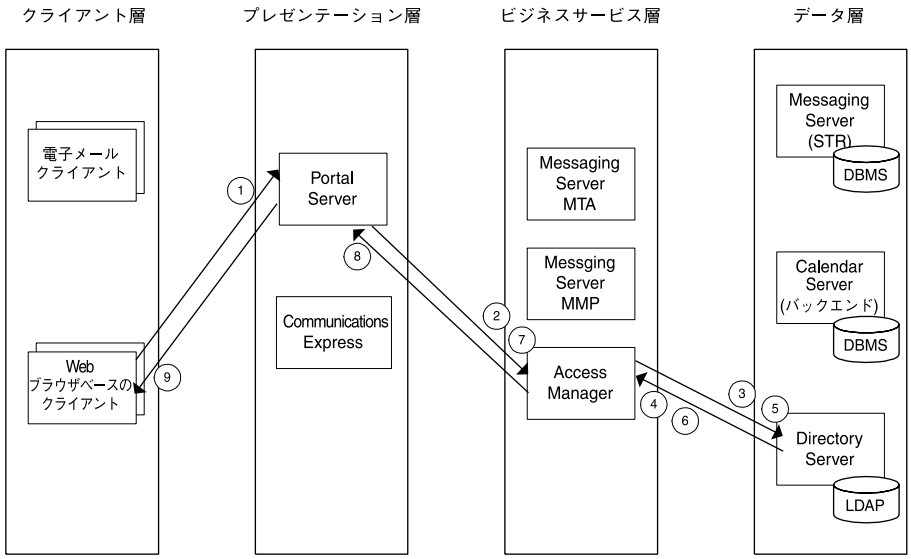
2 つのユースケースは、1 つの拡張されたユースケースと考えることができます。しかし、この例では、単純化のためにユースケースは分割されています。

▼ ユースケース 1: ユーザーが正常にログインし、ポータルがユーザーの設定を検索する

手順

1. **Web** ブラウザクライアントが、ユーザー **ID** とパスワードを **Portal Server** に送信する。
2. **Portal Server** は、**Access Manager** に認証を要求する。

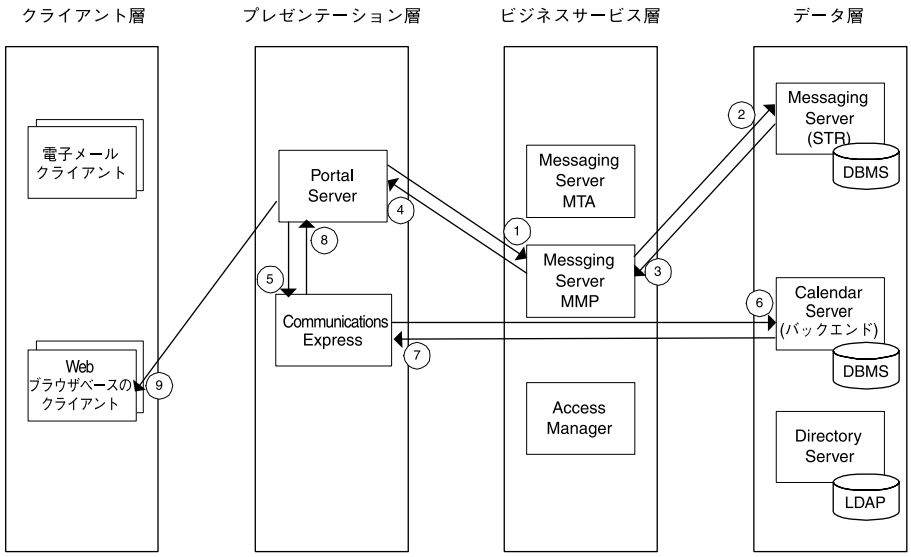
3. **Access Manager** は、ユーザー **ID** とパスワードの検証を **Directory Server** に要求する。
4. **Directory Server** は、ユーザー **ID** とパスワードを検証する。
5. **Access Manager** は、**Directory Server** にユーザープロファイルを要求する。
6. **Directory Server** は、ユーザープロファイルを返す。
7. **Portal Server** は、**Access Manager** にユーザー表示プロファイルを要求する。
8. **Access Manager** は、ポータル設定を返す。
9. ポータル設定が **Web** ブラウザクライアントに表示される。



▼ ユースケース 2: Portal Server が電子メール情報とカレンダ情報を表示する

手順
1. ログイン、認証、およびポータル設定の検索が正常に完了したあと、**Portal Server** は、**Messaging Server MMP** に電子メールメッセージを要求する。
2. **MMP** は、**Messaging Server STR** にメッセージリストを要求する。
3. **STR** は、**MMP** にメッセージリストを返す。
4. **MMP** は、**Portal Server** にメッセージヘッダーを転送する。

5. **Portal Server** は、**Communications Express** にカレンダ情報を要求する。
6. **Communications Express** は、**Calendar Server** バックエンドにカレンダ情報を要求する。
7. **Calendar Server** バックエンドは、**Communications Express** にカレンダ情報を返す。
8. **Communications Express** は、**Portal Server** にカレンダ情報を転送する。
9. **Portal Server** は、すべてのチャネル情報を **Web** ブラウザクライアントに送信する。



アクセスゾーン

論理アーキテクチャーのコンポーネントを表す別の方法は、アーキテクチャーでセキュリティー保護されたアクセスがどのように提供されるかを示すアクセスゾーンにコンポーネントを配置することです。次の図は、Java Enterprise System コンポーネントを配備するためのアクセスゾーンを示しています。各アクセスゾーンは、インターネットおよびイントラネットを経由するセキュリティー保護されたリモートアクセスをコンポーネントがどのように提供するかを示しています。

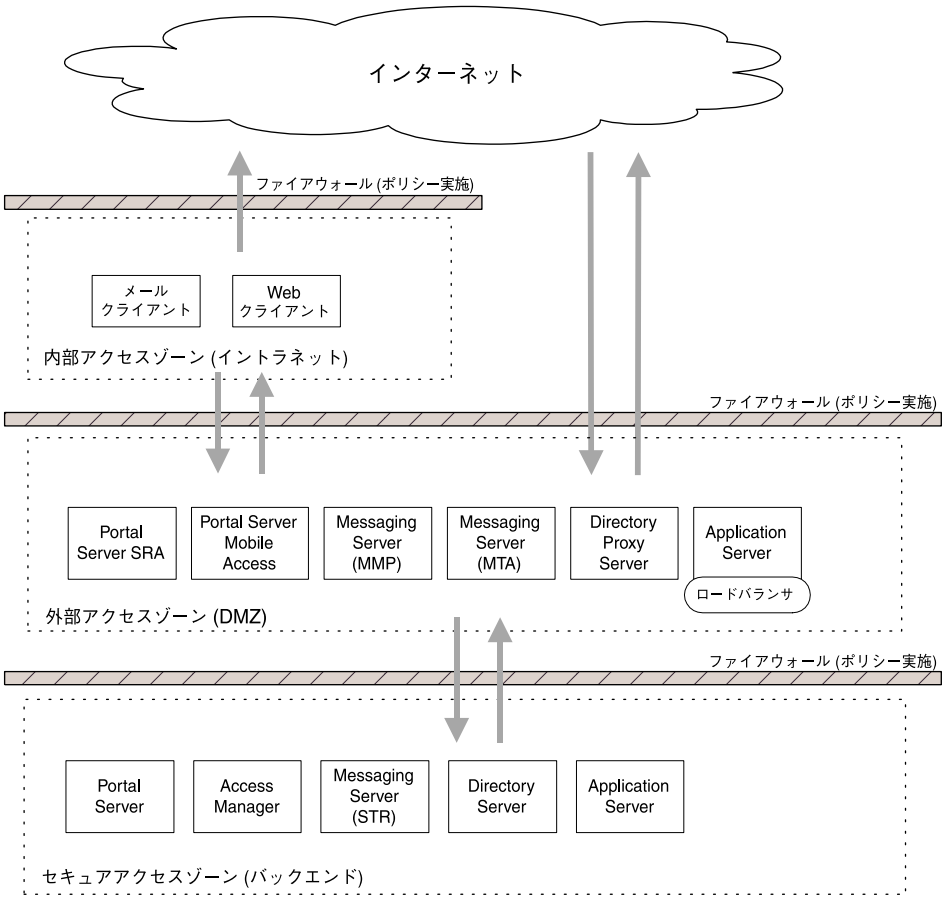


図 **4–5** アクセスゾーンに配置された論理コンポーネント

次の表は、65 ページの「アクセスゾーン」に示されているアクセスゾーンについて説明しています。

表 4–6 セキュアアクセスゾーンとその中に配置されたコンポーネント

アクセスゾーン	説明
内部アクセスゾーン (イントラネット)	イントラネットとインターネット間の、ファイアウォールによって実施されるポリシーが適用されるインターネットへのアクセス。内部アクセスゾーンは、通常、エンドユーザーが Web をブラウズしたり電子メールを送信したりするために使用します。 インターネットに直接アクセスして Web ブラウズすることが許可される場合もあります。しかし、通常は、インターネットに対するセキュリティー保護されたアクセスが、外部アクセスゾーンを通じて提供されます。
外部アクセスゾーン (DMZ)	重要なバックエンドサービスに対するセキュリティーバッファーとして機能し、インターネットに対するセキュリティー保護されたアクセスを提供します。
セキュアアクセスゾーン (バックエンド)	重要なバックエンドサービスに対する制限付きのアクセスを提供します。これらのサービスには、外部アクセスゾーンからのみアクセス可能です。

65 ページの「アクセスゾーン」には、前出の例で示されていた論理層は示されていませんが、代わりに、リモートアクセスおよび内部アクセスを提供するコンポーネント、これらのコンポーネントとファイアウォールなどのセキュリティー手段との関係、および実施する必要のあるアクセスルールの視覚的説明に焦点が当てられています。アクセスゾーンを示す図と組み合わせて多層アーキテクチャー設計を使用し、計画中の配備の論理モデルを提供します。

配備シナリオ

完成した論理アーキテクチャー設計だけでは、ソリューションライフサイクルの配備設計フェーズに進むには不十分です。論理アーキテクチャーと、技術要件フェーズで特定したサービス品質 (QoS) 要件を対応付ける必要があります。論理アーキテクチャーと QoS 要件の対応付けにより、配備シナリオが構成されます。第 5 章で説明するように、配備シナリオは、配備アーキテクチャーの設計の開始点となります。

第 5 章

配備設計

ソリューションライフサイクルの配備設計フェーズでは、上位レベルの配備アーキテクチャーと下位レベルの実装仕様を設計し、ソリューションを実装する上で必要な一連の計画と仕様を作成します。プロジェクトの承認は、配備設計フェーズで行われます。

この章で説明する内容は、次のとおりです。

配備設計について

配備設計は、ソリューションライフサイクルの論理設計フェーズと技術要件フェーズで作成された配備シナリオに基づいて行います。配備シナリオには、ソリューションの論理アーキテクチャー要件とサービス品質 (QoS) 要件が含まれています。論理アーキテクチャーで特定されている、物理サーバーおよびその他のネットワークデバイスにわたるコンポーネントをマッピングして、配備アーキテクチャーを作成します。QoS 要件により、パフォーマンス、可用性、スケーラビリティー、およびその他の関連 QoS 仕様に対応するハードウェア構成についてのガイドラインが提供されます。

配備アーキテクチャーの設計は、繰り返しのプロセスです。通常、QoS 要件を再確認し、予備設計を再検討します。得失評価および保有コスト問題のバランスをとりながら QoS 要件間の相互関連を考慮することで、最終的にプロジェクトのビジネス上の目的を達成する最適なソリューションに到達します。

プロジェクトの承認

プロジェクトの承認は、配備設計フェーズで行われます。 一般には、配備アーキテクチャーを作成したあとになります。配備アーキテクチャーと、場合によっては次に説明する実装仕様も使用して、配備の実際のコストを見積もり、利害関係者の承認を得るために提出します。プロジェクトが承認された場合は、配備の完了に関する契約に署名がなされ、プロジェクトの実装に必要なリソースの確保と割り当てが行われます。

配備設計のアウトプット

配備設計フェーズでは、次の仕様および計画のいずれかを作成します。

- 配備アーキテクチャー**:** 論理アーキテクチャーから物理環境へのマッピングを表す、上位のアーキテクチャー。物理環境には、イントラネットまたはインターネット環境にあるコンピュータノード、プロセッサ、メモリ、ストレージデバイスなどのハードウェアおよびネットワークデバイスが含まれます。
- 実装仕様**:** 配備を構築するための青写真として使用される詳細な仕様です。これらの仕様は、必要なコンピュータおよびネットワークハードウェアについての詳細を示すとともに、配備ネットワークの配置を説明するものです。実装仕様には、ディレクトリ情報ツリー (DIT: Directory Information Tree) の詳細やディレクトリアクセスに定義されているグループとロールなど、ディレクトリサービスに関する仕様も含まれます。
- 実装計画**:** 企業ソフトウェアソリューションの実装におけるさまざまな側面をカバーする計画の集合です。実装計画には、次の計画が含まれます。
  - 移行計画*:* 企業データの移行および企業ソフトウェアのアップグレードを行うための戦略とプロセスを説明します。移行したデータは、新しくインストールした企業アプリケーションの形式と標準に適合する必要があります。相互運用するには、すべての企業ソフトウェアは正しいリリースバージョンレベルにある必要があります。
  - インストール計画*:* 配備アーキテクチャーから派生します。 分散型の配備をインストールおよび設定するために必要なハードウェアサーバー名、インストールディレクトリ、インストール順序、各ノードのインストールタイプ、および設定情報を指定します。
  - ユーザー管理計画*:* 既存のディレクトリおよびデータベース内のデータの移行戦略、配備アーキテクチャーで指定されているレプリケーション設計を考慮に入れたディレクトリ設計仕様、およびディレクトリに新しいコンテンツをプロビジョニングする手順が含まれます。

- テスト計画: 配備されたソフトウェアをテストする手順を説明します。 プロトタイプの作成とパイロット実装の具体的な計画、想定される負荷を処理する能力を判定するストレステスト、および計画した機能が期待どおり動作するかどうかを判定する機能テストが含まれます。
- ロールアウト計画: 実装を計画およびテスト環境から本稼働環境に移行するための手順とスケジュールを説明します。本稼働環境への実装の移行は、通常、さまざまなフェーズで行われます。たとえば、最初のフェーズでは、限られたユーザーグループ用にソフトウェアを配備し、配備全体が完了するまで、各フェーズでユーザーベースを拡大していきます。段階的な実装には、配備全体が完了するまでの、特定のソフトウェアパッケージの計画的な実装も含まれることがあります。
- 障害回復計画: 予期しないシステム全体の障害からシステムを回復させる手順を説明します。回復計画には、大規模な障害と小規模な障害の両方の場合の手順が含まれます。
- 運用計画 (運用マニュアル): 監視、保守、インストール、およびアップグレードの手順を説明する運用マニュアルです。
- トレーニング計画: 新しくインストールされた企業のソフトウェアに関して、オペレータ、管理者、およびエンドユーザーをトレーニングするためのプロセスと手順を説明します。

配備設計に影響する要因

配備設計における決定にいくつかの要因が影響します。次の主要な要因を考慮します。

- 論理アーキテクチャー: 論理アーキテクチャーでは、提案ソリューションの機能サービスおよびそれらのサービスを提供するコンポーネント間の相互関係が詳しく説明されています。論理アーキテクチャーを、サービスを提供する最善の方法を決定するための鍵として使用します。配備シナリオには、次に説明するサービス品質要件と対応する論理アーキテクチャーが含まれています。
- サービス品質要件: サービス品質 (QoS) 要件には、ソリューションの運用に関するさまざまな条件が指定されています。QoS 要件を参照して、パフォーマンス、可用性、スケーラビリティー、保守性などのサービス品質上の目標を達成する戦略を策定します。配備シナリオには、前述の論理アーキテクチャーをサービス品質要件と対応付けて記述します。
- 使用パターンの分析: ソリューションライフサイクルの技術要件フェーズで作成された使用パターンの分析結果は、配備されたシステムに対する負荷とストレスを見積もる上で役立つ情報を提供します。使用パターンの分析結果を使用して、パフォーマンス障害を特定するとともに、QoS 要件を満たす戦略を策定します。
- ユースケース: ソリューションライフサイクルの技術要件フェーズで作成されたユースケースには、配備に対するものとして特定された、特徴のあるユーザーインタラクションが示されています。多くの場合は、最も一般的なユースケースが特定されています。ユースケースは使用パターン分析に組み込まれていますが、配備設計を評価する際は、ユースケースを参照して、それらが適切に取り上げられていることを確認する必要があります。

- サービスレベル契約**:** サービスレベル契約 (SLA) には、最小パフォーマンス要件およびそれらの要件が満たされない場合に提供する必要のあるカスタマーサポートのレベルと範囲が規定されています。配備設計は、サービスレベル契約で規定されているパフォーマンス要件を十分に満たすものである必要があります。
- 総保有コスト**:** 配備設計では、可用性、パフォーマンス、スケーラビリティーなどの QoS 要件に対応できるソリューションを検討する必要があります。ただし、検討するソリューションごとに、そのソリューションのコストおよびそのコストが総保有コストに及ぼす影響についても考慮する必要があります。決定によって具体化する得失評価を考慮してください。 また、ビジネス制約内のビジネス要件を達成するために、リソースを最適化したことを確認してください。
- ビジネス上の目的**:** ビジネス上の目的は、ソリューションライフサイクルのビジネス分析フェーズで明確にされるもので、それらの目的を達成する上でのビジネス要件とビジネス制約が含まれています。配備設計は、ビジネス上の目的をどれだけ達成できるかによって最終的に評価されます。

配備の設計方法

配備計画のほかの作業と同様に、配備設計は技術であるとともに技法でもあるため、具体的な手順とプロセスを詳細に説明することは不可能です。配備設計の成功に貢献する要因は、過去の設計経験、システムアーキテクチャーに関する知識、ビジネス領域に関する知識、およびクリエイティブな思考の応用です。

通常、配備設計は、パフォーマンス要件の達成を中心に展開し、その他の QoS 要件も達成します。採用する戦略は、設計上の意思決定による得失評価のバランスをとることでソリューションを最適化するものである必要があります。一般的に採用される方法には、次の作業が伴います。

- プロセッサ要件の見積もり**:** 多くの場合、配備設計は、論理アーキテクチャーの各コンポーネントに必要な CPU の見積もりに基づきます。最大の負荷を示しているユースケースから始めて、各ユースケースに進みます。そのユースケースをサポートしているすべてのコンポーネントの負荷を考慮し、それに応じて見積もりを修正します。また、企業システムの設計に関するこれまでの経験も参考にします。
- セキュリティー保護された転送のためのプロセッサ要件の見積もり**:** セキュリティー保護された転送を必要とするユースケースを分析して、それに応じて CPU の見積もりを修正します。
- 可用性とスケーラビリティーのためのサービスのレプリケーション**:** プロセッサの見積もりを十分に行なったあと、可用性とスケーラビリティー の QoS 要件に対応するために、設計を修正します。可用性およびフェイルオーバーの考慮事項に対処するロードバランスソリューションについて考慮します。

  分析時に、設計上の意思決定による得失評価を考慮します。たとえば、可用性とスケーラビリティーの戦略がシステムの保守性 (保守) に与える影響です。また、戦略のその他のコストには何があるかなども考慮できます。

- 障害の特定**:** 分析の過程で、配備設計を調べて、データの伝送が要件を下回る原因となる障害を特定し、調整を行います。
- リソースの最適化**:** 配備設計をリソース管理について確認し、要件を満たしながらコストを最小化する選択肢を検討します。
- リスクの管理**:** ビジネス分析および技術分析の設計面を見直して、計画の早期時点で予測されていなかった可能性のあるイベントや状況に対応できるように修正を行います。

プロセッサ要件の見積もり

ここでは、配備設計で、サービスをサポートするために必要な CPU プロセッサ数および対応するメモリを見積もるプロセスについて説明します。また、通信配備シナリオの例を使用して、見積もりプロセスを段階的に説明します。

CPU 処理能力の見積もりは、次のことを考慮して繰り返し検討します。

- 論理アーキテクチャーでコンポーネントの依存関係によって示される、論理コンポーネントとそれらのインタラクション
- 特定されたユースケースの使用パターン分析
- サービス品質要件
- 配備設計と Java Enterprise System に関する過去の経験
- さまざまな種類の配備シナリオの設計と実装の経験を持つ、Sun プロフェッショナルサービスとの相談

見積もりプロセスには、次の手順が含まれます。これらの手順の順序は、あまり重要ではありませんが、最終結果に影響する要因を分析する 1 つの方法を提供するものです。

1. システムへのユーザーエントリポイントとなるコンポーネントの、基準となる CPU の見積もりを決定します。

   設計上の意思決定の 1 つは、CPU をフルに搭載するか、一部だけ搭載するかについてです。CPU をフルに搭載すると、システムの処理能力は最大化されます。処理能力を向上するために、保守コストおよび場合によっては CPU の追加による停止時間が発生します。増大するパフォーマンス要件を満たすために、マシンの追加を選択できる場合もあります。

   CPU を一部だけ搭載した場合は、保守コストをただちに発生させることなく、パフォーマンス要件に対応できる余裕が持てます。ただし、活用し切れないシステムへの先行投資費用が発生します。
2. コンポーネント間のインタラクションを考慮して、CPU の見積もりを調整します。

   論理アーキテクチャーのコンポーネント間のインタラクションを調べて、依存コンポーネントのためにさらなる搭載が必要であるかを決定します。

3. 使用パターン分析で特定のユースケースを調べてシステムのピーク負荷を割り出し、ピーク負荷を処理するコンポーネントに対する調整を行います。

   最も重み付けされた (最も負荷を必要とする) ユースケースから始めて、各ユースケースに進み、計画したすべての使用シナリオを考慮したことを確認します。

4. CPU の見積もりを調整して、セキュリティー、可用性、スケーラビリティーの各要件を反映します。

この見積もりプロセスは、実際に必要な処理能力を決定するための開始点となります。通常は、これらの見積もりに基づいてプロトタイプ配備を作成し、その後、想定されるユースケースに対して厳格なテストを実行します。テストを繰り返したあとに初めて、配備設計の実際の処理要件を決定できます。

プロセッサ要件の見積もりの例

ここでは、配備例に必要な処理能力を見積もる 1 つの方法を示します。配備例は、1,000 ～ 5,000 人の従業員を持つ中規模企業用のアイデンティティーベースによる通信ソリューションの論理アーキテクチャーに基づいています。62 ページの「アイデンティティーベースの通信の例」の節を参照してください。

この例で使用される CPU とメモリの値は、図で使用するための任意の見積もり結果に過ぎません。これらの値は、理論上の例が基になっている任意のデータから計算されています。プロセッサ要件を見積もるには、さまざまな要因を徹底的に分析する必要があります。この分析には次の情報が含まれているものとしますが、これらに限定されません。

- 徹底的なビジネス分析に基づいた詳細なユースケースと使用パターン分析
- ビジネス要件の分析によって決定されたサービス品質要件
- 処理およびネットワークハードウェアの具体的なコストと仕様
- 類似の配備を実装した過去の経験

注意 – これらの例に示される情報は、何らかの具体的な実装の提案を示すものではなく、システムの設計プロセスの説明を目的に記載されています。

ユーザーエントリポイント用の基準 CPU の見積もり

ユーザーエントリポイントとなる各コンポーネントの想定負荷を処理するために必要な CPU の数を見積もることから開始します。次の図は、62 ページの「アイデンティティーベースの通信の例」で説明したアイデンティティーベースの通信シナリオの論理アーキテクチャーを示しています。

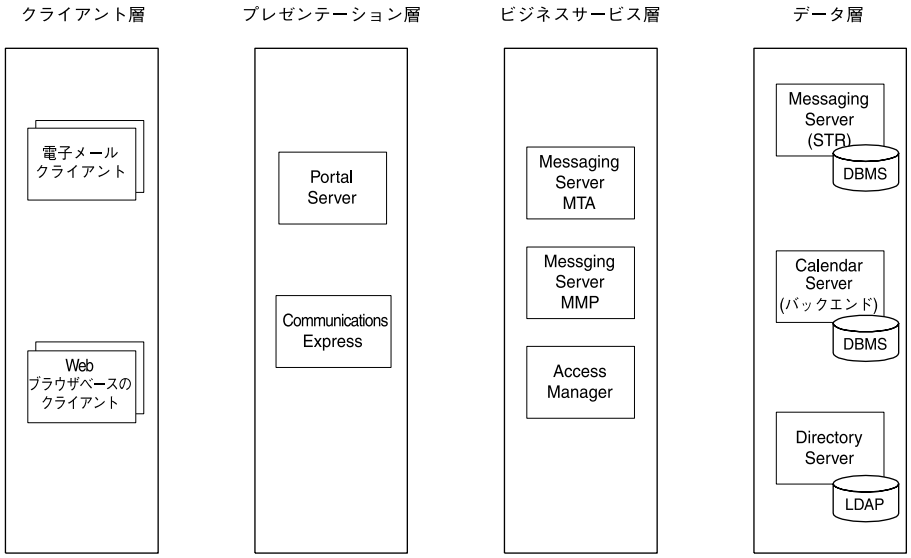


図 **5–1** アイデンティティーベースの通信シナリオの論理アーキテクチャー

次の表は、配備のエンドユーザーと直接インタフェースをとる、論理アーキテクチャーのプレゼンテーション層のコンポーネントを示しています。この表には、技術要件の分析から得られた基準 CPU の見積もり、ユースケース、具体的な使用パターン分析、および同種の配備に関する過去の経験が反映されています。

表 **5–1** アクセスユーザーのエントリポイントを含むコンポーネントの CPU の見積もり

コンポーネント	**CPU** の数	説明
Portal Server	4	ユーザーエントリポイントであるコンポーネント。
Communications Express	2	Portal Server のメッセージングチャネルおよびカレンダチャネルにデータを配信します。

サービス依存関係に関する CPU 見積もり値の取り込み

ユーザーエントリポイントとなるコンポーネントは、他の Java Enterprise System コンポーネントからのサポートを必要とします。パフォーマンス要件の特定を継続するには、必要となる他のコンポーネントからのサポートを考慮して、パフォーマンスの見積もり値を組み込みます。コンポーネント間のインタラクションの種類は、論理アーキテクチャーを設計する際に詳細化します。57 ページの「論理アーキテクチャーの例」の節で、論理アーキテクチャーの例を参照してください。

表 **5–2** サポートするコンポーネントのための CPU の見積もり

コンポーネント	**CPU** の数	説明
Messaging Server MTA (着信)	1	Communications Express および電子メールクライアントからの着信メールメッセージをルーティングします。
Messaging Server MTA (発信)	1	発信メールメッセージを受取人に配信します。
Messaging Server MMP	1	電子メールクライアントの Messaging Server メッセージストアにアクセスします。
Messaging Server STR (メッセージストア)	1	電子メールメッセージを取得および保存します。
Access Manager	2	認証と承認のサービスを提供します。
Calendar Server (バックエンド)	2	Communications Express、Calendar Server フロントエンドのカレンダデータを取得および保存します。
Directory Server	2	LDAP ディレクトリサービスを提供します。
Web Server	0	Portal Server および Access Manager に対して Web コンテナサポートを提供します。 追加の CPU サイクルは不要です。

ユースケースによるピーク負荷の調査

ユースケースおよび使用パターン分析に戻り、ピーク負荷の領域を特定し、CPU の見積もりを調整します。

たとえば、この例に関して次のようなピーク負荷状況を特定したとします。

- ユーザーが同時にログインすることによる、初期のユーザー数の増加
- 特定の時間枠内での電子メールの交換

このピーク負荷時の使用パターンを考慮に入れるには、これらのサービスを提供するコンポーネントに対する調整を行います。次の表は、このピーク負荷時の使用パターンを考慮した調整の概要を示しています。

表 **5–3** ピーク負荷による CPU 見積もりの調整

コンポーネント	**CPU** の数 **(調整後)**	説明
Messaging Server MTA 着信	2	ピーク時の着信電子メール用に 1 CPU を追加します
Messaging Server MTA 発信	2	ピーク時の送信電子メール用に 1 CPU を追加します

表 **5–3** ピーク負荷による CPU 見積もりの調整　(続き)

コンポーネント	**CPU** の数 **(**調整後**)**	説明
Messaging Server MMP	2	増加する負荷用に 1 CPU を追加します
Messaging Server STR (メッセージストア)	2	増加する負荷用に 1 CPU を追加します
Directory Server	3	増加する LDAP 検索用に 1 CPU を追加します

その他の負荷条件による見積もりの修正

CPU の見積もりを続行し、負荷に影響を与える可能性のあるその他のサービス品質要件を考慮に入れます。

- セキュリティー**:** 技術要件フェーズで、セキュリティー保護されたデータ転送が負荷要件に与える影響を特定し、それに応じて見積もりを修正します。後述の、78 ページの「セキュリティー保護されたトランザクションのためのプロセッサ要件の見積もり」の節では、調整を行うプロセスについて説明しています。
- サービスのレプリケーション**:** 可用性、ロードバランス、およびスケーラビリティーのためのサービスのレプリケーションを考慮して、CPU の見積もりを調整します。後述の、81 ページの「可用性戦略の決定」の節では、可用性ソリューションのためのサイズ設定について説明します。89 ページの「スケーラビリティーのための戦略の決定」の節では、ディレクトリサービスへのアクセスによるソリューションについて説明します。
- 潜在処理能力およびスケーラビリティー**:** 配備上での予期しない大きな負荷に対処するために必要な潜在処理能力に応じて、CPU の見積もりを修正します。規模拡大の段階的な達成目標および時間の経過とともに想定される負荷の増加を考慮し、水平方向または垂直方向にシステムを拡張する目標を達成できることを確認します。

CPU 見積もりの更新

通常は、CPU の数を偶数値に切り上げます。偶数値に切り上げることで、2 つの物理サーバーに CPU 見積もりを均等に分割できるとともに、潜在処理能力の小さな要因が追加されます。ただし、サービスのレプリケーションに対する特定のニーズに基づいて切り上げる必要があります。

原則として、CPU ごとに 2G バイトのメモリを許容します。必要な実際のメモリは、特定の使用パターンによって異なります。 これは、テストで決定できます。

次の表は、アイデンティティーベースの通信に関する最終見積もりの例です。これらの見積りには、セキュリティーおよび可用性のために追加される可能性のある、追加の処理能力は含まれていません。セキュリティーおよび可用性のための各合計は、この後の節で加えられます。

表 **5–4** サポートするコンポーネントのための CPU 見積もりの調整

コンポーネント	**CPU** の数	メモリ
Portal Server	4	8G バイト
Communications Express	2	4G バイト
Messaging Server (MTA、着信)	2	4G バイト
Messaging Server (MTA、発信)	2	4G バイト
Messaging Server (MMP)	2	4G バイト
Messaging Server (メッセージストア)	2	4G バイト
Access Manager	2	4G バイト
Calendar Server	2	4G バイト
Directory Server	4	8G バイト (3 CPU/6G バイトメモリから切り上げ)
Web Server	0	0

セキュリティー保護されたトランザクションのためのプロセッサ要件の見積もり

セキュリティー保護されたデータ転送には、SSL (Secure Seckets Layer) や TLS (Transport Layer Security) など、セキュリティー保護された転送プロトコルを通じたトランザクションの処理が伴います。多くの場合、セキュリティー保護された転送を通じて処理されるトランザクションは、セキュリティー保護されたセッションの確立 (ハンドシェークと呼ばれる) および転送データの暗号化と復号化のために、追加の処理能力を必要とします。使用する暗号化アルゴリズム (たとえば、40 ビットまたは 128 ビットの暗号化アルゴリズム、など) によっては、かなりの追加処理能力が必要になることもあります。

セキュリティー保護されたトランザクションを、保護されていないトランザクションと同一レベルで実行するには、追加の処理能力を計画する必要があります。トランザクションおよびそれを処理する Sun Java™ Enterprise System サービスの性質によっては、セキュリティー保護されたトランザクションの処理に、保護されていないトランザクションの場合の 4 倍の処理能力が必要になることもあります。

セキュリティー保護されたトランザクションを処理するための処理能力を見積もるには、ユースケースを分析し、セキュリティー保護された転送を必要とするトランザクションの割合を求めます。セキュリティー保護されたトランザクションと保護されないトランザクションのパフォーマンス要件が同じであれば、セキュリティー保護されたトランザクションに必要な追加の処理能力を考慮して CPU の見積もりを調整します。

場合によっては、セキュリティー保護された転送が認証時にだけ必要になることもあります。システムに対するユーザーの認証が完了すれば、データの転送に追加のセキュリティー対策は必要なくなります。また、場合によっては、セキュリティー保護された転送がすべてのトランザクションで求められることもあります。

たとえば、オンラインの電子商取引サイトで製品カタログを参照する場合、顧客が商品の選択を終え、購入のための支払い手続きを開始するまでは、すべてのトランザクションはセキュリティー保護の必要がありません。しかし、銀行や仲介取引業向けの配備では、ほとんど、またはすべてのトランザクションをセキュリティー保護し、セキュリティー保護されたトランザクションと、保護されていないトランザクションの両方に同じパフォーマンス標準を適用する必要があります。

セキュリティー保護されたトランザクションのための CPU の見積もり

ここでは、引き続き同じ配備例を使用して、セキュリティー保護されたトランザクションと保護されないトランザクションの両方を含む理論上のユースケースに基づいて CPU 要件を計算する方法について説明します。

セキュリティー保護されたトランザクションの CPU 要件を見積もるには、次の計算を行います。

1. 前節の、74 ページの「プロセッサ要件の見積もりの例」に示されている CPU 見積もりの基準値から開始します。
2. セキュリティー保護された転送を必要とするトランザクションの割合を計算し、セキュリティー保護されたトランザクションの CPU 見積もりを計算します。
3. セキュリティー保護されないトランザクションによって減少する分の CPU 見積もりを計算します。
4. セキュリティー保護されたトランザクションと保護されないトランザクションの見積もりを集計し、合計 CPU 見積もりを計算します。
5. 合計 CPU 見積もりを偶数値に切り上げます。

79 ページの「セキュリティー保護されたトランザクションのための CPU の見積もり」は、次を前提とした Portal Server の場合のユースケースおよび使用パターン分析に基づいた計算例を示しています。

- すべてのログインは、セキュリティー保護された認証を必要とする。
- すべてのログインは、Portal Server の負荷全体の 10% に相当する。
- セキュリティー保護されたトランザクションのパフォーマンス要件は、セキュリティー保護されないトランザクションのパフォーマンス要件と同じである。

セキュリティー保護されたトランザクションの処理によって必要となる追加処理能力を考慮するには、これらのトランザクションを処理するための CPU の数を 4 倍に増やします。この例のその他の CPU の見積もり値と同様に、これは説明用の任意の値に過ぎません。

表 **5–5** セキュリティー保護されたトランザクションのための CPU の見積もりの修正

手順	説明	計算	結果
1	すべての Portal Server トランザクションの基準見積もりから始めます。	76 ページの「ユースケースによるピーク負荷の調査」から、基準見積もり値は 4 CPU です。	-----

2	セキュリティー保護されるトランザクションの追加の CPU 見積もりを計算します。セキュリティー保護されたトランザクションには、保護されないトランザクションの 4 倍の CPU 能力が必要であると仮定します。	基準見積もり値の 10% がセキュリティー保護された転送を必要とします。
		`0.10 x 4 CPU = 0.4 CPU`
		4 倍して、セキュリティー保護されたトランザクションのための CPU を増やします。
		`4 x 0.4 = 1.6 CPU`
			1.6 CPU

3	セキュリティー保護されないトランザクションによって減少する分の CPU 見積もりを計算します。	基準見積もり値の 90% が、セキュリティー保護されない転送です。 `0.9 x 4 CPU = 3.6 CPU`	3.6 CPU
4	セキュリティー保護されるトランザクションと保護されないトランザクションの調整後の合計 CPU 見積もりを計算します。	セキュリティー保護される場合の見積もり + セキュリティー保護されない場合の見積もり = 合計 `1.6 CPU + 3.6 CPU = 5.2 CPU`	5.2 CPU
5	偶数値に切り上げます。	`5.2 CPU ==> 6 CPU`	6 CPU

この例でのセキュリティー保護されるトランザクションのための計算に基づいて、追加の 2 CPU と 4G バイトのメモリを加算して 79 ページの「セキュリティー保護されたトランザクションのための CPU の見積もり」の CPU 見積もりの合計を修正すると、Portal Server について次の合計が得られます。

コンポーネント	**CPU** の数	メモリ
Portal Server	6	12G バイト

SSL トランザクション処理の専用ハードウェア

SSL アクセラレータカードなどの特殊なハードウェアデバイスを使用して、セキュリティー保護されたセッションの確立およびデータの暗号化と復号化を処理するための追加の処理能力を供給することができます。SSL 操作に特化されたハードウェアを使用した場合、その処理能力は一部の特定の SSL 処理専用に使用され、通常は、セキュリティー保護されたセッションを確立するための「ハンドシェーク」処理だけに使用されます。

このハードウェアは、配備の最終的なアーキテクチャーにメリットをもたらす可能性があります。しかし、これは特殊なハードウェアであるため、まず、セキュリティー保護されたトランザクションのパフォーマンス要件を CPU の処理能力単位で見積もり、そのあとで追加負荷を処理する専用ハードウェアの導入メリットを検討してください。

専用ハードウェアの導入を検討するときは、ユースケースがそのハードウェアの利用をサポートするかどうか (たとえば、多数の SSL ハンドシェーク処理を必要とするユースケースなど)、また、このようなハードウェアによって設計にもたらされる追加処理層の複雑さを考慮します。この複雑さには、これらのデバイスのインストール、設定、テスト、管理が含まれます。

可用性戦略の決定

可用性要件のための戦略を策定する際は、コンポーネントのインタラクションと使用パターンの分析に基づいて、検討する可用性ソリューションを決定します。コンポーネントごとに分析を行い、可用性とフェイルオーバーの要件に最も適したソリューションを決定します。

次に、可用性戦略の決定に役立つ情報の例を示します。

- 可用性の「ナイン」はいくつ必要か。
- フェイルオーバー状況に関するパフォーマンス仕様は何か (たとえば、フェイルオーバー中も 50% 以上のパフォーマンス、など)。
- 使用パターンの分析によって、ピーク時と非ピーク時は特定されるか。
- 地理的に考慮すべき事項は何か。

可用性戦略は、93 ページの「リソースの最適な使用方法の設計」で説明する保守性要件も考慮した上で選択する必要があります。膨大な管理や保守を必要とする複雑なソリューションは避けます。

可用性戦略

Java Enterprise System 配備の可用性戦略には、次のものがあります。

- ロードバランス: 冗長ハードウェアコンポーネントおよびソフトウェアコンポーネントを使用して、処理による負荷を分散します。ロードバランサは、サービスに対するあらゆる要求を、そのサービスの複数の対称インスタンスのいずれかに送信します。インスタンスの 1 つが失敗した場合は、ほかのインスタンスにその負荷を負わせることができます。
- フェイルオーバー: コンポーネントで障害が発生した場合、サービスの継続的なアクセスと重要なデータのセキュリティーを提供するためには、冗長ハードウェアおよびソフトウェアの管理が必要とされます。

  Sun Cluster ソフトウェアは、Messaging Server のメッセージストレージや Calendar Server のカレンダデータなど、バックエンドコンポーネントによって管理される重要なデータのためのフェイルオーバーソリューションを提供します。
- サービスのレプリケーション: サービスのレプリケーションにより、同じデータへのアクセス用に複数のソースが提供されます。Directory Server は、LDAP ディレクトリアクセス用に、多数のレプリケーション戦略および同期戦略を提供します。

以降の各節では、さまざまなレベルのロードバランス、フェイルオーバー、およびサービスのレプリケーションを提供する可用性ソリューションの例をいくつか示します。

シングルサーバーシステム

サービスのすべてのコンピュータリソースを 1 つのサーバーに配置します。サーバーに障害が発生した場合、サービス全体が失敗します。

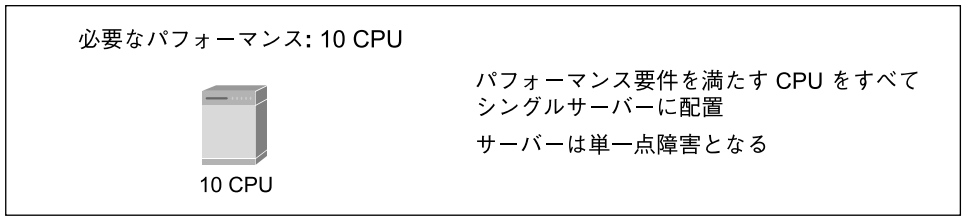


図 **5–2** シングルサーバーシステム

Sun は、次の利点を提供するハイエンドサーバーを提供しています。

- システムが稼働した状態でのハードウェアコンポーネントの交換と再設定
- サーバー上の、障害から分離されたドメインで複数アプリケーションを稼働する機能
- システムの再起動を必要とせずに、処理能力、パフォーマンス速度、および I/O 設定をアップグレードする機能

通常、ハイエンドサーバーは、同等の機能を持つマルチサーバーシステムより高価になります。ただしシングルサーバーでは、管理、監視、データセンターでのサーバーのホスティング費用が軽減されます。ロードバランス、フェイルオーバー、障害シングルポイントの除去は、マルチサーバーシステムのほうが柔軟に対応できます。

水平方向の冗長システム

ロードバランスとフェイルオーバーの両方を提供する平行冗長サーバーを使用することで、いくつかの方法で可用性を向上できます。次の図は、N+1 フェイルオーバーシステムを構成する 2 つのレプリカサーバーを示しています。N+1 システムには、1 つのサーバーで障害が発生した場合に、その処理能力の 100% を提供するための追加サーバーが含まれます。

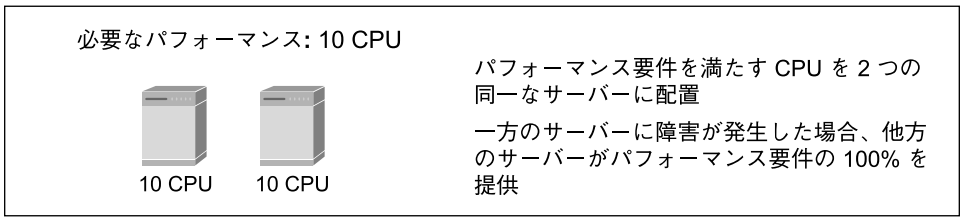


図 **5–3** 2 つのサーバーで構成される N+1 フェイルオーバーシステム

上の83 ページの「水平方向の冗長システム」に示される各サーバーは、同一処理能力を持ちます。1 つのサーバーが単独でパフォーマンス要件を満たします。もう一方のサーバーは、バックアップとしてサービスに適用された場合に 100% のパフォーマンスを提供します。

N+1 フェイルオーバー設計の利点は、フェイルオーバー状況でも 100% の処理能力を提供できることです。欠点には、1 つのサーバーはフェイルオーバー状況でだけ使用されるスタンバイであるため、パフォーマンス全体の向上なしにハードウェアコストが上昇することが挙げられます。

次の図は、2 つのサーバーの間でパフォーマンスを分散するロードバランスとフェイルオーバーを実装するシステムを示しています。

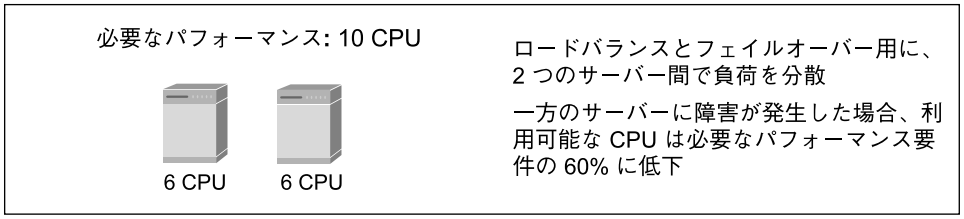


図 **5–4** 2 つのサーバー間のロードバランスとフェイルオーバー

上の83 ページの「水平方向の冗長システム」で示されているシステムでは、一方のサーバーに障害が発生した場合に、処理能力は低下しますが、すべてのサービスを利用できます。残ったサーバーは 6 CPU の処理能力を提供します。 これは、10 CPU という要件の 60% に相当します。

この設計の利点は、両方のサーバーを利用可能な状態では、2 CPU の潜在処理能力を追加されることです。

次の図は、パフォーマンス向上とロードバランス用に、多数のサーバーの間でパフォーマンスを分散する例を示しています。

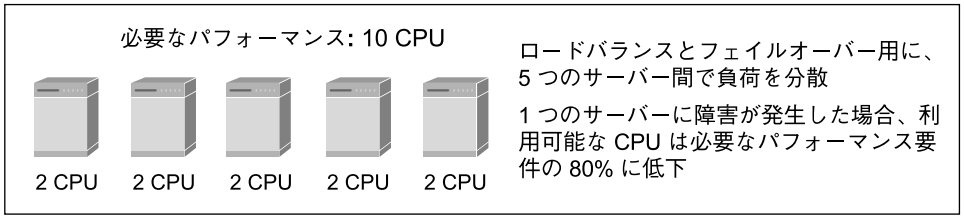


図 **5–5** *n* サーバー間での負荷の分散

83 ページの「水平方向の冗長システム」に示される設計では 5 つのサーバーが使用されるため、1 つのサーバーに障害が発生した場合に、残りのサーバーが 8 CPU の処理能力を提供します。 これは、10 CPU のパフォーマンス要件の 80% に相当します。この設計に 2 CPU の処理能力を持つサーバーを追加すると、効率的に N+1 設計を実現できます。1 つのサーバーに障害が発生しても、残りのサーバーによってパフォーマンス要件は 100% 満たされます。

この設計には、次のような利点があります。

- 1 つのサーバーに障害が発生した場合のパフォーマンスの追加
- 複数のサーバーがダウンした場合の可用性
- 保守とアップグレードのためにサーバーを順に停止できる
- 複数のローエンドサーバーは、通常は 1 つのハイエンドサーバーより安い

ただし、サーバーを追加することで、管理と保守のコストは大きく上昇します。データセンターでのサーバーのホスティング費用も考慮する必要があります。サーバーの追加投入によるメリットは、ある時点から減少に転じます。

Sun Cluster ソフトウェア

高い可用性 (4 ナインまたは 5 ナインなど) が必要となる状況では、可用性設計の一部として Sun Cluster ソフトウェアの利用を検討することができます。クラスタシステムは、冗長サーバーとストレージやその他のネットワークリソースを結合したものです。クラスタ内のサーバーは、継続的に相互通信します。1 つのサーバーがダウンした場合、クラスタ内の残りのデバイスはそのサーバーを分離し、障害のあるノードから別のノードにアプリケーションまたはデータをフェイルオーバーします。このフェイルオーバープロセスは、比較的迅速に行われるため、システムの利用者がサービスの中断を感じることはほとんどありません。

Sun Cluster ソフトウェアは、専用ハードウェアの追加と、設定、管理、維持のための特別なスキルを必要とします。

可用性設計の例

ここでは、62 ページの「アイデンティティーベースの通信の例」で説明されている、1,000 ～ 5,000 人の従業員を持つ中規模企業用の、アイデンティティーベースの通信ソリューションに基づいた、可用性戦略の 2 つの例を示します。1 つ目の可用性戦略では、Messaging Server のロードバランスについて説明します。2 つ目では、Sun Cluster ソフトウェアを使用するフェイルオーバーソリューションについて説明します。

Messaging Server のロードバランスの例

次の表は、論理アーキテクチャー内の論理 Messaging Server コンポーネントの CPU 能力の見積もり値を示しています。この表は、77 ページの「CPU 見積もりの更新」の節で計算した最終見積もり値を再掲しています。

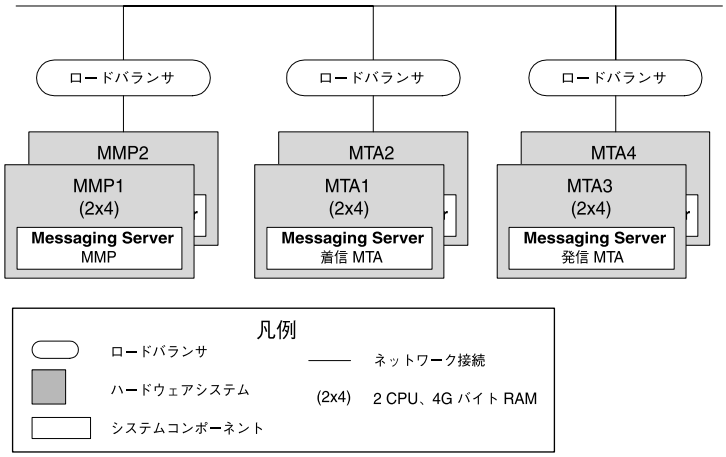
表 **5–6** サポートするコンポーネントのための CPU 見積もりの調整

コンポーネント	**CPU** の数	メモリ
Messaging Server (MTA、着信)	2	4G バイト
Messaging Server (MTA、発信)	2	4G バイト
Messaging Server (MMP)	2	4G バイト
Messaging Server (メッセージストア)	2	4G バイト

たとえば、技術要件フェーズで、次のサービス品質要件が指定されたとします。

- 可用性**:** システム全体の可用性は 99.99% (予定された停止時間は含めない) とする。個々のコンピュータシステムの障害によってサービス障害が発生してはならない。
- スケーラビリティー**:** 日々のピーク負荷時において、どのサーバーも 80% を超えて利用されてはならない。 また、システムは 1 年当たり 10% の長期成長に対応する必要がある。

可用性要件を満たすには、各 Messaging Server コンポーネントについてインスタンスを 2 つずつ、それぞれを異なるハードウェアサーバーに提供します。一方のコンポーネントのサーバーに障害が発生した場合、もう一方がサービスを提供します。次の図は、この可用性戦略のネットワーク図です。



上の図では、CPU の数は元の見積もりの 2 倍になっています。CPU の数は、次の理由から 2 倍になりました。

- 一方のサーバーに障害が発生した場合、残りのサーバーが CPU 能力を提供して負荷を処理するため。
- ピーク負荷時にどのサーバーも 80% を超えて利用されてはならないとするというスケーラビリティー要件に基づき、追加分の CPU 能力でこの安全マージンを提供するため。
- 1 年当たり 10% の負荷増大に対応することというスケーラビリティー要件に基づき、追加分の CPU 能力により、さらなる拡張が必要とされるまで増大する負荷を処理できる潜在処理能力を高めるため。

Sun Cluster ソフトウェアを使用したフェイルオーバーの例

次の図は、Calendar Server バックエンドおよび Messaging Server メッセージストアのフェイルオーバー戦略の例を示しています。Calendar Server バックエンドおよびメッセージストアは、別個のハードウェアサーバーにレプリケートされ、Sun Cluster ソフトウェアを使用してフェイルオーバー用に設定されます。CPU の数および対応するメモリは、Sun Cluster で各サーバー上にレプリケートされます。

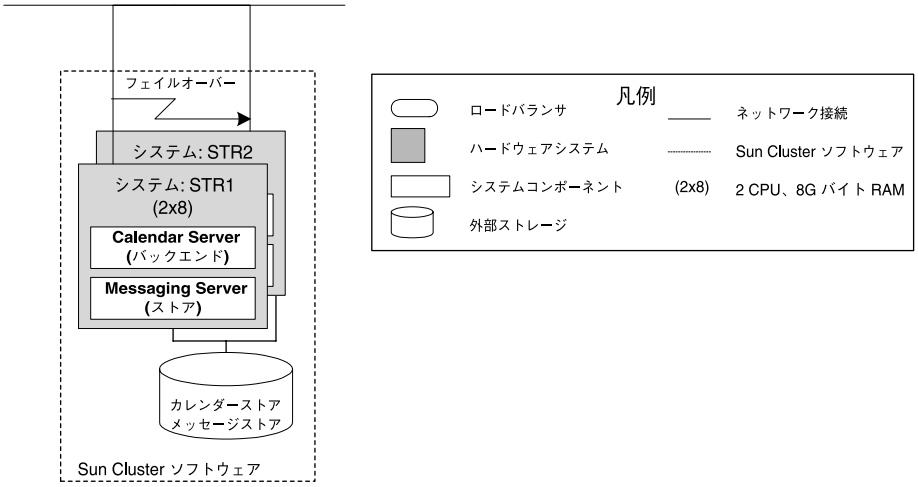


図 **5–6** Sun Cluster ソフトウェアを使用したフェイルオーバー設計

ディレクトリサービスのレプリケーションの例

ディレクトリサービスをレプリケートして複数のサーバーにトランザクションを分散すると、高い可用性が得られます。Directory Server により、サービスのレプリケーションためのさまざまな戦略が提供されます。これには、次の戦略が含まれます。

- 複数データベース**:** 別個のデータベースに、ディレクトリツリーのさまざまな部分を保存します。
- 連鎖およびリフェラル**:** 分散されたデータを 1 つのディレクトリツリーにリンクします。
- シングルマスターレプリケーション**:** マスターデータベース用の中央ソースを提供します。これはその後、コンシューマレプリカに分散されます。
- マルチマスターレプリケーション**:** 複数のサーバーにマスターデータベースを分散します。これらのマスターはその後、それぞれのデータベースをコンシューマレプリカに分散します。

Directory Server の可用性戦略は複雑な事項であり、このマニュアルの範囲に含まれません。以降の各節、87 ページの「シングルマスターレプリケーション」および 88 ページの「マルチマスターレプリケーション」では、基本的なレプリケーション戦略の概要を説明します。詳細については、『Sun Java System Directory Server 5 2005Q1 Deployment Plannning Guide』の第 12 章「Designing a Highly Available Deployment」を参照してください。

シングルマスターレプリケーション

次の図は、基本的なレプリケーション概念を表すシングルマスターレプリケーションを示しています。

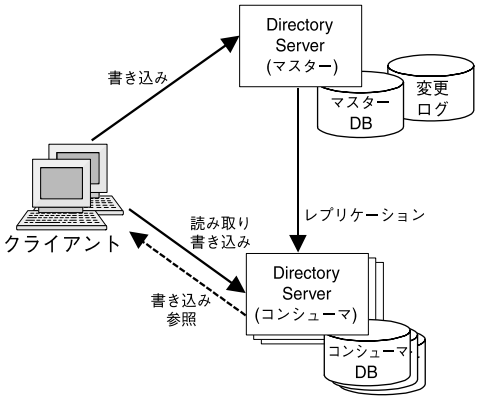


図 **5–7** 単一マスター複製の例

シングルマスターレプリケーションでは、Directory Server の 1 つのインスタンスがマスターディレクトリデータベースを管理し、すべての変更を記録します。マスターデータベースは、任意の数のコンシューマデータベースにレプリケートされます。Directory Server のコンシューマインスタンスは、読み取り操作および検索操作用に最適化されています。コンシューマに対する書き込みは、マスターに反映されます。マスターは、定期的にコンシューマデータベースを更新します。

シングルマスターレプリケーションには、次の利点があります。

- データベースの読み取り操作および書き込み操作用に最適化された Directory Server のシングルインスタンス
- 読み取り操作および検索操作用に最適化された、Directory Server の任意の数のコンシューマインスタンス
- Directory Server のコンシューマインスタンスの水平方向のスケーラビリティー

マルチマスターレプリケーション

次の図は、ディレクトリアクセスをグローバル規模で分散するために使用される可能性のあるマルチマスターレプリケーション戦略を示しています。

マルチマスターレプリケーションでは、Directory Server の 1 つ以上のインスタンスがマスターディレクトリデータベースを管理します。各マスターは、マスターデータベースの同期手順を規定するレプリケーションアグリーメントを持ちます。各マスターは、任意の数のコンシューマデータベースに対してレプリケートを行います。シングルマスターレプリケーションの場合と同様に、Directory Server のコンシューマインスタンスは、読み取りアクセスおよび検索アクセス用に最適化されています。コンシューマに対する書き込みは、マスターに反映されます。マスターは、定期的にコンシューマデータベースを更新します。

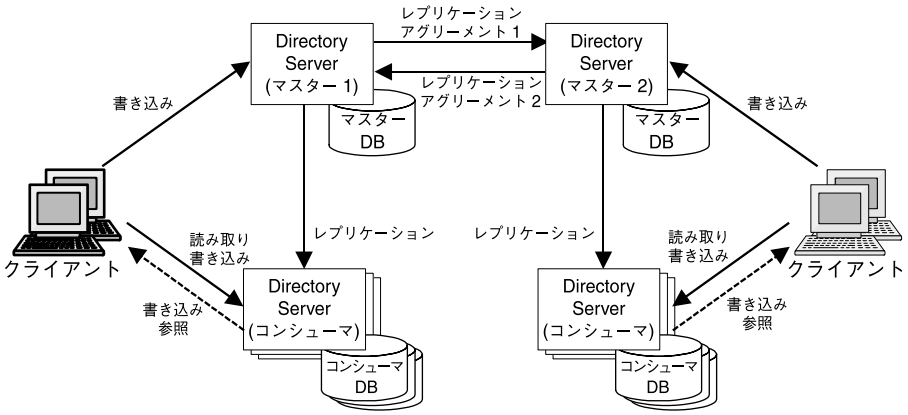


図 **5–8** マルチマスターレプリケーションの例

マルチマスターレプリケーション戦略では、シングルマスターレプリケーションのすべての利点が提供されることに加えて、マスターに対する更新にロードバランスを実行できる可用性戦略が提供されます。また、ディレクトリ操作のローカル制御を提供する可用性戦略も実装できます。 このことは、グローバル規模で分散しているデータセンターを持つ企業にとって重要な考慮事項です。

スケーラビリティーのための戦略の決定

スケーラビリティーとは、システムに処理能力を追加する能力です。 これは、通常はシステムリソースの追加によって行われ、配備アーキテクチャーの変更はスケーラビリティーの対象に含まれません。要件を分析するときに、通常は、ビジネス要件と使用パターンの分析に基づいて、システムの成長予測が立てられます。システムのユーザー数や、目的達成に必要なシステムの処理能力に関する予想は、配備されるシステムの実際の数値とは大きく異なることが珍しくありません。予想の不一致に対応できるだけの十分な柔軟性を設計に盛り込む必要があります。

スケーラブルな設計とは、システムが追加リソースでアップグレードされるまで、増大する負荷を処理できる、十分な潜在処理能力が含まれている設計です。スケーラブルな設計では、システムの設計を変更することなく、増大する負荷の処理に容易に対応できます。

潜在処理能力

潜在処理能力は、例外的なピーク負荷に容易に対応できるように、パフォーマンスと可用性のためのリソースをシステムに含めることで得られるスケーラビリティーです。また、配備されるシステムで潜在処理能力がどのように使用されるかを監視することも、リソースの追加によるシステムの規模拡大が必要となるタイミングを計る上で役立ちます。潜在処理能力は、設計に安全性を組み込む手法の 1 つです。

ユースケースの分析は、例外的なピーク負荷が発生するシナリオを特定する上で役立ちます。例外的なピーク負荷の分析と、予期せぬ急成長に対応するための方法を考慮して、システムに安全性をもたらす潜在処理能力を設計します。

システム設計は、想定される負荷を妥当な期間にわたって処理できる必要があります。 この期間は、通常、運用開始後の最初の 6 〜 12 か月です。定期的に行われる保守を通じて、必要に応じてリソースを追加したり、処理能力を向上します。システムの定期的なアップグレードをスケジューリングできれば理想的ですが、多くの場合、処理能力向上の必要性を予測することは困難です。システムをいつアップグレードするかは、リソースの慎重な監視と、ビジネス予測に頼ることになります。

ソリューションを段階的に実装する計画の場合は、各フェーズで予定されるその他の向上内容に合わせて、システムの処理能力の向上をスケジューリングする必要がある場合もあります。

スケーラビリティーの例

この節の例では、Messaging Server を実装するソリューションの水平方向のスケーリングおよび垂直方向のスケーリングを説明します。垂直方向のスケーリングの場合、CPU をサーバーに追加することで、増大する負荷を処理します。水平方向のスケーリングの場合、負荷の分散に使用するサーバーを追加することで、増大する負荷を処理します。

この例の基準は、ロードバランスのために分散された 2 つのメッセージストアインスタンスによってサポートされる 50,000 ユーザーベースを前提としています。各サーバーには 2 つの CPU があり、合計で 4 CPU です。次の図は、25 万人または 200 万人のユーザーに増大する負荷を処理するために、このシステムの規模をどのように拡大できるかを示しています。

注 – 90 ページの「スケーラビリティーの例」は、垂直方向のスケーリングと水平方向のスケーリングの違いを示しています。この図には、ロードバランス、フェイルオーバー、使用パターンの変化など、スケーリング時に考慮するべきその他の要因は示されていません。

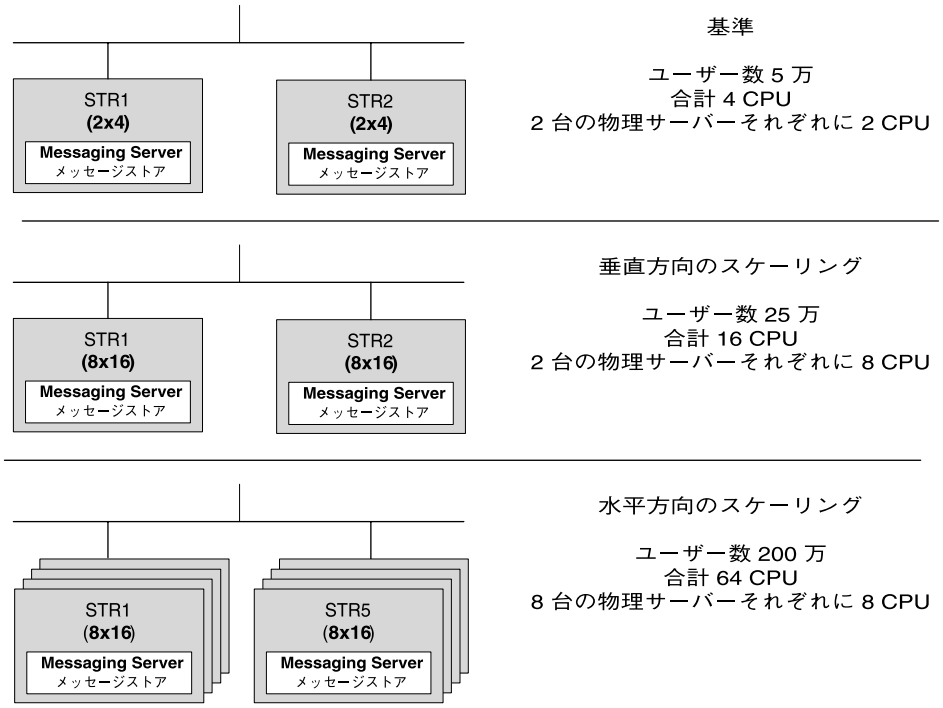


図 **5–9** 水平方向のスケーリングと垂直方向のスケーリングの例

パフォーマンス障害の特定

成功する配備設計の鍵の 1 つには、潜在的なパフォーマンス障害を特定し、それらを回避するための戦略を策定することです。パフォーマンス障害は、データにアクセスする速度が指定されたシステム要件に満たない場合に発生します。

システム内のデータアクセスポイントに関する次の表に示されているように、障害はハードウェアのクラス別に分類されます。この表は、各ハードウェアクラスにおける障害に対する対応策も示しています。

表 **5–7** データアクセスポイント

ハードウェアクラス	相対アクセス速度	パフォーマンス改善のための対応策
プロセッサ	ナノ秒	垂直方向のスケーリング: 処理能力の追加、プロセッサキャッシュの強化を行います
		水平方向のスケーリング: ロードバランス用の並列処理能力を追加します
システムメモリ (RAM)	マイクロ秒	システムメモリを特定の作業専用にします
		垂直方向のスケーリング: メモリを追加します
		水平方向のスケーリング: 並列処理およびロードバランス用に追加インスタンスを作成します
ディスクの読み書き	ミリ秒	ディスクアレイ (RAID) でディスクアクセスを最適化します
		ディスクアクセスを、読み取り専用や書き込み専用など、特定の機能専用にします
		頻繁にアクセスされるデータをシステムメモリにキャッシュします
ネットワークインタフェース	ネットワーク上のノードの帯域幅とアクセス速度によって異なります	帯域幅を増やします
		セキュリティー保護されたデータを転送する際、アクセラレータハードウェアを追加します
		ネットワーク内のノードのパフォーマンスを改善して、データを利用しやすくします

注 – 91 ページの「パフォーマンス障害の特定」は、相対アクセス速度に基づくハードウェアクラスを示しています。ディスクなど、低速なアクセスポイントは、障害の原因となる可能性が高いことを意味しています。また、大きな負荷を処理する能力がないプロセッサも、障害の原因となる可能性があります。

通常、配備設計は、配備内の各コンポーネントとそれらの依存関係について、基準となる処理能力の見積りから始めます。その後、システムメモリおよびディスクアクセスに関連する障害の回避方法を決定します。最後に、ネットワークインタフェースを調べて潜在的な障害を特定し、それらを克服する戦略に力を注ぎます。

ディスクアクセスの最適化

配備設計で不可欠な要素の 1 つは、LDAP ディレクトリなど、頻繁にアクセスされるデータセットに対するディスクアクセスの速度です。ディスクアクセスは、データに対する最も低速なアクセスであり、パフォーマンス障害の原因となる可能性があります。

ディスクアクセスを最適化する方法の 1 つは、書き込み操作と読み取り操作を分けることです。書き込み操作が読み取り操作より負荷が大きいばかりでなく、読み取り操作 (LDAP ディレクトリに対する検索操作) は、通常、書き込み操作 (LDAP ディレクトリのデータに対する更新) よりはるかに頻繁に実行されます。

ディスクアクセスを最適化する別の方法は、ディスクをさまざまな種類の入出力操作の専用とすることです。たとえば、トランザクションログやイベントログなどの Directory Server のロギング操作と、LDAP の読み書き操作に別個のディスクアクセスを提供します。

また、読み書き操作専用の Directory Server のインスタンスを 1 つ以上実装すること、読み取りと検索のためのアクセス用に、ローカルサーバーに分散された、レプリケートされたインスタンスを使用することも検討します。連鎖およびリンクのオプションも、ディレクトリサービスへのアクセスを最適化するために利用できます。

ディスクアクセスの計画におけるさまざまな要因については、『Sun Java System Directory Server 5 2005Q1 Deployment Plannning Guide』の第 6 章「Defining System Characteristics」を参照してください。この章では、次の内容が取り上げられています。

- メモリおよびディスクの最小容量要件**:** さまざまなサイズのディレクトリに必要なディスクとメモリの見積もりが示されています。
- キャッシュアクセス用の物理メモリのサイズ設定**:** Directory Server の使用計画に応じたキャッシュサイズの見積もりおよびメモリの合計使用量の計画に関するガイドラインが示されています。
- ディスクサブシステムのサイズ設定**:** ディレクトリサフィックスおよびディスクの使用に影響する Directory Server 要因に応じたディスク容量要件の計画について説明されています。また、ディスクへのファイルの分散についても、さまざまなディスクアレイの選択肢を挙げながら説明されています。

リソースの最適な使用方法の設計

配備設計は、QoS 要件を満たすために必要となるリソースを見積もるだけではありません。配備設計では、可能な選択肢をすべて分析し、コストを最小化しつつ QoS 要件を満たす最善のソリューションの選択も行います。ある領域での利点が、別の領域のコストで相殺されないように、設計上のそれぞれの意思決定による得失評価を分析する必要があります。

たとえば、可用性のための水平方向のスケーリングは、全体的な可用性を向上するかもしれませんが、保守とサービスのコストは上昇します。パフォーマンスのための垂直方向のスケーリングは、費用効率良く処理能力を向上できるかもしれませんが、一部のサービスでは追加された処理能力を活用し切れない可能性があります。

設計戦略を完成させる前に、リソースの使用と提案ソリューションの全体的なメリットのバランスについて、決定事項をよく見直してください。通常、この分析では、ある領域のシステム品質が、別のシステム品質にどのように影響するかを調べます。次の表は、いくつかのシステム品質と、それに対応するリソース管理に関する考慮事項を示しています。

表 **5–8** リソース管理に関する考慮事項

システム品質	説明
パフォーマンス	個々のサーバーに CPU を集中させるパフォーマンスソリューションで、サービスは処理能力を効率的に活用できるか。(たとえば、一部のサービスでは、効率的に利用できる CPU の数に制限がある。)
潜在処理能力	採用した戦略により、想定されるパフォーマンスを超過した負荷が処理されるか。
	過度な負荷の処理に、サーバーの垂直方向のスケーリング、他のサーバーへのロードバランスのどちらを使用するか、または両方使用するか。
	配備の規模拡大に関する次回の目標を達成するまで、例外的なピーク負荷を処理できるだけの潜在処理能力があるか。
セキュリティー	セキュリティー保護されたトランザクションの処理に必要なパフォーマンス上のオーバーヘッドは十分に考慮されているか。
可用性	水平方向の冗長性ソリューションで、長期的な保守コストを十分に想定しているか。
	システムの保守に必要な予定された停止時間は考慮されているか。
	ハイエンドサーバーとローエンドサーバーのコストのバランスはとれているか。
スケーラビリティー	配備の規模拡大に関する一連の達成目標は設定されているか。
	配備の規模拡大に関する次回の目標を達成するまで、想定される負荷の増大に対応できるだけの十分な潜在処理能力を提供する戦略を持っているか。
保守性	管理、監視、保守のコストを踏まえて可用性を設計しているか。
	管理コストを作成するために、委任管理ソリューション (エンドユーザーが一部の管理作業を実行できるようにする) を検討したか。

リスクの管理

サービス品質要件や使用パターンの分析など、配備設計が基づく情報の多くは、実証的なデータではなく、ビジネス分析から最終的に得られた予測に基づいています。これらの予測は、ビジネス動向での予期しない状況、データ収集方法上の問題、あるいは単なる人為ミスなど、さまざまな理由から正確でない可能性があります。配備設計を完了する前に、設計の土台となった分析結果を再確認し、見積もりや予測のずれが設計で十分に考慮されているかどうかを確認してください。

たとえば、使用パターンの分析でシステムの実際の使用状況が低く見積もられている場合、発生するトラフィックに対応し切れないシステムを構築してしまうリスクが生じます。予定を下回るパフォーマンスを示す設計は、失敗と見なされます。

一方で、必要処理能力より数倍強力なシステムを構築した場合、別の場所で使用できたはずのリソースを無駄にすることになります。重要なことは、要件に加えて安全性のマージンを含めるが、リソースの浪費は回避するということです。

リソースの浪費も設計の失敗をもたらします。 十分に活用されなかったリソースを別の領域に適用できたためです。また、行き過ぎたソリューションを適用することは、利害関係者に、誠実に契約を履行していないと見なされる可能性もあります。

配備例のアーキテクチャー

次の図は、この白書ですでに紹介した配備例の完全な配備アーキテクチャーを示しています。この図は、配備アーキテクチャーを表現する方法を示しています。

注意 – 次の図の配備アーキテクチャーは、説明のためのアーキテクチャーに過ぎません。実際に設計、構築、またはテストされた配備を示しているわけではないので、配備計画のアドバイスと見なさないでください。

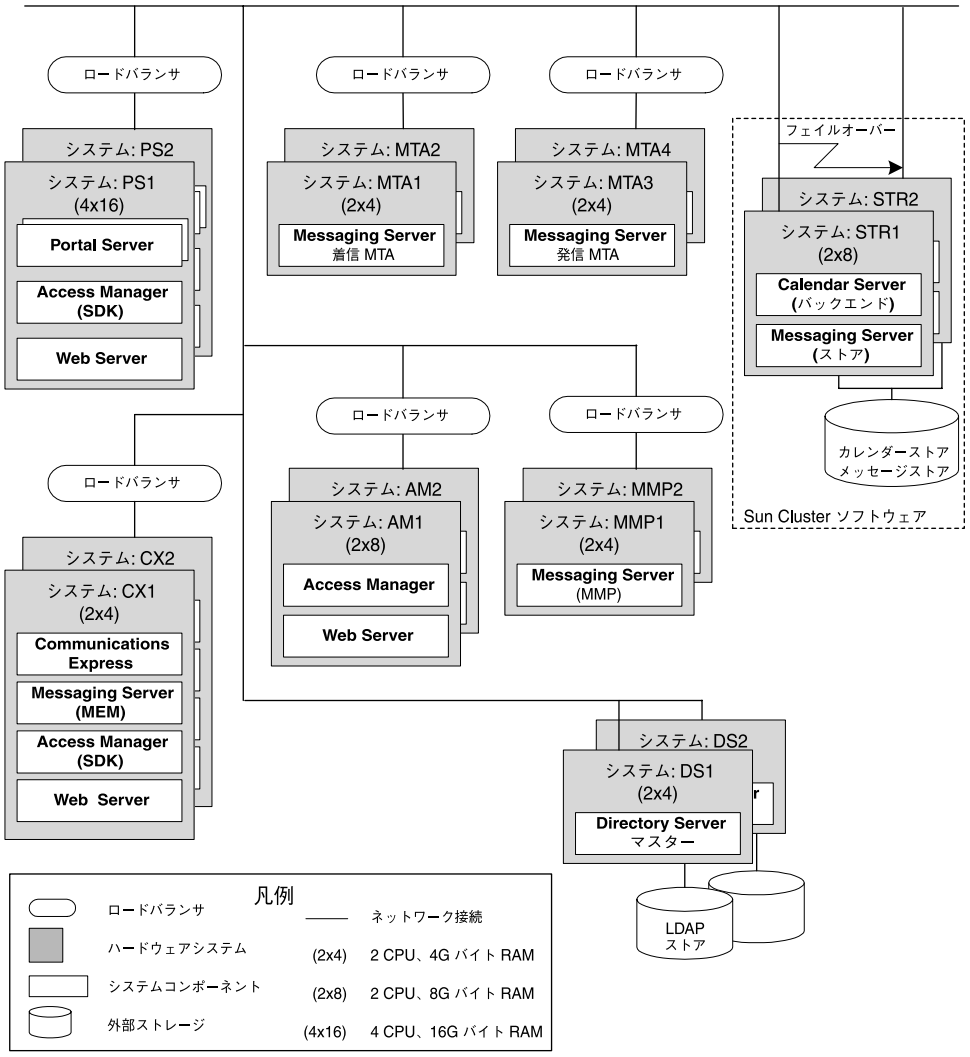


図 **5–10** 配備例のアーキテクチャー

第 6 章

配備設計の実装

ソリューションライフサイクルの実装フェーズでは、配備設計で作成された仕様と計画に基づいて、配備アーキテクチャーを構築してテストを行い、最終的に配備を本稼働環境にロールアウトします。実装についてはこのマニュアルの対象範囲外ですが、この章では、実装フェーズの概要を示します。

この章で説明する内容は、次のとおりです。

配備設計の実装について

配備アーキテクチャーが承認され、実装の仕様と計画が完成したら、ソリューションライフサイクルの実装フェーズに進みます。実装は複雑な一連のプロセスと手順であり、成功を確実にするには、慎重な計画が必要です。実装には次のタスクが含まれます。

- ネットワークとハードウェアインフラストラクチャーの構築
- インストール計画に基づく、ソフトウェアのインストールと設定
- 既存のアプリケーションから現在のソリューションへのデータの移行
- ユーザー管理計画の実装
- テスト計画に基づく、パイロットまたはプロトタイプの設計とテスト環境への配備
- テスト計画に基づく、機能テストおよびストレステストの設計と実行
- ロールアウト計画に基づく、テスト環境から本稼働環境へのソリューションのロールアウト

- トレーニング計画に基づく、配備の管理者およびユーザーのトレーニング

実装の詳細については、このマニュアルの対象範囲に含まれません。以降の各節では、これらのタスクの一部について、概要のみを説明します。

ソフトウェアのインストールと設定

企業の分散型アプリケーション用の Sun Java™ Enterprise System のインストールと設定においては、数多くのタスクと手順を計画し、検討する必要があります。配備設計フェーズで、Java Enterprise System ソフトウェアのインストールに必要なインストールと設定に関する情報が示された高レベルな配備アーキテクチャーに基づいて、インストール計画を作成します。

このインストール計画での重要な点は次のとおりです。

- インストールの順序と種類の決定
- 以前にインストールされたソフトウェアおよびインストールの受容可能性についてのホストの調査
- インストールする各 Java Enterprise System コンポーネントの設定情報の収集

『Sun Java Enterprise System 2005Q4 Installation Planning Guide』では、インストール計画に必要な情報を収集する方法が詳細に説明されています。『Sun Java Enterprise System 2005Q4 Installation Reference』に示されている詳細な設定情報とワークシートを使用して、この情報についてのドキュメントを作成できます。また、『Sun Java Enterprise System 2005Q4 インストールガイド(UNIX 版)』には、複数の Java Enterprise System コンポーネントが関わる一般的なインストールシナリオについてのガイドラインも示されています。詳細については、『Sun Java Enterprise System 2005Q4 インストールガイド(UNIX 版)』の第 1 章「インストールの準備」を参照してください。

パイロットとプロトタイプの開発

通常、Java Enterprise System の配備は、主に Java Enterprise System が提供するサービスに基づく配備と、Java Enterprise System サービスに統合される多数のカスタムサービスを必要とする配備の 2 種類に分類されます。前者の配備を 80:20 配備 (Java Enterprise System が提供するサービスが 80%)、後者を 20:80 配備と考えることができます。

80:20 配備では、通常は実装フェーズでテスト用のパイロット配備を開発します。80:20 配備では、すぐに利用できる機能を提供する、完成された Java Enterprise System サービスを使用するため、パイロット配備の開発、テスト、修正から本稼働配備への移行は比較的迅速に行われます。パイロット配備により、ソリューションの機能が検証されるとともに、システムのパフォーマンスレベルに関する情報も得られます。

一方、20:80 配備では、80:20 配備で利用できた相互運用性の実績を持たない、新しいカスタムサービスを使用します。このため、配備の概念を証明するプロトタイプを作成することになり、本稼働開始前の開発、テスト、修正のサイクルをより厳格に行う必要が生じます。プロトタイプにより、テスト環境における提案ソリューションの問題解決能力が判明します。機能が十分であることがプロトタイプによって証明されたあとは、より厳格なテストに進み、その後、パイロット配備に進むことができます。

注 – 企業の実際の配備では、要件に合わせて独自に開発されたサービスが占める割合は大きく異なります。パイロット配備とプロトタイプ配備をどのようにテストに使用するかは、複雑さと配備の性質によって異なります。

パイロット配備とプロトタイプ配備のテスト

パイロット配備とプロトタイプ配備をテストする目的は、配備がシステム要件を満たし、ビジネス上の目的を達成できるかどうかを、テスト環境でできるだけ正確に検証することにあります。

特定したすべてのユースケースに基づいて機能テストを行なってシナリオを作成し、配備の適格性を計測するための手法を確立できれば理想的です。機能テストには、ビジネス要件が満たされるかどうかを判断するための、選択されたベータユーザーグループに対する限定的な配備が含まれることもあります。

ストレステストは、ピーク負荷時のパフォーマンスを測定します。通常、これらのテストでは、シミュレートされた一連の環境とロードジェネレータを使用して、データスループットとパフォーマンスを測定します。一般に、配備のシステム要件がストレステストの設計と合格の基準となります。

注 – システム要件が明確に定義されず、予測の基本となる過去の実装がなく、新たな開発を大量に必要とする大規模な配備では、機能テストとストレステストは特に重要です。

テストによって配備の設計仕様に関する問題が検出されることもあり、配備を本稼働環境に送り出すまでに、設計、構築、およびテストを何度か繰り返さなければならないこともあります。プロトタイプ配備のテスト時に、配備設計に関する問題が発見される場合があります。 その場合は、ソリューションライフサイクルのそれまでのフェーズに戻って作業を繰り返し、問題に対処します。

配備計画を徹底的にテストしてから、パイロット配備に進んでください。パイロット配備は、それまでの一連のテストですでに配備設計が検証されていることを前提とします。パイロット配備のテストで発見される問題は、通常、配備設計のパラメータ内で対処できる必要があります。

テストは本稼働環境を完璧にシミュレートすることはできません。 また、配備されたソリューションの性質は変化発展する可能性があります。 このため、調整、保守、サービスが必要な領域を特定できるように、配備されたシステムの監視を継続する必要があります。

本稼働配備のロールアウト

パイロット配備、または概念を証明するための配備がテスト条件に合格すると、配備を本稼働環境にロールアウトする準備が整います。通常は、本稼働環境へのロールアウトは段階的に行われます。段階的なロールアウトは、多数のユーザーに影響する大規模な配備では特に重要です。

段階的な配備は、少数のユーザーセットから開始し、すべてのユーザーが配備を利用できるようになるまで、ユーザーベースを段階的に拡張します。また、限定されたサービスセットから段階的な配備を開始し、最終的に残りのサービスを配備することもできます。サービスの段階的な配備は、本稼働環境でサービスに発生する可能性のある問題を分離、特定、解決する上で役立ちます。

索引