Microsoft Solutions Framework

White Paper

Published: 2004

Microsoft Solutions Frameworkの詳細情報は
http://www.microsoft.com/japan/msdn/vstudio/productinfo/enterprise/msf/ をご覧ください

MSF プロセス モデル v. 3.1

目次

クレジット

MSF Team, Microsoft

Scott Getchell, Program Manager, US Frameworks

Laura Hargrave, Technical Editor, US Frameworks

Paul Haynes, Program Manager, US Frameworks

Mike Lubrecht, Technical Writer, US Frameworks

Pervez Kazmi, Program Manager, US Frameworks

Rob Oikawa, Principal Consultant, Microsoft Consulting Services, US

Enzo Paschino, Program Manager, US Frameworks

Allison Robin, Director, US Frameworks

Mark Short, Program Manager, US Frameworks

Reviewers

Andrew Delin, Microsoft Consulting Services, Australia

Paulo Henrique Leocadio, Microsoft Consulting Services, LATAM

Joe Lopesilvero, Microsoft Consulting Services, US

David Millet, Microsoft Consulting Services, US

Thierry Paquay, Microsoft Premier Support, US

Paulo Rocha, Microsoft Consulting Services, New Zealand

Anthony Saxby, Microsoft Consulting Services, UK

Ralph Schimpl, Microsoft Consulting Services, Austria

Ron Stutz, Microsoft Consulting Services, US

Brian Willson, Microsoft Consulting Services, US

Andres Vinet, Microsoft Consulting Services, Chile

※ 本ホワイトペーパーの日本語翻訳にあたっては、Microsoft Consulting Services の小泉 浩氏に多大なご協力をいただきました。この場を借りて御礼申し上げます。

Part Number: 602-i399a

要約

MSF プロセス モデルは、IT ソリューションの構築および展開(デプロイメント) の活動のおおよその順序を記述します。一連の詳細な作業手順を規定する場合に比べ、広範な IT プロジェクトに対応できる柔軟性を持っています。MSF プロセス モデルは 2 つの業界標準モデル、すなわちウォーターフォールとスパイラルを組み合わせています。この新しいバージョンの MSF モデルの画期的な点は、プロジェクトの発端から運用環境への展開まで、ソリューションのライフサイクル全体をカバーしていることです。ソリューションは実際に展開され運用されてからでないと何の価値も生み出さないため、このことは、プロジェクト チームが顧客のビジネス上の価値に集中するのに役立ちます。

MSF はマイルストン駆動のプロセスです。マイルストンは、プロジェクトにおいて重要な成果物が完成しレビューされるポイントです。プロジェクトに関する多くの重要な質問について吟味します。たとえば、チームはプロジェクトのスコープに合意しているか、先に進むために十分な計画を立てたか、予定どおりのものを構築したか、ソリューションは顧客の役に立っているか、などです。

MSF プロセス モデルは、短期間の開発サイクルでソリューションを増分的に構築するイテレーション (繰り返し) を進めることで、変化するプロジェクト要求に対応するように設計されています。

プロジェクト チームがプロセス モデルを使用して成功するのに役立つ、多くのプラクティスが推奨されています。

フレームワークの概要

IT プロジェクトの成功を最大にするために、Microsoft は、Microsoft テクノロジの上に構築するソリューションの効果的な設計、開発、展開 、運用、およびサポートに関するパッケージ化されたガイドを提供しています。この知識は、Microsoft 社内のソフトウェア開発やサービス運用の大規模プロジェクトから得た経験、企業顧客のためのプロジェクトを遂行する Microsoft のコンサルタントの経験、および世界中の IT 業界の最高のナレッジから集められたものです。このガイドは、互いに補い合う密接に統合された 2 つの知識体系、すなわち「フレームワーク」に編成されています。これらが Microsoft Solutions Framework (MSF) と Microsoft Operations Framework (MOF) です。

納期どおりに予算の範囲内でビジネス ソリューションを作成するには、実証済みのアプローチが必要です。MSF は、IT ソリューションの計画、設計、開発、および展開に成功するための実証済みのプラクティスを提供します。MSF が提供するのは規定的な方法論ではなく、柔軟性に富み規模調整可能なフレームワークであるため、あらゆる規模の組織やプロジェクト チームのニーズに対応できます。MSF のガイドは、人、プロセス、テクノロジの要素、およびほとんどのプロジェクトが遭遇するそれらのトレードオフを管理するための原則、モデル、規範によって構成されています。MSF の詳細については、
http://www.microsoft.com/japan/msdn/vstudio/productinfo/enterprise/msf/ を参照してください。

MOF は、Microsoft の製品やテクノロジを使用して構築された IT ソリューションでミッションクリティカルなシステムの信頼性、可用性、サポート性、可管理性を達成するためのテクニカル ガイドを提供します。MOF のガイドは、複雑な異種分散 IT 環境の運用に関連する人、プロセス、テクノロジ、および管理の問題に対応しています。MOF は、英国政府機関の CCTA (Central Computer and Telecommunications Agency) の ITIL (IT Infrastructure Library) で文書化された業界のベスト プラクティスをベースにしています。MOF の詳細については、
http://www.microsoft.com/japan/technet/itsolutions/techguide/mof/ を参照してください。

はじめに

一般に種々のプロセスのモデルはプロジェクト活動の順序を確立し、これによりプロジェクトのライフサイクル全体を表します。現在、ビジネスの世界ではさまざまなプロセス モデルが用いられています。MSF プロセス モデルの起源は、Microsoft がアプリケーション開発に使用しているプロセスです。このモデルは他の一般的なプロセス モデルの最も有効な原則のいくつかを、どのような種類のプロジェクトにも適用できる単一のモデルに結合して発展させた、フェーズ ベースでマイルストン駆動のイテレーティブ(繰り返し)型のモデルです。このモデルは伝統的なアプリケーション開発環境にも適用できますが、e コマース、Web 分散型アプリケーション、および将来現れるであろうその他の多面的イニシアチブを目的としたエンタープライズ ソリューションの開発と展開にも同様に適しています。

その他のプロセス モデル

現在は、ウォーターフォール モデルとスパイラル モデルの 2 つが IT 業界で最も一般的に使用されているプロセス モデルです。

- *ウォーターフォール モデル*[1]
- このモデルでは、マイルストンを移行および評価ポイントとして使用します。ウォーターフォール モデルでは、各々の一連のタスクを完了しないと次のフェーズを開始できません。ウォーターフォール モデルは、開始時に不変で固定的なプロジェクト要件を明確に示せるプロジェクトで最も良く機能します。各フェーズ間の固定の移行ポイントは、スケジュールの追跡および実行責任と説明責任割り当てを容易にします。

  図 1 は ウォーターフォール モデルを表したものです。マイルストンは菱形で、フェーズは矢印で示しています。

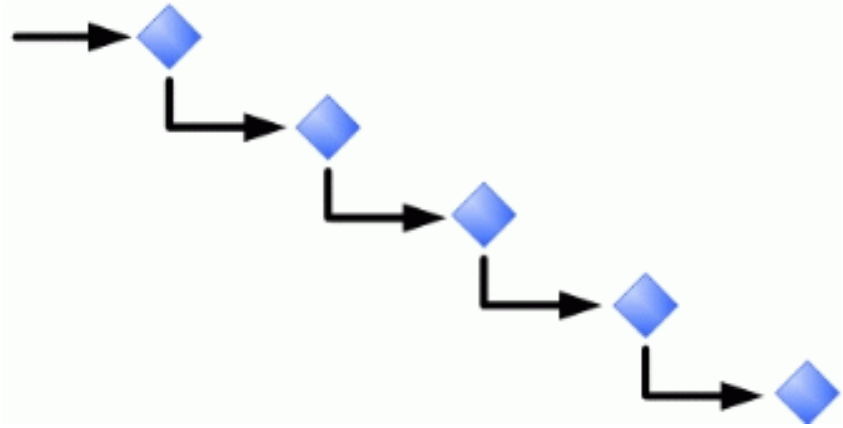

図 1: ウォーターフォール モデル

- *スパイラル モデル*[2] このモデルでは、プロジェクトの要求と見積りを継続的に洗練させる必要性に焦点を置いています。図 2 に示すスパイラル モデルは、非常に小さいプロジェクトでアプリケーションを迅速に開発する場合に使用すると、非常に効果的であり得ます。顧客がプロジェクトのあらゆる段階でフィードバックと承認を提供するため、このアプローチは開発チームと顧客との間に非常に大きい相乗効果を促進します。しかし、このモデルは明確なチェックポイントを含まないため、開発プロセスが混乱する恐れがあります。

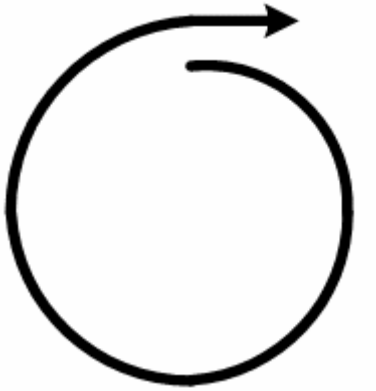

図 2: スパイラル モデル

最高の組み合わせ

図 3 に示す MSF プロセス モデルはウォーターフォール モデルとスパイラル モデルの最良の原則を組み合わせています。MSF は、ウォーターフォール モデルのマイルストンベースの計画から予測可能性の利点を、さらにスパイラル モデルからフィードバックと創造性の利点を引き出しています。マイルストンとフェーズの詳細については、後述します。

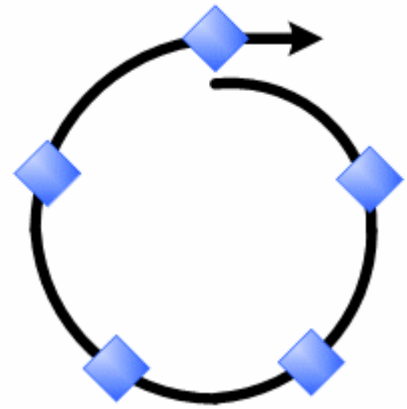

図 3: MSF プロセス モデル

基礎をなす MSF 基本原則

MSF プロセス モデルは、以下の 4 つの MSF 基本原則と直接結び付いています。

ビジョンを共有する

あらゆる共同作業において成功の基礎となるのは、チーム メンバと顧客が共有のビジョン、すなわちソリューションの目的と目標に関する明確な理解を持つことです。チーム メンバと顧客は皆、その活動が組織にもたらさんとしていることについての仮定を抱いています。共有されたビジョンはそのような仮定を明らかにし、すべての関係者が同じ目標の達成に向けて取り組むことを確実にします。

共有のビジョンを明確にしてそれへの確約を得ることは極めて重要なので、MSF プロセス モデルではこれを目的としたフェーズと主要マイルストンを設けています。

俊敏であり続け, 変化を予期する

伝統的なプロジェクトマネジメントの規範とウォーターフォール プロセス モデルでは、最初に要求を明確に述べることが可能で、それらはプロジェクトのライフサイクルにおいて大きく変化しないと仮定しています。これとは対照的に、MSF では絶え間の無い変化を予期し、管理する必要があるということを基本的な前提にしています。

ビジネス上の価値に集中する

ソリューションを成功させるには、対象が組織か個人かに関係なく、何らかの基本的なニーズを満たし、顧客に価値や便益を提供する必要があります。個人にとっては、コンピュータ ゲームなど、何らかの情緒面のニーズを満たすことが便益であるあるかもしれません。しかし、組織にとって重要な推進要因はビジネス上の価値です。

ソリューションは、実際の運用環境に完全に展開されて初めてビジネス上の価値を提供します。そのため、MSF プロセス モデルのライフサイクルには、ソリューションの開発と展開の両方が含まれています。

オープンなコミュニケーションを促進する

これまで多くの組織やプロジェクトでは、必要になったときにのみ情報を知らせるという原則が採られてきました。言い換えると、情報を与えられるのは、作業を行うのに情報が必要なことを証明できる人だけです。このようなアプローチは多くの場合、成功するソリューションを提供するチームの能力を損なう誤解につながります。

MSF プロセス モデルは、チーム内および主要ステークホルダーとのコミュニケーションについてのオープンで率直なアプローチを規定しています。情報が自由に流れるようにすることで、誤解や無駄な労力の生まれる機会が減るだけでなく、すべてのチー ム メンバが、プロジェクトに関わる不確実性の削減に貢献できるようになります。

このような理由から、MSF プロセス モデルではレビューのタイミングを提供します。文書化された成果物により、プロジェクトの進捗が見える状態に維持され、チーム、ステークホルダー、および顧客の間で十分に伝達されます。

MSF プロセス モデルの主要コンセプト

プロセス モデルを理解するには、MSF で以下の概念や用語がどのように定義されているかを理解することが重要です。

顧客

MSF では顧客とユーザーを区別します。

コンシューマ向けソフトウェア製品、ゲーム、Web アプリケーションに関しては、顧客とユーザーが同じ場合があります。

しかし、ビジネス ソリューションの場合、顧客はプロジェクトを委託し、資金を供給して、ソリューションからビジネス上の価値を得ようと期待する人または組織です。ユーザーは、自身の業務においてソリューションと相互に作用する人です。たとえば、あるチームが、従業員が会社のイントラネットを使って経費報告書を提出できる会社経費報告書作成システムを構築しているとしましょう。ユーザーは従業員ですが、顧客は新しいシステムの構築を任されている経営者の1人です。

- *顧客の参加* IT プロジェクトへの顧客の参加は成功に不可欠です。MSF プロセス モデルでは、顧客が要求を形成および修正し、進捗をレビューするためのチェックポイントを設定する機会が数多くあります。このような活動は、顧客に時間と参加を要求します。
- *内部または外部の顧客* プロジェクトを取り巻く事情により、顧客とチームが同じ組織に属していない場合があります。たとえば、顧客は外部の "供給者"(さまざまな提携企業からなるバーチャルチームかもしれません) と契約している購入者である可能性があります。
- *契約* MSFは、顧客、供給者、およびソリューション チーム間の契約上および法的な関係が非常に重要であり、慎重に管理する必要があることを認めています。このようなアプローチは調達マネジメントと呼ばれ、MSFプロジェクトマネジメント規範のホワイトペーパーで述べられてています。しかし、このトピックに関してはさまざまな情報源からの指針を利用できるため、ここでは詳しく説明しません。

ステークホルダー

ステークホルダーは、プロジェクトの結果に利害関係がある個人またはグループです。その目標や優先順位は、必ずしも顧客のそれと一致するとは限りません。各ステークホルダーには、自身にとって重要な要求や機能があるでしょう。

プロダクトマネジメントの責務には、プロジェクトの主要なステークホルダーを識別し、そのニーズを考慮して、ステークホルダーとの関係を管理することが含まれます。

IT プロジェクトにおいて共通して見られるステークホルダーの例には以下があります。

- チームが構築するソリューションによって、スタッフやビジネス プロセスが変更される部門の管理者
- ソリューションの実行やサポートを担当したり、ソリューションの影響を受ける可能性がある他のアプリケーションを実行する IT 運用スタッフ
- プロジェクト チームに要員を提供している職能組織の管理者

ソリューションとは

日常の使用方法では、ソリューションとは単に問題を解決する戦略または方法です。IT 業界では、製品を "ソリューション" と表現するのが一般的なマーケティング用語になっています。このような事情により、“ソリューション”の正確な意味については混乱はもちろん、懐疑さえあります。

MSF では、"ソリューション" という用語が非常に明確な意味を持っています。つまり、固有の顧客のビジネス上の問題にうまく対応するのに必要な要素 (テクノロジ、ドキュメント、トレーニング、サポートなど) の調和した提供を意味します。MSF はマスマーケット向けの商品を開発するのに実績がありますが、主に個別の顧客に応じて仕立てられたソリューションの提供に重点を置いています。

ソリューションには 1 つ以上のソフトウェア製品を含み得ますが、製品とソリューションの違いは明確にしておく必要があります。その違いを以下の表にまとめます。

製品	MSF ソリューション
マスマーケットのニーズに合わせて設計	個別の顧客のニーズに適合するように設計または調整
パッケージ商品または“ビット”(ダウンロードや CD-ROM など) として提供	プロジェクトとして提供

図 4 は、成功するソリューションの主要な要素を示しています。

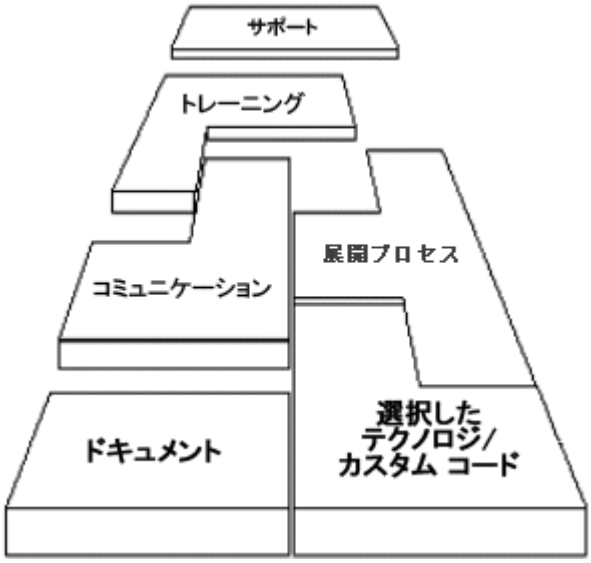

図 4: ソリューションの要素

プロジェクトによって、複雑さや開発に必要な工数は異なります。図 4 で示される要素には、比較的簡単な展開では必要のないものもあります。しかし、多くの複雑で大規模なプロジェクトでは、上述の要素のすべてが必要になりそうです。

付け加えて、以下のような要素も考えられます。

- 選択したテクノロジやカスタム コードは、新規のものであったり、改良または更新されていたり、あるいはコンポーネントが追加されている場合があります。
- テクノロジにはハードウェア、ソフトウェア、周辺機器、またはネットワーク コンポーネントが含まれる場合があります。カスタム コードは、特定プロジェクトのために開発されたコードです。
- トレーニングは、展開するソリューションを使用したりサポートするすべての人に適用されます。
- ドキュメントでは、ソリューションをインストール、保守、サポート、および使用するのに必要なすべての情報について言及しています。
- サポート プロセスには、バックアップ、復元、災害からの復旧、トラブルシューティング、およびヘルプ デスク機能を実行するのに必要な手続きが含まれます。
- 外部のコミュニケーションは、展開の進捗状況とソリューションが外部のステークホルダーにどのように影響するかについて、外部のステークホルダーに通知し続ける必要があります。
- 展開プロセスには、ハードウェアとソフトウェアを展開するためのインストール/アンインストール手順、自動展開ツール、および緊急ロールバック手順が含まれます。

ベースライン

MSF プロセス モデルでは、ベースラインは何かの測定や比較に用いられる尺度または既知の状態です。ベースラインの確立は、MSF で繰り返されるテーマです。MSF でベースラインを設定する成果物の例には、ソース コード、サーバー構成、スケジュール、仕様、ユーザー マニュアル、および予算などがあります。ベースラインがなければ、変更を管理することはできません。

スコープ

スコープは、プロジェクトで提供される成果物およびサービスの集合です。スコープは、共有されたビジョンを支援するために実行しなければならないことを定義します。スコープは、現実に対応しつつ、共有されたビジョンを統合し、顧客がリリースの成功に不可欠であると考えるものを反映します。スコープ定義の一部として、優先度の低い機能は将来のプロジェクトに移します。

スコープを定義する利点は以下のとおりです。

- 長期的なビジョンを達成可能な塊に分割します。
- 各リリースに持たせる機能を定義します。
- 変化に対する柔軟性を許容します。
- トレードオフのベースラインを提供します。

プロジェクト チームが提供する作業とサービスのスコープだけでなく、ソリューションの機能のスコープを定義して管理する必要があります。

"スコープ" という用語には、ソリューション スコープおよびプロジェクト スコープの 2 つの側面があります。両者には相関関係がありますが、同じではありません。この違いを理解すると、チームがプロジェクトのスケジュールとコストを管理するのに役立ちます。

*ソリューション スコープ*は、ソリューションの機能と成果物 (コード以外の成果物を含む) について記述します。機能は、アプリケーションまたはハードウェアの望ましい、あるいは顕著な側面です。たとえば、印刷する前にプレビューできるのはワープロ アプリケーションの機能であり、電子メール メッセージを送信する前に暗号化できるのはメッセージング アプリケーションの機能です。また、付属のユーザー マニュアル、オンライン ヘルプ ファイル、運用ガイド、およびトレーニングも総合的なソリューションの機能です。

*プロジェクト スコープ*は、ソリューション スコープで記述した各アイテムを提供するために、チームが実行する作業について記述します。組織によっては、プロジェクト スコープを実行すべき作業指示書 (SOW; statement of work) として定義します。

プロジェクト スコープを明確にすると、以下の利点があります。

- チームは、実行しなければならない作業の識別に集中できます。
- 大きくて曖昧なタスクを小さくて理解しやすいタスクに分解できます。
- 進捗報告の準備など、いかなる特定の機能とも明確な関連のない特定のプロジェクト作業を識別します。
- 作業をチーム内の下請業者やパートナーに細分するのが容易です。
- ソリューションに対してチームが責任を負う部分と責任を負わない部分を明確にします。
- ソリューションのすべての部分に対して開発または保守に対する責任を負う担当者を明確にします。特に大規模なソリューションでは、ソリューションの一部である機能がプロジェクト チームの成果物に含まれないことがあります。たとえば、チームが会社の ERP (Enterprise Resource Management) システムと対話する企業調達ソリューションを開発するとします。統合は全体的なソリューション スコープの一部ですが、必ずしもそのチームのプロジェクト スコープに入っているとは限りません。

トレードオフの管理

プロジェクトの成功にはスコープの管理が必要不可欠です。多くの IT プロジェクトは、スコープの管理が不十分なために、完成が遅れたり、大幅に予算をオーバーして失敗します。スコープの管理には、早期にスコープを明確化すること、および十分なプロジェクトの追跡と変更管理が含まれます。

IT プロジェクトには不確実性とリスクがつきものであるため、トレードオフに対する効果的な意思決定が成功の秘訣です。

トレードオフ トライアングル

プロジェクトには、資源 (人と資金) 、スケジュール (時間) 、および製品機能 (スコープ) のプロジェクト変数の間に周知の関係があります。これらの変数は、図 5 に示すように三角形を構成する関係にあります。

一旦この三角形を確立した後は、ある 1 辺に変更を加えた場合、他の 1 辺または 2 辺を修正して、プロジェクトのバランスを維持する必要があります。この修正には潜在的に、最初に変更を加えた辺も含まれます。

顧客のニーズに一致するソリューションを顧客が必要とする時に展開するには、資源、展開日、および製品機能の適正なバランスを見つけることが重要です。

顧客はお気に入りの機能を削除したがらないことがあります。トレードオフ トライアングルは、制約と現在のトレードオフの選択肢を説明するのに役立ちます。

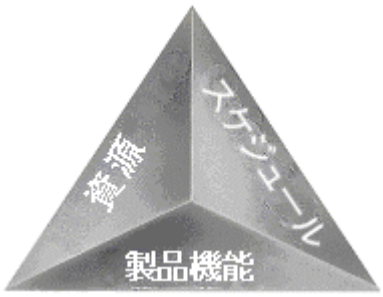

図 5: トレードオフ トライアングル

機能には、交渉不可能と思われる一定の品質があります。品質は、三角形を四面体 (または三面のピラミッド) に変える第 4 の次元と見なすことができます。品質基準を下げれば、同時に資源を削減し、スケジュールを短縮し、および製品機能を増強できますが、これは明らかに失敗のレシピです。

プロジェクト トレードオフ マトリクス

トレードオフを管理するもう 1 つの強力なツールは、図 6 に示すプロジェクト トレードオフ マトリクスです。これはトレードオフを決定する際の既定の優先順位に関するチームと顧客の間の合意で、プロジェクトの初期に作成されます。必要であれば、既定の優先順位に例外を設定することもできます。既定の優先順位を確立する主な利点は、トレードオフについての論争を少なくするのに役立つことです。

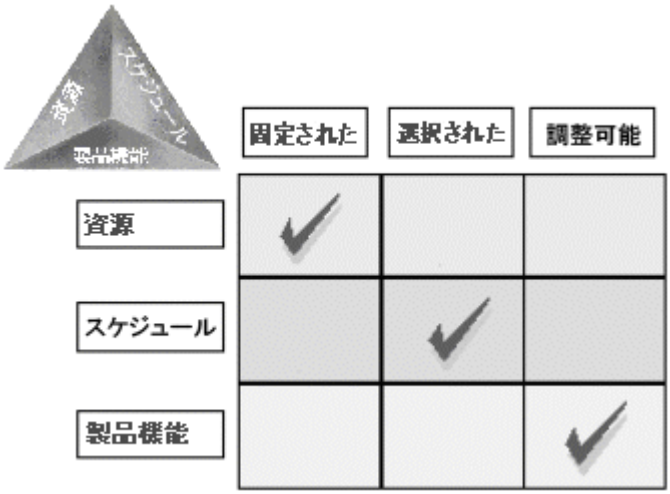

図 6: トレードオフ マトリクス

図 6 は、Microsoft 製品チームが使用する典型的なトレードオフ マトリクスを示しています。このマトリクスは、本質的に変更できないプロジェクトの制約 (“固定された”の列に表示) 、できるだけ優先したい制約 (“選択された”の列に表示) 、および固定されたおよび選択された制約に対応するように調整できる制約 (“調整可能”の列に表示) を識別するのに役立ちます。

製品機能を安易に削減することはありません。チームと顧客の両方が、すべてのプロジェクの制約を慎重に見直し、難しい選択をする用意をしておかなければなりません。

トレードオフ マトリクスがどのように機能するかを理解するには、資源、スケジュール、および製品機能の変数を以下の文の空白に挿入してみてください。

固定された _____が与えられ、 _____を選択し、必要に応じて _____ を調整する。

論理的には以下のような文が考えられます。

- 固定された資源が与えられ、スケジュールを選択し、必要に応じて製品機能を調整する。
- 固定された資源が与えられ、製品機能を選択し、必要に応じてスケジュールを調整する。
- 固定された製品機能が与えられ、資源を選択し、必要に応じてスケジュールを調整する。
- 固定された製品機能が与えられ、スケジュールを選択し、必要に応じて資源を調整する。
- 固定されたスケジュールが与えられ、資源を選択し、必要に応じて製品機能を調整する。
- 固定されたスケジュールが与えられ、製品機能を選択し、必要に応じて資源を調整する。

チームと顧客が、そのプロジェクトのトレードオフ マトリクスについて完全に確信していることが不可欠です。

プロセス モデルの特徴

MSF プロセスの 3 つの顕著な特徴は以下のとおりです。

- フェーズおよびマイルストンベースのアプローチ
- 繰り返し (イテレーティブ) アプローチ
- ソリューションの構築と展開の統合アプローチ

マイルストンベースのアプローチ

マイルストンベースのアプローチの特徴

マイルストンは MSF の中心的テーマの1つで、プロジェクトの進捗を計画、監視するために使用します。

主要マイルストンと中間マイルストン

MSF では、2 種類のマイルストン、すなわち主要マイルストンと中間マイルストンを区別しています。主要および中間マイルストンの特徴は以下のとおりです。

- 主要マイルストンは、あるフェーズから次のフェーズへの移行や役割間の責任の移行をもたらします。
- MSFは、あらゆる種類の IT プロジェクトに対して十分に一般的な、具体的な主要マイルストンを定義します。
- 中間マイルストンは早期の進捗状況指標として機能し、大規模な作業を実行可能な部分に分割します。
- 中間マイルストンは、プロジェクトのタイプに応じて異なります。MSF では一連の中間マイルストンを提案しますが、チームはそのプロジェクトに適切な具体的な中間マイルストンを定義します。

同期ポイントとしてのマイルストン

主要マイルストンは、プロジェクト ライフサイクルにおいて、チーム全体がマイルストンの成果物を互いに同期させたり、顧客の期待と同期させるポイントです。この時点で、プロジェクトの成果物は顧客、ステークホルダー、およびチームによって正式にレビューされます。主要マイルストンの達成の成功は、チームと顧客がプロジェクトを先に進めることに合意したことを表します。

非常に遅いリリース日を選択することによりプロジェクトを完全に予測可能にできますが、これはコストがかかり、ビジネス ニーズを満たしません。マイルストンにより、顧客とチームはプロジェクト スコープを再確認したり、変化する顧客要求の反映やリスクへの対処のためにスコープを調整したりできます。

マイルストン駆動型の説明責任

プログラムマネジメントの役割は各フェーズにおいてプロセス全体を組織化しますが、各マイルストンを無事達成するには、チームの他の役割の各々に固有のリーダーシップと説明責任が要求されます。プロジェクトが順次各フェーズを推移するにつれ、各役割の活動レベルは変化します。マイルストンの使用は、このようなプロジェクトへの関与度合いの増減を管理するのに役立ちます。

各フェーズを推進する役割

5 つの対外的なマイルストンのそれぞれとチームの役割を並べてみると、どの役割が各マイルストンの達成に対して主に責任を持つかが明確になります。これにより明確な説明責任が与えられます。プロジェクトが異なるフェーズに移行する際には、多くの場合、そのプロセスの一部に他の役割への責任の移行が含まれます。

以下の表に、各マイルストンを推進する役割を示します。各マイルストンの完了は 1 つか 2 つの役割によって推進されますが、プロジェクト ライフサイクルを通じてすべての役割が参加します。

マイルストン	主要な推進者
ビジョン/スコープ合意	プロダクトマネジメント
プロジェクト計画合意	プログラムマネジメント
スコープ完成	開発とユーザー エクスペリエンス
リリース準備合意	テストとリリースマネジメント
展開完了	リリースマネジメント

ポストマイルストンレビュー

それぞれの主要マイルストンは、完了したばかりのフェーズの進捗に関する学習と反省の機会を与えます。ポストマイルストンレビューは、このような反省に適した場を提供します。このレビューは、顧客および他のステークホルダーと共にマイルストンの成果物を評価するために実施されるマイルストン レビュー ミーティングとは目的が異なります。最後のポストマイルストンレビューはプロジェクトの終わりに行います。組織によっては、これをポストモーテムと呼びます。

繰り返し型 (イテレーティブ) アプローチ

繰り返し型 (イテレーティブ) アプローチの特徴

繰り返し型 (イテレーティブ) 開発は、MSF で繰り返し現れるテーマです。コード、ドキュメント、設計、計画、および他の成果物は繰り返し開発されます。

バージョンドリリース

MSF では、中核の機能性を構築、テスト、および展開することでソリューションを開発することを推奨しています。後で機能群を追加していきます。これはバージョンドリリース戦略として知られています。もちろん、小さいプロジェクトの中には 1 つのバージョンしか必要としないものもあります。それでもやはり、ソリューションを複数のバージョンに分割する機会を探すことが推奨されます。図 7 は、多数のバージョンにわたって機能性が発展する様を示しています。

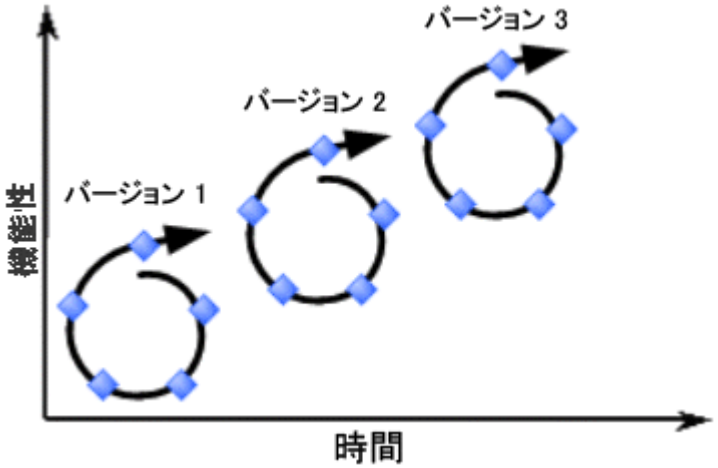

図 7: バージョンドリリースの利用

バージョンドリリースは必ずしも逐次的に発生するわけではありません。成熟したソフトウェア製品は、しばしば、重なり合ったリリース サイクルで作業している複数のバージョン チームによって開発されます。バージョン間の間隔は、プロジェクトの規模やタイプだけでなく、顧客のニーズや戦略によって異なります。

生きたドキュメントの作成

制御を失ってぐるぐる回るのを回避するために、繰り返し型(イテレーティブ)開発では、プロジェクトの変更に応じて変化するドキュメントが必要になります。これらの "生きた ドキュメント" は、すべての要求と仕様が揃って固定されるまで開発を開始しない、ウォーターフォール アプローチとは異なる方法で維持されます。

MSF プロジェクト ドキュメントは、コードのように繰り返し作成されます。計画ドキュメントは多くの場合、おおまかな"アプローチ" として開始されます。これらのドキュメントは、ビジョン化フェーズでチームとステークホルダーがレビューするために回覧されます。プロジェクトが計画フェーズに移行すると、これらは詳細なプランに発展します。これらはまた、繰り返しレビューされて変更されます。これらのプランのタイプと数は、プロジェクトの規模によって異なります。

混乱を避けるために、ビジョン化フェーズで着手される計画ドキュメントを "アプローチ" と呼びます。たとえば、後のフェーズでのテスト計画に発展する簡潔なテスト アプローチをビジョン化フェーズで作成できます。

できるだけ早くベースラインを作成し、できるだけ遅く凍結する

プロセスのできるだけ早い時期にプロジェクト ドキュメントを作成してベースラインにすることで、チーム メンバは過度の計画で被るかもしれない遅延なしに、開発作業を開始することができます。ドキュメントを柔軟なものにして、それに対応するフェーズのできるだけ遅い時期に凍結することによって、開発中の変化に対応することができます。このような柔軟性には、変更制御プロセスに対する慎重な注意が必要です。変更を追跡して、認定されていない変更が行われないことを確実にすることが不可欠です。

デイリービルド

MSF は、テストとレビューのために、ソリューションのすべての要素を頻繁にビルドするよう唱えています。このアプローチはコードの開発はもちろん、ハードウェアおよびソフトウェア要素の "ビルド" も対象にしています。このようなアプローチでは、ソリューションを運用環境にリリースする前に、十分なテスト データによってソリューション全体の安定性を良く知られた状態にできます。

大規模で複雑なプロジェクトは多くの場合、複数の部分に分割され、各部分は別々のサブチームや製品機能チームによって開発およびテストされ、その後で全体に統合されます。Microsoft の製品開発は典型的ですが、このようなプロジェクトでは、"デイリービルド" アプローチはプロセスの基本的な部分の1つです。ソリューションや製品の中核の機能性をまず完成させ、次に付加的な機能を追加します。開発とテストは並行して、継続的かつ同時に行われます。デイリービルドにより、すべてのコードが互換性を検証され、さまざまなサブチームが開発とテストの繰り返し(イテレーション)を継続できます。

これらの繰り返しビルドが実運用環境に展開されないことに注意してください。ビルドが十分にテストされ、安定している場合にのみ、運用環境の一部分に限定されたパイロット (ベータ) リリースとして準備されます。

ビルドの同期を維持するには、厳格な構成管理が不可欠です。

構成管理

構成管理は、さまざまなプロジェクト要素の状態の追跡と制御を定式化したものです。これらの要素には、コード、ドキュメント、ユーザー マニュアルとヘルプファイル、スケジュール、および計画のバージョン管理を含みます。また、ソリューションのハードウェア、ネットワーク、およびソフトウェア設定の状態の追跡も含みます。チームは、必要な場合に、ソリューション全体の以前の構成を再現できる、または以前の構成へ"ロールバック"できなければなりません。

構成管理はしばしば、後述するプロジェクト変更制御と混同さます。両者は相互に関係がありますが、同じではありません。構成管理は、プロジェクト成果物およびドキュメントの状態を追跡することです。変更制御は、変更をレビューして承認するのに使用するプロセスです。構成管理は、チームが変更制御上の決定を効果的に行うのに必要な基礎データを提供します。

たとえば、あるチームが病院チェーン向けの電子医療請求システムに取り組んでいるとしましょう。チームは Microsoft® BizTalk® サーバー上で選択した設定を記録し、開発およびテスト中に行った変更を追跡します。これは構成管理の例です。新しい政府規制に従うために、誰かが新しい EDI 対応スキーマの追加を提案しました。主要なチーム メンバは、提案された変更、その技術的なリスク、およびコストとスケジュールへの影響をレビューするために、プロジェクトに資金を供給する管理者および運用スタッフとの会議を開催します。これは変更管理の例です。

MOF を使用する組織では、運用で使用している構成管理プロセスの多くをプロジェクトの構成管理に適合させることができます。

バージョンドリリースの指針

バージョンドリリースは、チームと顧客の関係を改善し、最良のアイデアがソリューションに反映されることを確実にします。顧客は、チームが初期およびその後のソリューション リリースを適時的に提供すると信じている場合は、機能を後のリリースまで保留することをより受け入れ易いでしょう。バージョンドリリースの採用を促進する指針は以下のとおりです。

- 複数リリースの計画を作成する。
- 核となる機能を最初に提供する。
- 繰り返し（イテレーション）を迅速にまわす。
- 変更制御を確立する。
- 価値が高まらない場合は、新しいバージョンの作成を止める。

複数リリースの計画を作成する

現行のバージョンの範囲を超えて思考することにより、チームは何を今構築し何を延期するかに関する意思決定をより適切に行えるようになります。将来の機能開発のタイムテーブルを提供することで、チームは利用可能な資源とスケジュール制約を最大限に活用できるだけでなく、望ましくないスコープの拡大を防ぐこともできます。

核となる機能を最初に提供する

顧客の手にある基本的で堅牢な使用できるソリューションは、何週間、何か月、または何年も待たないと利用できない豪華なバージョンより直接的な価値があります。最初に核となる機能を提供することで、開発者は構築の強固な基盤を確立し、その後の繰り返しにおける機能開発の推進に役立つ顧客のフィードバックから利益を得られます。

リスク駆動スケジュールを利用した優先順位付け

チームによるリスク査定は、どの製品機能が最もリスクが大きいかを識別します。リスクの詳細については、ホワイト ペーパー「MSF リスクマネジメント規範」を参照してください。最もリスクが大きい製品機能を最初に完了するようにスケジュールしてください。アーキテクチャに対する大きな変更を必要とする問題をプロジェクトの早い時期に処理できるため、スケジュールと予算に対する影響を最小限に抑えることができます。

繰り返し (イテレーション) を迅速にまわす

バージョン分割の重要な便益は、使用可能なソリューションを顧客に暫定的に提供し、それらを増分的に改良することです。このプロセスが失速すると、継続的な製品改良に対する顧客の期待に応えられません。スコープを管理できる状態に維持して、許容期間内に繰り返し(イテレーション)を完成できるようにください。

変更制御を確立する

一旦仕様のベースラインが確立すると、ソリューションのすべての機能が変更制御の下に置かれたと考えるべきです。チーム全体と顧客がこの意味と変更制御プロセスを理解することが不可欠です。

MSF は、特定の変更制御手順を規定しません。プロジェクトの規模や性質よって、その手順は簡単であったり、非常に複雑であったりします。しかしながら、効果的な変更制御には以下の要素が必要です。

- チームと顧客の両方のレビューと承認がないと、製品機能を追加したり変更できません。
- レビューを容易にするために、製品機能の変更要求を書面で提出します。これにより、変更要求のグループを追跡できます。Microsoft では、これは設計変更要求 (DCR; design change request) と呼ばれています。
- 各機能要求の影響、実現可能性、および優先順位を分析します。ユーザーおよび運用ドキュメント、トレーニング資料、運用環境などを含む、他の機能との依存関係を考慮してください。
- 各変更要求のコストとスケジュールに対する影響を見積ります (詳細については、「ボトムアップ見積り」の節を参照) 。
- 変更を承認する変更制御委員会の一員となる個人 (顧客、プログラムマネジメント、およびステークホルダーと他のチーム メンバの組み合わせを含む) を指定します。このようなグループは、コスト、スケジュール、および機能性に対する変更を承認する権限を与えられる限り、様々な形態を取り得ます。
- 変更を追跡し、アクセスしやすいものにします。たとえば、機能仕様書と他の重要なドキュメントの変更履歴の節を保守するのは良い習慣です。
- 効果的な変更管理には、効果的な構成管理が必要です。

開発と展開の統合ビュー

前述したように、ソリューションは、実際に運用環境に完全に展開されて初めてビジネス上の価値を提供します。これが、MSF プロセス モデルがソリューションが価値を提供し始めるまで (すなわち展開が完了するまで)ソリューションの軌道を追跡する理由です。

統合プロセス モデルの利点

アプリケーションの開発と展開を統合するプロセス モデルには以下の利点があります。

エンタープライズ ニーズへの集中

エンタープライズ (特に、ビジネスの意思決定者) は一般に、ソリューションの開発と展開を単一の統合された仕事と理解しています。たとえソリューションを無事に開発しても、エンタープライズに展開するまで、企業の意思決定者は投資に対する回収と見做しません。

伝統的な Web 開発に対するサポート強化

Web 開発チームは今日、計画および調整された一つの活動として、Web サイトを構築および展開(ホスト)します。

Web サービスに対するサポート強化

Web サービスは、ホスティング環境への迅速な展開のために設計および構築する必要があります。Web サービスはソフトウェア提供のチャネルとしてよる頻繁に使用されるようになったため、商用ソフトウェア ベンダですら、展開を製品ライフサイクルの不可欠な一部分と考えることを理解するでしょう。

運用への“壁越し”の引渡しの除去

運用要求を十分に考慮することなしにソリューションを構築することは、多くの開発チームにとっては普通のことです。その結果、アプリケーションはパフォーマンス、可用性、および管理 可能性が貧弱なものになります。MSF の統合プロセス モデルは、1 回の“冷酷な”引渡しではなく、一連の中間マイルストン上で開発チームから運用チームに担当を移行します。

統合プロセス モデルの使用に関する注意

期間の等しくないフェーズ

プロセス モデルの図はフェーズを同じ大きさで示していますが、これは各フェーズで同じくらいの時間がかかることを意味するものではありません。プロジェクトによっては、各フェーズの時間が大幅に異なることがあります。

複数のフェーズにわたる活動

新たに MSF を実践しようとする者は、あるフェーズに関連した活動はそのフェーズのみにおいて実行されると考えるかもしれません。実際はそうではありません。たとえば、計画は計画フェーズだけで起こるのではなく、テストは安定化フェーズ以外でも発生し、開発フェーズ以外で開発が進められる場合もあります。フェーズは目標と成果物によって特徴付けられ、付け加えて、多くの場合にチームが焦点を置く典型的な活動によって、より少ない程度に特徴付けられます。

計画の作成、更新、洗練はプロジェクトを通して継続します。しかし、計画の大部分は計画フェーズで起こり、主要なプランの成果物は計画フェーズで全面的にレビューされます。

純粋なアプリケーション開発とインフラストラクチャ展開プロジェクト

プロジェクトの中には、ソリューションの構築と展開の両方を含まないものがあります。"シュリンクラップ" 製品を構築する商用ソフトウェア ベンダは明らかに、顧客のために構築したものを展開することはありませんが、関連するものを徹底的に理解する必要があります。同様に、インフラストラクチャ展開プロジェクトのチームは、展開するテクノロジを作成しませんが、自動インストール スクリプトの作成など、開発活動を行う必要があります。

純粋なアプリケーション開発や純粋なインフラストラクチャ展開プロジェクトのチームは、そのようなプロジェクトに適用されない参照や中間マイルストンを単純にとばすことができます。

プロセス モデルのフェーズとマイルストン

MSF v.3.0 は、2 つの以前の MSF プロセス モデル、アプリケーション開発 (AD; application development) とインフラストラクチャ展開 (ID; infrastructure deployment) を統合しています。この単一のモデルは、ソリューション開発の開始から完全な展開までに携わります。そのため、以前の 4 フェーズのパターンが 5 フェーズに拡張されています。各フェーズは外部から見えるマイルストンに結実します。図 8 は MSF プロセス モデルのフェーズとマイルストンを示しています。図 8 に示す図の外観は一部の MSF 実践者を驚かせるかもしれませんが、変更は見かけほど大幅なものではありません。これは、元の 2 つのモデルの重要な本質はいずれも失われていないからです。両者の細部は、順に単純に併合されました。MSF v.3.0 で行われた変更の背景と理由については、付録 A を参照してください。

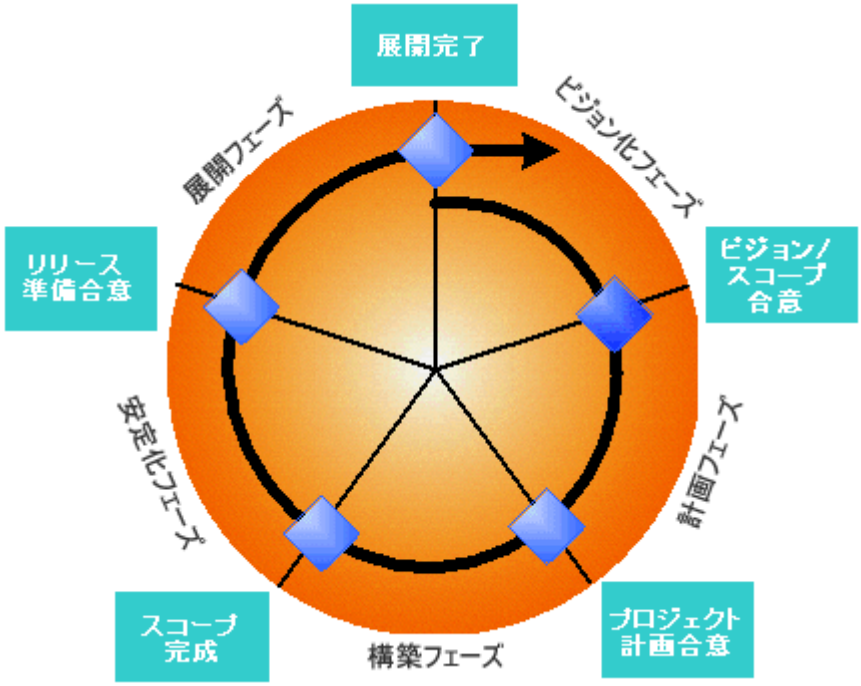

図 8: MSF プロセス モデルのフェーズとマイルストン

ビジョン化フェーズ

概要

ビジョン化フェーズでは、プロジェクトの成功のための最も基本的な要求の 1 つ、すなわち一つの共通のビジョンの元でのプロジェクト チームの統一に取り組みます。チームは、顧客のために達成せんとすることの明確なビジョンを持ち、チーム全体と顧客を動機付けるような表現でそのビジョンを述べることができなければなりません。ビジョン化は、プロジェクトの目標と制約に関する大まかな見方を作成することで、計画の初期の形態として役立ちます。つまり、プロジェクト計画フェーズで行われるより本格的な計画プロセスのお膳立てをします。

ビジョン化において行われる主要な活動は、コア チーム (後述) の編成とビジョンスコープ文書の準備と配布です。プロジェクト ビジョンの叙述とプロジェクト スコープの識別は別個の活動です。プロジェクトの成功にはどちらも必要です。*ビジョン*は、ソリューションの目指す姿に関する境界のない展望です。*スコープ*は、プロジェクトの制約内で達成可能なビジョンの一部を識別します。

リスク管理は、プロジェクトを通じて継続する再帰的プロセスです。ビジョン化フェーズでは、チームはリスク ドキュメントを準備し、ビジョンスコープ文書と共に上位リスクを提示します。詳細については、ホワイト ペーパー「MSF リスクマネジメント規範」を参照してください。

ビジョン化フェーズでは、ビジネスの要求を識別して分析する必要があります。これらの要求は、計画フェーズでより精密に洗練されます。

ビジョン化フェーズを推進する主なチームの役割はプロダクトマネジメントです。

ビジョン/スコープ合意マイルストン

ビジョン/スコープ合意マイルストンはビジョン化フェーズを完結させます。この時点で、プロジェクト チームと顧客はプロジェクトの全体的な方向だけでなく、ソリューションに含む機能と含まない機能、および提供の大まかなタイムテーブルに合意します。

成果物

ビジョン化フェーズの成果物は以下のとおりです。

- ビジョンスコープ文書
- リスク査定文書
- プロジェクト構造文書

ビジョン化フェーズにおけるチームの焦点

以下の表では、ビジョン化フェーズにおけるチームの各々の役割の焦点と責務について説明します。

役割	焦点
プロダクトマネジメント	全体的な目標; 顧客ニーズ、要求の識別; ビジョンスコープ文書
プログラムマネジメント	設計目標;ソリューション コンセプト;プロジェクト構造
開発	プロトタイプ;開発とテクノロジの選択肢;実現可能性の分析
ユーザーエクスペリエンス	ユーザーの実効性についてのニーズと考慮点
テスト	テスト方針;テストの受け入れ基準;考慮点
リリースマネジメント	展開の考慮点;運用管理とサポート性;運用の受け入れ基準

中間マイルストンの提案

コアチーム形成

この時点で、プロジェクトに主要なチーム メンバが任命されます。通常、完全なチームはまだ召集されていません。すべてのメンバが配置されるまで、初期のチームはたいてい複数の役割を果たします。

プロジェクト構造ドキュメントには、チームがどのように編成されているか、誰がどの役割を果たし、特定の責任を有するかといった情報が含まれています。また、プロジェクト構造ドキュメントでは、顧客に対する説明責任の系統、顧客との指定連絡先を明確にします。これらの情報はプロジェクトの状況に応じて異なります。

ビジョンスコープベースライン

この中間マイルストンでは、ビジョンスコープ文書の初稿を完成し、レビューのためにチーム内、顧客、およびステークホルダーに回覧します。レビュー サイクルでは、ドキュメントに対するフィードバック、検討、および変更が繰り返されます。

計画フェーズ

概要

計画フェーズでは、プロジェクトの計画の大部分を完成させます。このフェーズでは、チームは機能仕様書を準備し、設計プロセスを通して作業し、さまざまな成果物に対する作業計画、費用見積もり、およびスケジュールを用意します。

計画フェーズの初期に、チームは要求を分析してリストまたはツール上で文書化します。要求は 4 つの大きなカテゴリ、すなわちビジネス要求、ユーザー要求、運用要求、およびシステム要求 (ソリューション自体の要求) に分類されます。チームがソリューションを設計し、機能仕様書を作成するようになるので、要求と製品機能の間の*追跡可能性*を維持することが重要です。追跡可能性が 1 対 1 である必要はありません。追跡可能性の維持は、設計の正確性をチェックし、設計がソリューションの目標と要求を満たすかを確認する方法として役に立ちます。

設計プロセスは、抽象的な概念からはじめて具体的な技術的詳細までに落とすための系統立った方法をチームに提供します。これは、さまざまな種類のユーザーとその職能について述べる*ユーザー プロファイル* (“登場人物”とも呼ばれます) の系統的分析から始まります(例えば運用スタッフもユーザーです)。この作業の大部分は、しばしば、ビジョン化フェーズで行います。これらは一連の*利用シナリオ*に分解されます。利用シナリオでは、ホテルのフロントにおける受付登録やシステム管理者向けのユーザー パスワード管理など、特定の種類のユーザーは一種の活動を完了しようと試みます。最終的に、それぞれの利用シナリオは、*ユースケース*という、ユーザーが当該の活動を完了するために実行する特定の一続きのタスクに分割されます。これは "ストーリーボーディング" と呼ばれています。

設計プロセスには、概念設計、論理設計、および物理設計という 3 つのレベルがあります。各レベルは時間的にずれて順に仕上げられ、ベースライン化されます。

設計プロセスの結果は、*機能仕様書*に文書化されます。機能仕様書には、各製品機能がどのように見え、振る舞うかを詳細に記述します。また、すべての製品機能に対するアーキテクチャと設計についても述べます。

機能仕様書は以下のような複数の目的に役立ちます。

- 何を構築するかに関する開発者への指示
- 作業を見積るための基礎
- まさしく何を構築するかに関する顧客との合意
- チーム全体の同期ポイント

機能仕様をベースライン化すると、詳細な計画を開始できます。各チーム リーダーは、その役割に関連する成果物の計画を準備し、チームの計画活動に参加します。そのような計画の例には、展開計画 、テスト計画、運用計画、セキュリティ計画、教育計画などがあります。グループとして、チームは計画をレビューし、計画間の依存関係を識別します。

すべての計画は、Microsoft® Project®の .mpp ファイルと混同されないように、マスタープロジェクト計画として同期され、一緒に示されます。マスタープロジェクト計画に含まれる補助的な計画の数と種類は、プロジェクトのスコープと種類によって異なります。

各役割を代表するチーム メンバは、成果物に対する時間見積りとスケジュールを作成します (詳細については、「ボトムアップ見積もり」の節を参照) 。次に、さまざまなスケジュールをマスタープロジェクトスケジュールに同期、統合します。

計画フェーズの頂点 (プロジェクト計画合意マイルストン) で、顧客とチーム メンバは何が何時提供されるかについて詳細に合意します。プロジェクト計画合意マイルストンでは、チームはリスクを再査定し、優先順位を更新して、資源とスケジュールの見積もりを完成させます。

プロジェクト計画合意

プロジェクト計画合意マイルストンでは、プロジェクト チームと主要なプロジェクト ステークホルダーは、中間マイルストンが達成されていること、納入期日が現実的であること、プロジェクトの役割と責任が十分に定義されていること、プロジェクト リスクに対処する仕組みができていることに合意します。機能仕様書、マスタープロジェクト計画、およびマスタープロジェクトスケジュールは将来のトレードオフを決定する基礎になります。

チームが仕様、計画、およびスケジュールを承認すると、ドキュメントはプロジェクトのベースラインになります。ベースラインは 3 つのプロジェクト計画変数、すなわち資源、スケジュール、および製品機能を適用することで、合意に達しているさまざまな決定を考慮に入れます。ベースラインが完了して承認されると、チームは構築フェーズに移行します。

チームがベースラインを定義すると、それは変更制御の下に置かれます。ただし、計画フェーズで達したすべての決定が最終的なものであるというわけではありません。作業が構築フェーズに進むと、チームはベースラインに対するあらゆる変更の提案をレビューして承認する必要があるということです。

組織が MOF を使用している場合、チームはこのマイルストンで IT 運用に対する変更要求 (RFC; Request for Change) を提出します。

成果物

計画フェーズでは、以下の成果物が作成されます。

- 機能仕様書
- リスク管理計画
- マスタープロジェクト計画とマスタープロジェクトスケジュール

計画フェーズにおけるチームの焦点

以下のテーブルでは、計画フェーズにおけるチームの各々の役割の焦点と責務について説明します。

役割	焦点
プロダクトマネジメント	概念設計;ビジネス要求分析;広報計画
プログラムマネジメント	概念および論理設計;機能仕様書;マスタープロジェクト計画とマスタープロジェクトスケジュール;予算
開発	テクノロジ評価;論理および物理設計;開発計画/スケジュール;開発見積もり
ユーザー エクスペリエンス	利用シナリオ/ユースケース;ユーザー要求;ローカライズ/アクセシビリティ要求;ユーザー ドキュメント/教育計画/使用性テスト、ユーザー ドキュメント、トレーニングのスケジュール
テスト	設計評価;テスト要求;テスト計画/スケジュール
リリースマネジメント	設計評価;運用要求;パイロットおよび展開計画 /スケジュール

中間マイルストンの提案

テクノロジ妥当性確認完了

テクノロジ妥当性確認時に、チームはソリューションの構築や展開に使用する見込みの製品やテクノロジを評価し、ベンダの仕様どおり機能するか確認します。これは後に概念検証(プルーフオブコンセプト)や、最終的にはソリューションそのものの開発を引き起こす一連の活動の、最初の繰り返し(イテレーション)です。

しばしば、テクノロジ妥当性確認には、競合するテクノロジや供給者間の競争的評価 ("撃ち合い" とも呼ばれる) が伴います。

このマイルストンで完了しなければならないもう 1 つの活動は、顧客環境のベースライン化です。チームは、ソリューションが運用されることになる運用環境の現状に関する監査 ("発見" とも呼ばれる) を実施します。これには、サーバー構成、ネットワーク、デスクトップ ソフトウェア、およびすべての関連ハードウェアが含まれます。

機能仕様ベースライン

このマイルストンでは、顧客とステークホルダーのレビューに十分な程度に機能仕様書が仕上げられます。この時点で、チームは仕様をベースラインとし、正式に変更の追跡を開始します。

機能仕様書は、マスタープロジェクト計画とスケジュールを構築する基礎になります。機能仕様書は、ユーザーの観点から見て、ソリューションの外観や振る舞いに関する詳細な記述として維持されます。機能仕様書を変更できるのは、顧客の承認がある場合だけです。

しばしば設計プロセスの結果は、機能仕様書とは別の設計ドキュメントに文書化されます。設計ドキュメントは、ソリューションの内部作用の記述に重点を置いています。設計ドキュメントはチームの内部に留めておき、技術的な問題で顧客を悩ませることなしに変更できます。

マスタープロジェクト計画ベースライン

MSF では、マスタープロジェクト計画は、さまざまな役割が提供する計画の集合です。それ自身で独立した計画ではありません。プロジェクトの種類や規模によって、さまざまなプランがマスタープロジェクト計画に併合されます。これらの計画の一部を図 9 に示します。

図 9: マスタープロジェクト計画

計画をより小さい計画で構成する利点は、さまざまなチームの役割による計画立案作業の同時並行が容易になることと、各役割が特定の計画の責任を負うため、説明責任を明確にできることです。

これらの計画を 1 つのものとして提示する利点は、単一のスケジュールへの同期が容易になり、レビューと承認が促進され、ずれや不整合の識別に役立つことです。

マスタープロジェクトスケジュールベースライン

マスタープロジェクトスケジュールには、リリース日を含む、詳細なプロジェクト スケジュールがすべて含まれています。マスタープロジェクト計画と同様に、マスタープロジェクトスケジュールには、各チーム リーダーからのすべてのスケジュールが結合、統合されます。チームは機能仕様書の草案を協議し、マスタープロジェクト計画の草案をレビューした後に、リリース日を決定します。しばしば、チームは要求されたリリース日に合わせるために、機能仕様書やマスタープロジェクト計画の一部を修正します。製品機能、資源、およびリリース日は変わる可能性がありますが、固定されたリリース日は、チームが製品機能に優先順位をつけたり、リスクを査定したり、適切な計画を立てる動機になります。

開発/テスト環境設定

実用的な開発環境は、ソリューションの適切な開発とテストが可能であり、運用システムに悪影響を与えません。一般に、開発者が使用できる別個の開発用サーバーをセットアップするのが得策です。そのようなサーバー上の何かが不安定になったり、再インストールが必要な場合には、チーム全体に知らせる必要があります。

これはまた、サーバー構成、自動展開ツール、およびハードウェアなどのインフラストラクチャ要素を開発する環境でもあります。

遅延を避けるため、計画を完成してレビューすると同時に、開発およびテスト環境をセットアップする必要があります。これには開発用ワークステーション、サーバー、およびツールが含まれます。既存のものが無ければ、バックアップ システムも確立する必要があります。マシンを頻繁に“大掃除”したり、再フォーマットする場合は、通常、標準サーバー構成の CD-ROM イメージを使用します。

適切なテスト ラボが組織にまだ存在しない場合、チームはそれを構築する必要があります。テスト環境は、できる限り実際の環境に近い模写でなければなりません。これには費用がかかるかもしれませんが、非常に重要です。そうしないと、ソリューションを "実際の" 運用環境に展開するまで、ある種のバグが検知されない恐れもあります。MOF を使用する組織では、運用環境を複製するための一種の部品表として、エンタープライズの構成管理データベース (CMDB) に格納されている情報を活用できます。

構築フェーズ

概要

構築フェーズでは、チームはソリューションの構成要素 (コードだけでなく文書も含む) の構築の大部分を行います。ただし、いくらかの開発作業は、テストに応じて安定化フェーズでも続行されます。

構築フェーズに関与するのは、コード開発やソフトウェア開発者だけではありません。インフラストラクチャもこのフェーズで開発され、すべての役割が成果物の作成とテストで活動的な状態です。

スコープ完成マイルストン

構築フェーズは、スコープ完成マイルストンで終わります。このマイルストンでは、すべての製品機能が仕上げられ、ソリューションは外部テストと安定化のために準備できています。このマイルストンは、顧客とユーザー、運用とサポート スタッフ、および主要なプロジェクト ステークホルダーがソリューションを評価し、ソリューションをリリースする前に対処しなければならない残された問題を識別する機会です。

成果物

構築フェーズの成果物は以下のとおりです。

- ソース コードと実行可能ファイル
- インストール スクリプトおよび展開用の構成設定
- 凍結された機能仕様書
- パフォーマンス サポート要素
- テスト仕様とテスト ケース

構築フェーズにおけるチームの焦点

以下の表では、構築フェーズにおけるチームの各々の役割の焦点と責務について説明します。

役割	焦点
プロダクトマネジメント	顧客の期待管理
プログラムマネジメント	機能仕様書管理;プロジェクト追跡;計画の更新
開発	コード開発;インフラストラクチャ開発;構成ドキュメント
ユーザー エクスペリエンス	トレーニング;教育計画の更新;使用性テスト;グラフィック デザイン
テスト	機能テスト;問題の識別;ドキュメント テスト;テスト計画の更新
リリースマネジメント	展開チェックリスト;展開およびパイロット計画の更新;拠点準備チェックリスト

中間マイルストンの提案

概念検証完了

概念検証は、実際には運用しない既存環境のシミュレーション上でソリューションの主要な要素をテストします。チームは運用スタッフとユーザーとしてソリューションをウォークスルーし、それらの要求を検証します。

内部リリース n、内部リリース n+1

構築フェーズではソリューションの構築に重点を置いているため、プロジェクトには、チームがビルドの進捗を測定するのに役立つ中間マイルストンが必要です。

開発は区分ごとに並行して行われるため、チームは全体としての進捗を測定する方法を必要とします。内部リリースは、チームにソリューション レベルでの部分の同期を強要することによりこれを実現します。

内部リリースの数と頻度は、プロジェクトの規模と期間によります。

多くの場合、ビジュアル デザインやデータベースを凍結するために中間マイルストンを設定することは、これらに対する多くの依存関係があるために意味があります。たとえば、ドキュメントを作成するのに必要な画面や、全体的なアーキテクチャの深部を形成するデータベース スキーマなどです。

安定化フェーズ

概要

安定化フェーズでは、製品機能の完成したソリューションのテストを実施します。このフェーズにおけるテストでは、現実的な環境条件下での使用や運用が強調されます。チームはバグの解決とトリアージ(優先順位を付け)、リリースに向けてソリューションを準備することに重点を置きます。

このフェーズの初期には、開発者がバグを修正するよりも速いペースで、テストによってバグが報告されるのが一般的です。いくつのバグがあるか、それらを修正するにはどのくらいかかるかを正確に知る方法はありません。しかし、バグ収束およびゼロバグ バウンスと呼ばれる2つの統計的指標は、ソリューションがいつ安定するかをチームが推定するのに役立ちます。これらの指標について以下に説明します。

MSF では、IT プロジェクトの状態を説明するのに、"アルファ" とか "ベータ" という用語を使用しません。これらの用語は広く使用されていますが、あまりにもいろいろな意味で解釈されているため、業界では意味のあるものではありません。これらの用語を明確に定義し、チーム、顧客、およびステークホルダーがその定義を理解してさえいれば、チームは必要に応じて使用することができます。

ビルドがリリース候補として十分に安定していると判断した場合は、そのソリューションをパイロットグループに展開します。

安定化フェーズは、リリース準備合意マイルストンで終わります。レビューと承認が済むと、ソリューションは実運用環境に完全に展開する準備ができた状態になります。

リリース準備合意マイルストン

リリース準備合意マイルストンは、チームがすべての未解決の問題に対処し、ソリューションをリリースまたは供用する準備ができた時点で発生します。リリース準備合意マイルストンでは、顧客および主要ステークホルダーに加え、運用チームの承認が必要です。

成果物

安定化フェーズの成果物は以下のとおりです。

- ゴールデン リリース
- リリース ノート
- パフォーマンス サポート要素
- テスト結果とテスト ツール
- ソース コードと実行可能ファイル
- プロジェクト ドキュメント
- マイルストン レビュー

安定化フェーズにおけるチームの焦点

以下の表では、安定化フェーズにおけるチームの各々の役割の焦点と責務について説明します。

役割	焦点
プロダクトマネジメント	**広報計画の実行;発売（稼動開始）計画**
プログラムマネジメント	**プロジェクト追跡;バグ トリアージ**
開発	**バグの解決;コードの最適化**
ユーザー エクスペリエンス	**ユーザー パフォーマンス資料の安定化;トレーニング資料**
テスト	**テスト実施;バグ報告と状態管理;構成テスト**
リリースマネジメント	**パイロットのセットアップとサポート;展開計画 ;運用およびサポート トレーニング**

中間マイルストンの提案

バグ収束

バグ収束は、チームが未解決のバグ数を目に見えて改善するポイントです。すなわち、バグ解決の割合がバグ発見の割合を超えます。図 10 は、バグ収束を示したものです。

全体的に減少し始めた後でも、バグの割合は上下するため、バグ収束は通常、固定した時点ではなく傾向として現れます。バグ収束の後、バグの数はゼロバグ リリースまで引き続き減少します。バグ収束は、終わりが実際に近づいていることをチームに伝えます。

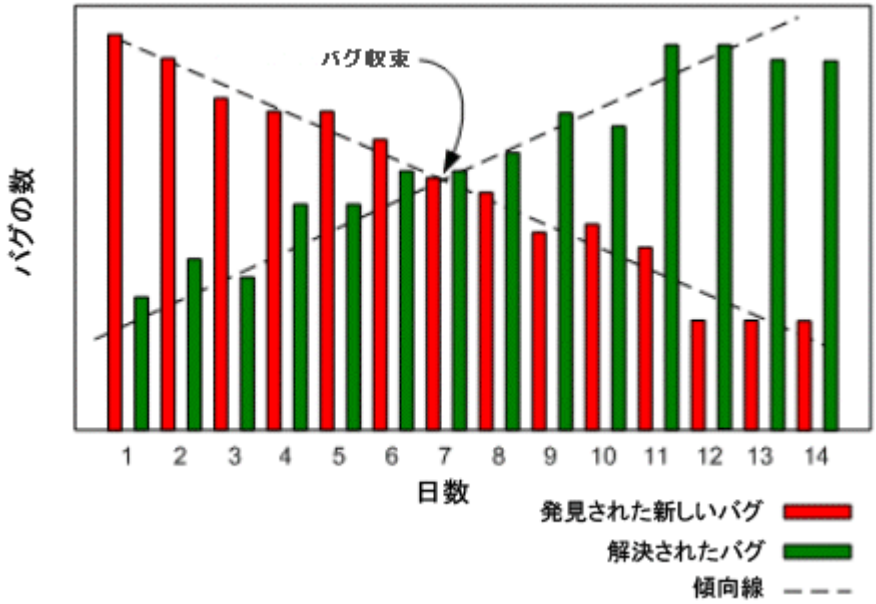

図 10: バグ収束

ゼロバグバウンス

ゼロバグバウンスは、プロジェクトにおいて開発がついにテストに追いつき、差し当たり未解決のバグがなくなったポイントです。図 11 は、ゼロバグバウンスを示したものです。ゼロバグバウンスの後、チームが最初のリリース候補をビルドするのに十分な程度にソリューションが安定するまで、バグのピークは著しく小さくなり、減少し続けるはずです。

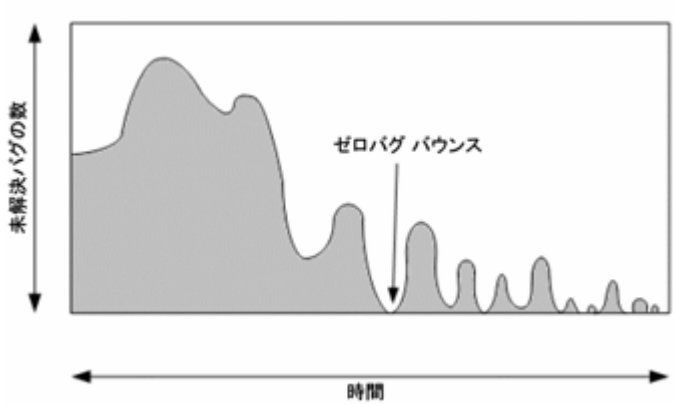

図 11: ゼロバグバウンス

バグを修正するたびに新しいバグを生み出す危険にさらされるため、慎重なバグのトリアージはきわめて重要です。ゼロバグバウンスへの到達は、チームが安定したリリース候補へ向かう最終段階にあることを明確に示すものです。

このマイルストンに達した後でも、間違いなく新しいバグが見つかることに注意してください。それでもなお、ゼロバグバウンスはチームが未解決のバグが存在しない（たとえ束の間であっても）と正直に報告できる初めての時であり、その状態に留まるようにチームを集中させます。

リリース候補

一連のリリース候補を用意して、パイロット グループに引き渡します。各リリース候補は中間マイルストンです。リリース候補のその他の特徴は以下のとおりです。

- 各リリース候補には、運用環境にリリースするのに必要な要素がすべてあります。
- リリース候補をビルドすることでリリースへの適合性、すなわちすべての必要な部分が揃っているかどうか、がテストされます。
- リリース候補の作成に続くテスト期間により、リリース候補を運用環境にリリースできる状態にあるか、あるいはチームが適切な修正をして新しいリリース候補を作成しなければならないかが決まります。
- リリース候補のテストは、チームが内部的に行いますが、非常に焦点を絞った徹底的な取り組みが要求され、プロジェクトの進捗を中断させるほどの重大なバグを洗い出すことに集中します。
- テストには、新たに発見したバグの解決のためのトリアージ プロセスが必要です。
- 最初のリリース候補がリリースされるものになることは、あまりありません。典型的には、プロジェクトの進捗を中断させるほどの重大なバグはリリース候補を徹底的にテストしているうちに見つかります。

運用テスト完了

この中間マイルストンは、パイロット リリースを用意することに的を絞ります。ソリューションが実運用環境にまさに"触れ"ようとしているため、この中間マイルストンは重要です。そのため、チームはパイロット テストが始まる前に、ソリューション全体のできるだけ多くをテストする必要があります。

この中間マイルストンの間に完了しなければならない活動は以下のとおりです。

- 成功基準に対してテスト結果を評価します。
- 拠点準備チェックリストと手順を完了します。
- 実装手順、スクリプト、および積荷を完了します。
- トレーニング資料を完了します。
- サポートの問題を解決します。
- ロールバック計画を完了し、テストします。

ソリューションを展開するために開発したすべてのものを完全にテストして準備ができたことをチームが確認するまで、運用テスト完了中間マイルストンは完了しません。

ユーザー受入れテスト完了

ユーザー受入れテストと使用性の検討は、構築フェーズの間に始まり、安定化フェーズでも継続します。これらは、新しいシステムがユーザーおよびビジネス ニーズを十分に満たせることを確実にするために行われます。これをプロジェクトの終わりに行われる*顧客受入れ*と混同しないよう注意してください。

ユーザーが*非実運用環境*でリリースをテストして受け入れ、システムが既存のビジネス アプリケーションや IT 運用環境と統合されることを確認すると、このマイルストンが達成されます。また、展開および切り戻し手順も、この期間中に確認する必要があります。

リリースマネジメントが承認すると、社内で開発したソフトウェアおよびすべての購入したアプリケーションは、厳重に管理された保管庫からきれいな保管場所に移されます。MOF では、これは確定版ソフトウェアの保管庫(DSL; Definitive Software Library) と呼ばれています。リリースマネジメントは DSL に格納されたアプリケーションから、テスト環境でリリースをビルドする (リリース要素を組み立てる) 責務を担っています。

ユーザー受け入れテストはサポート スタッフとユーザーに、実地トレーニングを通じて新しいテクノロジを理解し練習する機会を提供します。このプロセスは、ユーザーがソリューションの理解、習得、使用に困難を抱える領域を識別するのに役立ちます。また、リリース テストはリリースマネジメントに、実装の成功を阻害する恐れのある問題を識別する機会を与えます。

試験運用完了

この中間マイルストンでは、チームはできるだけ実際の運用環境で、ソリューション全体のできるだけ多くをテストします。MSF では、パイロット (試験運用) リリースは実運用環境またはユーザーグループの部分集合への展開です。プロジェクトの状況によって、パイロット リリースは以下の形態を取り得ます。

- エンタープライズでは、パイロットはユーザーの一部のグループまたはデータ センターの一部のサーバー セットを対象とできます。
- Web 開発では、パイロット リリースは、ステージング サーバーへのサイト ファイルのホスティング、あるいは実際のインターネット上のテスト用の Web アドレスだけを持つフォルダの形態をとります。
- Microsoft などの商用ソフトウェア ベンダは、多くの場合、最終的なリリースの前に製品を初期採用者の特別なグループにリリースします。

これらの形態のパイロットすべてに共通しているのは、それらが実際の条件下におけるテストの実例であるということです。

提案されたソリューションが運用環境で実行可能であり、ソリューションの全ての構成要素が展開の準備ができていることをチームが確認するまで、試験運用完了の中間マイルストンは完了しません。さらに、以下の措置を講じる必要があります。

- パイロットを開始する前に、チームとパイロット参加者はパイロットの成功基準を明確に識別して、合意しなければなりません。これらは開発活動の成功基準に戻って対応している必要があります。
- パイロット中に識別された問題は、さらなる開発か、インストール チームと運用サポート スタッフ向けの解決策や回避策の文書化か、あるいはトレーニング コースに補足資料として組み込むことによって解決しなければなりません。
- パイロットを開始する前に、サポート構造と問題解決プロセスを機能させる必要があります。そのためには、サポート スタッフを訓練する必要があるかもしれません。パイロット中に問題解決に使用した手続きは、展開中およびソリューションが完全に稼動している時に使用するものとは、大きく異なるかもしれません。
- 展開プロセスが機能するかどうかを判断するには、実際に展開する前に問題を識別できるように、展開の全要素を試運転またはリハーサルする必要があります。

十分なパイロット データを収集して評価すると、チームは決定を下すことができます。この時点では、以下の方策のいずれかを選択する必要があります。

- やり直し - 新しいリリースをパイロット グループに展開します。
- ロールバック - ロールバック計画を実行し、パイロット グループをパイロット以前の構成状態に (できるかぎり厳密に) 戻します。パイロットは、より安定したリリースで再度施行されます。
- 一時停止 - パイロット全体を一時停止します。
- パッチおよび続行 - パイロット グループに "パッチ" を発行して、既存コードを修正します。
- 展開フェーズに進みます。

展開フェーズ

概要

このフェーズでは、チームは中核技術と拠点コンポーネントを展開し、展開を安定化し、プロジェクトを運用とサポートに移管し、プロジェクトに関する最終的な顧客承認を獲得します。展開後、チームはプロジェクト レビューと顧客満足度調査を実施します。

プロジェクトの構成要素をテスト環境から運用環境へ移行するのに伴い、安定化活動はこの期間も続く可能性があります。

展開完了マイルストン

展開完了マイルストンで展開フェーズは終わります。この時までに、展開されたソリューションは期待されたビジネス上の価値を顧客に提供しており、また、チームはこの目標に達するために使用したプロセスと活動を効果的に終了していなければなりません。

チームがその目的を達成したことに顧客が同意しなければ、ソリューションの実稼動とプロジェクトの終結を宣言することができません。そのためには安定したソリューションはもちろん、明記された成功基準も必要とされます。ソリューションが安定していると判断されるには、適切な運用およびサポート システムが機能している必要があります。

成果物

成果物は以下を含みます。

- 運用およびサポート情報システム
- 手続きおよびプロセス
- ナレッジ ベース、レポート、ログ
- 全バージョンのドキュメントを格納するドキュメント リポジトリ、積荷、プロジェクトで開発したコード
- プロジェクト終結レポート
- 全プロジェクト ドキュメントの最終バージョン
- 顧客/ユーザー満足度データ
- 次のステップに関する記述

展開フェーズにおけるチームの焦点

以下の表では、展開フェーズにおけるチームの各々の役割の焦点と責務について説明します。

役割	焦点
プロダクトマネジメント	顧客からのフィードバック、評価、署名
プログラムマネジメント	ソリューション/スコープの比較;安定化管理
開発	問題解決;エスカレーション サポート
ユーザー エクスペリエンス	トレーニング;トレーニング スケジュール管理
テスト	パフォーマンステスト;問題
リリースマネジメント	拠点展開管理;変更の承認

中間マイルストンの提案

中核技術展開完了

ほとんどのインフラストラクチャ ソリューションには、ソリューション全体の枠組みや背骨を提供する多数のコンポーネントが含まれています。これらのコンポーネントは、特定のユーザー集合や特定の拠点の観点からすると、ソリューションとは考えられません。しかし、拠点やユーザーの展開は一般に、この枠組みに依存します。さらに、

- これらのコンポーネントはエンタープライズ ソリューションを可能にするテクノロジです。たとえば、ドメイン コントローラ、メール ルーター、リモート アクセス サーバー、データベース サーバーなどです。
- 拠点展開はこのテクノロジに依存しています。
- ソリューションによっては、中核技術は拠点展開の前に、あるいは並行して展開する必要があります。
- 遅延を避けるため、ソリューションの他の部分の安定化の最中に、中核コンポーネントを前もってレビューして展開を承認することもできます。ただし、ソリューション全体が安定していると証明される前にこの確約を行うことについて、運用スタッフが確信を持っていなければなりません。

拠点展開完了

このマイルストンを完了すると、すべての対象ユーザーがソリューションにアクセスできます。また、いくつかの問題があるかもしれませんが、各拠点の責任者はその拠点が運用中であることについて署名済みです。

顧客とユーザーのフィードバックにより、いくつかの問題が明らかになるかもしれません。トレーニングがうまくいかなかったのかもしれませんし、チームがその拠点を離れた後でソリューションの一部が誤動作したのかもしれません。拠点満足度調査からのフィードバックに基づいて、いくつかの拠点に再訪する必要があるかもしれません。

この時点で、チームは展開活動を終了し、プロジェクトを終結するために集中します。

多くのプロジェクト、特に Web 開発では、クライアント側の展開は含まないため、このマイルストンは適用されません。

展開安定化

展開安定化中間マイルストンでは、顧客とチームは各拠点が満足に運用していることに合意します。しかし、さまざまな拠点展開においていくらかの問題が発生することを予期します。これらの問題は引き続き追跡して、解決します。

いつ展開が“完了”して、チームを解放するかを判断するのが難しい場合もあります。新たに展開したシステムはしばしば、一定の流動的な状態にあり、運用サポートの問題を識別し管理する継続的なプロセスを伴います。チームは展開後に表面化する進行中の問題のために、プロジェクトを終結するのが難しいと判断することもあります。このため、チームは絶対的な最終ポイントへの到達を試みるよりも、展開の完了マイルストンを明確に定義する必要があります。

プロジェクトチームのメンバが継続的なメンテナンスとサポートに関与することを顧客が期待している場合、その要員はプロジェクト終結後に、運用とサポート構造の一部として新しい役割に移行する必要があります。

このような遅い段階になると、チーム メンバや外部のステークホルダーはプロジェクトから移り始めるでしょう。

プロジェクトからの退却の一部には、運用とサポート機能を専門スタッフに移行することも含まれています。多くの場合、新しいシステムを管理する資源は既に存在します。そうではない場合には、新しいサポートシステムを設計する必要があるかもしれません。後者の場合のスコープを考えると、別のプロジェクトと考えるほうが賢明です。

展開安定化と展開完了マイルストンとの間の期間は、“静寂期間”と呼ばれることもあります。チームはもはや活動的ではありませんが、チームの要員はエスカレーションされてきた問題に対応します。典型的な静寂期間は 15 ～ 30 日間です。

静寂期間の目的は、ソリューションが通常の運用でどの程度うまく機能するかを測定し、ソリューションを実行するのにどの程度のメンテナンスが必要であるかを理解するベースラインを確立することです。MOF を使用する組織はインシデントの件数やダウンタイムの長さを測定し、ソリューションのパフォーマンス メトリクスを収集します。このデータは、運用のサービスレベル・アグリーメント (SLA ; Service Level Agreement) で使用する、サービスとパフォーマンスの年間期待レベルに関する仮定を形成するのに役立ちます。SLA の詳細については、「MOF Operations Guide for Service Level Management」を参照してください。

MSF プロセス モデルの推奨プラクティス

以下の支援プラクティスは、チームが MSF プロセス モデルをプロジェクトに適用するのに役立ちます。

製品機能の発展と資源の制約により創造性に集中する

Microsoft の一般的な開発アプローチでは、開発要員と予算を制限することにより、創造性に集中し、意思決定を推し進め、リリース日を最適化します。

固定的な納期を確立する

内部的な期限 ("タイムボクシング" と呼ばれる手法) は、プロジェクト チームに製品機能と活動を優先順位付けするようにプレッシャーを与え続けます。

不確実な将来に備えてスケジュールする

チームが予期せぬ問題や変化に対応できるようにするには、プロジェクトのスケジュールに緩衝 (付加) 期間を加えます。適用する緩衝期間の量はリスクの量によります。プロジェクトの初期にリスクを査定することにより、最尤リスクのスケジュールに対する影響を評価し、プロジェクトスケジュールに緩衝期間を追加して補償することができます。

緩衝期間は、未知のタスクや事象の見積もりと考えることもできます。チームがどんなに経験豊富であっても、プロジェクトのあらゆるタスクをあらかじめ認識し、見積もることはできません。しかも、なんらかのプロジェクト リスクが発生し、プロジェクトに影響を与えることは間違いありません。このようなリスクに対応するのに必要な是正措置は時間がかかります。

緩衝期間を使用する推奨ガイドラインは以下のとおりです。

- 緩衝期間は、個々のタスクの見積もりを水増しして追加しないでください。作業は予定した時間いっぱいになるように拡大するため (パーキンソンの法則)、計画されていない出来事ではなく計画したタスクによって緩衝期間が消化されてしまいます。
- 緩衝期間は、別のタスクのように予定する必要があります。通常、緩衝期間は主要マイルストン、特に後の方のものの直前に割り当てます。プロジェクトのクリティカル パス上には必ず配置すべきです。クリティカル パスはプロジェクトに従属するタスクの最長の連鎖であり、プロジェクトの期間を直接決定します。
- 緩衝期間はプロジェクトの過程で消費やされるため、残量を慎重に追跡して大事に使う必要があります。
- 製品機能を追加したり、プロジェクトから要員を抜く場合には、緩衝期間を使用して補償しないでください。そうすると、リスクを補償する能力がその分減少します。図 5 に示すトレードオフ トライアングルを使用して、製品機能、資源、およびスケジュールについて協議してください。
- 緩衝期間を全部使いきってしまった場合は、今やいかなる混乱や遅延も (ラグビーの) “ノックオン”のような効果を有し、終了日を危険にさらすことになることをチーム全体に認知させてください。

頻繁に同期しながら並行作業する、小さいチームを使う

たとえ大規模で複雑なプロジェクトであっても、チームが並行して作業を進め、その活動と成果物を定期的に同期させれば、より小さく効率的なチームに分割できます。これによりプロジェクト全体にわたって一貫した品質に対する焦点を維持でき、プログラムマネージャが全体的な進捗状況のチャートを作成するのに役立ち、チーム内における各自の説明責任が強調されます。

大規模なプロジェクトを管理できる部分に分割する

Microsoft における基本的な開発戦略は、大規模なプロジェクトを複数のバージョンドリリースに分割し、別個のメンテナンス フェーズをほとんどあるいは全く設けません。

非難のないマイルストン レビューを行う

それぞれの主要なマイルストンでは、チーム、顧客、および主要なステークホルダーが集まり、そのマイルストンの成果物をレビューし、プロジェクトの全体的な進捗状況を評価します。大規模なプロジェクトでは、選択した中間マイルストンでも同じことを行います。

このようなミーティングの後で、チームは内部レビューを行い、チームのプロジェクト実績を評価します。このレビューは、プロジェクトの実施方法の変化の引き金となる、品質保証活動の一環と考えるべきです。

個々のチーム メンバの構成は、プロジェクトの途中でしばしば変更されます。主要マイルストンでは、異動するチーム メンバが離脱する前に、その情報や習得したことを必ず取得してください。

プロトタイピングを使う

プロトタイプにより、多くの観点、特に使用性、から開発前にテストを実施することができ、ユーザーの相互作用をより良く理解するのにも役立ちます。また、製品仕様の改善にもつながります。

頻繁なビルドと即時テストを使う

ソリューションの規則的なビルドは、プロジェクトの開発が順調に進んでおり、チームが十分に機能していることを示す最も信頼性の高い指標です。展開フェーズでは、パイロット テスト サイクルは同様の目的に役立ちます。

迅速に回す

エンタープライズ ソリューションでは、ビジネスの俊敏性を強調する必要があります。そのためには、顧客 ニーズの絶え間ない変化に対応しなければなりません。迅速な開発と展開サイクルによって、バージョンドリリースの作成が容易になります。バージョンドリリースにより、変化するニーズと要求に対応してソリューションを進化させることが可能になります。

スコープクリープを回避する

述べられたビジネス目標に対する焦点を維持し、重要な機能を元の要求まで追跡するために、ビジョン記述と仕様を使用してください。プロジェクトが定義された後で、適切な考慮なしに追加された恐れのある付加的な製品機能を識別、検討、除去するために、ビジョン記述と仕様をフィルタとして適用します。

ボトムアップ見積もり

IT プロジェクトの見積もりは、その作業をする人々によって行われるべきです。ボトムアップ見積もりには、以下のような利点があります。

*正確さの向上。*実際に作業を行う人が作成する見積もりは、同様の作業を実行した経験に基づいているので、正確です。

*説明責任。*自身の作業見積もりを作成する人は、その作業に、より責任を感じます。また、作成した見積りにうまくあわせることにも一層の責任を感じます。

*チームのエンパワーメント。*管理者から指示された期日とは対照的に、チームが設定した期日は、チーム メンバが現実的なものとして受け入れられる見積もりに基づいてスケジュールされているため、チームをエンパワーメントします。

チーム見積もりの統合

各チーム リーダーには、各々の役割が担当する成果物を完了するのに必要な時間の見積もりを準備する責任があります。(開発リーダーは開発者のために見積もりを準備します。ユーザー エクスペリエンス リーダーはユーザー向けドキュメントなどの見積もりを準備します。)

プログラムマネジメントの役割はチーム見積もりプロセスを調整して、すべての見積もりをマスタースケジュールと予算に統合します。

付録 A

旧バージョンの MSF からの変更点

旧バージョンの MSF は 2 つのプロセス モデルで構成され、それぞれが 4 つのフェーズと 4 つの主要マイルストンで構成されていました。プロジェクトの目的がアプリケーション開発 (ビルド) であるのかインフラストラクチャ展開 (デプロイメント) であるのかにより、フェーズとマイルストンの名前と定義が異なっていました。これら 2 つの補完的なモデルは、MSF v.3.0 の単一プロセス モデルに併合されました。2 つのプロセス モデルを併合した結果として、アプリケーション開発プロセス モデルに 1 つのフェーズが追加され、開発に関するフェーズの終わりと展開に関するフェーズの始まりを意味する諸活動に対応しています。

アプリケーション開発 (AD) のプロセス モデルが最初に開発されました。その目的は、Microsoft 社内の製品チームが使用している文化とプロセスの最も良いところを統合し、ソフトウェアを構築し、そのソフトウェアを顧客やパートナーに提供することでした。

その後、Microsoft の顧客から、エンタープライズ内での製品やサーバーの大規模な展開を手引きする類似のプロセスが要求されるようになりました。このニーズに応えるために、アプリケーション開発モデルを改造して、インフラストラクチャ展開 (ID) 向けのプロセス モデルが作成されました。

これらのモデルは何年にもわたってその有効性を実証してきましたが、単一の統合プロセス モデルのニーズが以下の理由で明白になりました。

- AD と ID モデルの相互作用を明確にする必要があります。
- エンタープライズ カスタム ソリューションおよび伝統的な Web 開発に取り組んでいるチームをより良く支援する必要があります。このような現場では、ビルドと展開は一般的に単一の統合された仕事です。
- Web サービスの出現、および Microsoft の .NET 戦略を十分に支援する必要があります。これらがソフトウェア配布の頻繁に使用されるチャネルになるにつれ、商用ソフトウェア ベンダも、展開を製品ライフサイクルの不可欠な部分と考えるようになるでしょう。
- 開発チームから運用チーム (特に、これらのチームが MOF を使用している場合) へのソリューションの引渡しをより容易にする必要があります。

付記

[1] Royce, Winston W., "Managing the Development of Large Software Systems," *Proceedings of IEEE Wescon* (August 1970): pp 1-9.

[2] Barry Boehm, "A Spiral Model of Software Development and Enhancement", *IEEE Computer*, Vol.21, No. 5 (May 1988): pp 61-72.