Microsoft Solutions Framework

White Paper

Published: June 2002 For more information on MSF, see: http://www.microsoft.com/msf

MSF Process Model v. 3.1

Contents

Credits

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

Part Number: 602-i399a

Abstract

The MSF process model describes a high-level sequence of activities for building and deploying IT solutions. Rather than prescribing a specific series of procedures, it is flexible enough to accommodate a broad range of IT projects. It combines two industry standard models: the waterfall and the spiral. An innovative aspect of this new version of the MSF model is that it covers the life cycle of a solution from project inception to live deployment. This helps project teams focus on customer business value, since no value is realized until the solution is deployed and in operation.

MSF is a milestone-driven process. Milestones are points in the project when important deliverables have been completed and can be reviewed. Many key questions about the project are asked and answered, such as: Does the team agree on the project scope? Have we planned enough to proceed? Have we built what we said we would build? Is the solution working properly for the customer?

The MSF process model is designed to accommodate changing project requirements by moving iterations through short development cycles and incremental versions of the solution.

A number of supporting practices are recommended that help project teams use the process model successfully.

Overview of Frameworks

To maximize the success of IT projects, Microsoft has made available packaged guidance on effectively designing, developing, deploying, operating, and supporting solutions built on Microsoft technologies. This knowledge is derived from the experience gained within Microsoft on large-scale software development and service operation projects, the experience of Microsoft's consultants in conducting projects for enterprise customers, and the best knowledge from the worldwide IT industry. The guidance is organized into two complementary and well-integrated bodies of knowledge, or "frameworks." These are Microsoft Solutions Framework (MSF) and Microsoft Operations Framework (MOF).

Creating a business solution on time and within budget requires a proven approach. MSF provides proven practices for planning, designing, developing, and deploying successful IT solutions. As opposed to a prescriptive methodology, MSF provides a flexible and scalable framework to meet the needs of any size organization or project team. MSF guidance consists of principles, models, and disciplines for managing the people, process, technology elements, and their tradeoffs that most projects encounter. Information on MSF is available on the Web at: http://www.microsoft.com/msf

MOF provides technical guidance that enables organizations to achieve mission-critical system reliability, availability, supportability, and manageability of IT solutions built using Microsoft products and technologies. MOF's guidance addresses the people, process, technology, and management issues pertaining to operating complex, distributed, heterogeneous IT environments. MOF is based on industry best practices as documented in the IT Infrastructure Library (ITIL) from the Central Computer and Telecommunications Agency, an agency of the UK government. Information on MOF is available on the Web at: http://www.microsoft.com/mof

Introduction

Process models establish the order of project activities. In this way, they represent the entire life cycle of a project. Currently, businesses employ a variety of process models. The MSF process model originated with the process used by Microsoft to develop applications. It has evolved to combine some of the most effective principles of other popular process models into a single model that may be applied across any project type—a phase-based, milestone-driven, and iterative model. This model may be applied to traditional application development environments, but is equally appropriate for the development and deployment of enterprise solutions for e-commerce, Web-distributed applications, and other multi-faceted initiatives that may appear in the future.

Other Process Models

Currently, the waterfall model and the spiral model are two popular process models used in the information technology industry:

- *Waterfall mode*[1]

- . This model uses milestones as transition and assessment points. In the waterfall model, each set of tasks must be completed before the next phase can begin. The waterfall works best for projects where it is feasible to clearly delineate a fixed set of unchanging project requirements at the start. Fixed transition points between phases facilitate schedule tracking and assignment of responsibilities and accountability.

  Figure 1 depicts the waterfall model. Milestones are shown as diamonds and phases are shown as arrows.

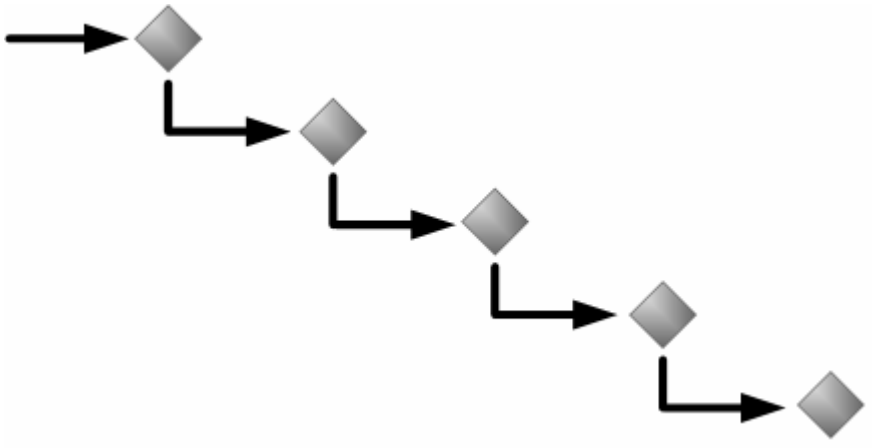

Figure 1 – Waterfall Model

- *Spiral mode[2]l.* This model focuses on the continual need to refine the requirements and estimates for a project. The spiral model, shown in Figure 2, can be very effective when used for rapid application development on a very small project. This approach stimulates great synergy between the development team and the customer because the customer provides feedback and approval for all stages of the project. However, since the model does not incorporate clear checkpoints, the development process may become chaotic.

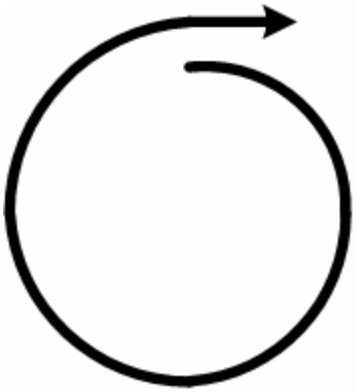

Figure 2 – Spiral Model

Best of Two Worlds

The MSF process model, shown in Figure 3, combines the best principles of the waterfall and spiral models. It derives the benefits of predictability from the milestone-based planning of the waterfall model, as well as the benefits of feedback and creativity from the spiral model. Details of the milestones and phases will be discussed later in this paper.

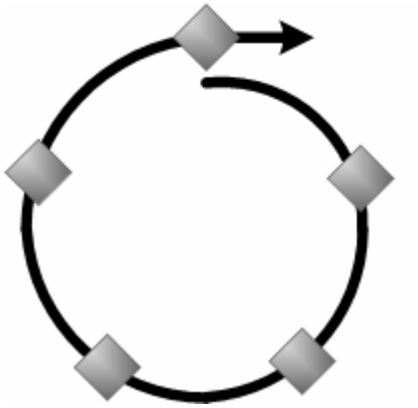

Figure 3 – MSF Process Model

Underlying MSF Principles

The MSF process model is directly associated with the following four MSF principles.

Work Toward a Shared Vision

Fundamental to the success of any joint activity is that team members and the customer have a shared vision—that is, a clear understanding as to what the goals and objectives are for the solution. Team members and customers all bring with them assumptions as to what the activity is going to do for the organization. A shared vision brings those assumptions to light and ensures that all participants are working to accomplish the same goal.

Clarifying and getting commitment to a shared vision is so important that the MSF process model designates a phase and a major milestone for that purpose.

Stay Agile—Expect Things to Change

Traditional project management disciplines and the waterfall process model assume that requirements can be clearly articulated at the outset and that they will not change significantly during a project life cycle. MSF, in contrast, makes the fundamental assumption that continual change should be expected and managed.

Focus on Delivering Business Value

Successful solutions, whether targeted at organizations or individuals, must satisfy some basic need and deliver value or benefit to the customer. For individuals, the benefit may be in satisfying some emotional need, such as most computer games. For organizations, however, the key driver is business value.

A solution does not provide value until it is fully deployed into live production. For this reason, the life cycle of the MSF process model includes both development and deployment of a solution.

Foster Open Communication

Historically many organizations and projects have operated purely on a need-to-know basis. In other words, information is only given to people who can prove that they must have the information to do their job. This approach frequently leads to misunderstandings which impair the ability of a team to deliver a successful solution.

The MSF process model prescribes an open and honest approach to communications, both within the team and with key stakeholders. A free flow of information not only reduces the chances of misunderstandings and wasted effort; it also ensures that all team members can contribute toward reducing uncertainties surrounding the project.

For these reasons, the MSF process model provides review points. Documented deliverables keep the progress of the project visible and well communicated among the team, stakeholders, and the customer.

Key Concepts for the MSF Process Model

To understand the process model, it is important to understand the way MSF has defined the following concepts and terms.

Customers

MSF distinguishes between the customer and the user.

For consumer software products, games, and Web applications, the customer and the user can be the same.

For business solutions, however, the customer is the person or organization that commissions the project, provides funding, and who expects to get business value from the solution. Users are the people who interact with the solution in their work. For example, a team is building a corporate expense reporting system that allows employees to submit their expense reports using the company intranet. The users are the employees, while the customer is a member of management charged with establishing the new system.

- *Customer participation.* Customer involvement in IT projects is essential for success. The MSF process model allows the customer many opportunities to shape and modify requirements and to set checkpoints to review progress. These activities require time and commitment from the customer.
- *Internal or external customers*. Depending on the circumstances of the project, the customer and the team may not belong to the same organization. For example, the customer may be a “buyer” contracting with an external “supplier” (which may be a virtual team of various partnering organizations).
- *Contracts*. MSF acknowledges that the contractual and legal relationship between a customer, its suppliers, and the solution team is very important and must be managed carefully. This approach, called Procurement Management, is described in the MSF Project Management Discipline white paper. However, as there are many sources of guidance available on this subject, this topic is not covered in depth.

Stakeholders

Stakeholders are individuals or groups who have an interest at stake in the outcome of the project. Their goals or priorities are not always identical to those of the customer. Each stakeholder will have requirements or features that are important to them.

Responsibilities of the product management role include identifying the key stakeholders of the project, taking their needs into account, and managing stakeholder relationships.

Examples of stakeholders commonly found in IT projects include:

- Departmental managers whose staff and business processes will be changed by the solution the team is building.
- IT operations staff that will be responsible for running and supporting the solution or who run other applications that may be affected by the solution.
- Functional managers who are contributing resources to the project team.

What Is a Solution?

In every day use, a solution is simply a strategy or method to solve a problem. It has become common marketing jargon in the IT industry to describe products as "solutions." As such, there is confusion, even skepticism, over exactly what "solution" means.

In MSF, the term "solution" has a very specific meaning. It is the *coordinated delivery* of the elements needed (such as technologies, documentation, training, and support) to successfully respond to a *unique customer's* business problem. While MSF is used to develop commercial products for a mass market, it focuses mainly on delivering solutions tailored to a specific customer.

A solution may include one or more software products, but the difference between products and solutions must be kept clear. The differences are summarized in the table below.

Products	MSF Solution
Designed for the needs of a mass market.	Designed or tailored to fit individual customer needs.
Delivered as a packaged goods or "bits" (by way of download, CD-ROM, and so on).	Delivered as a project.

Figure 4 shows the main elements of a successful solution.

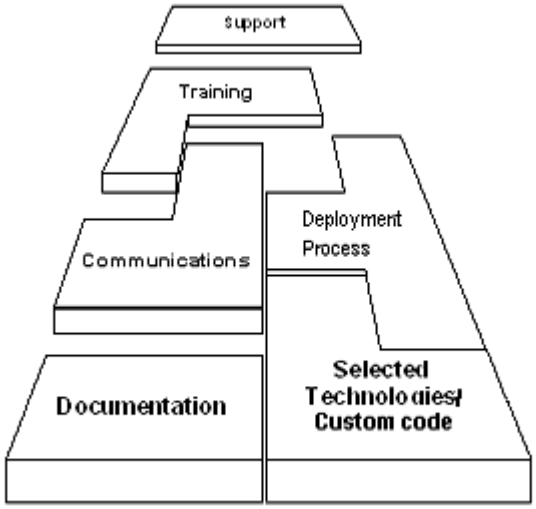

Figure 4 – Elements of a Solution

Projects may vary in complexity and the amount of effort necessary for development. Some of the elements shown in Figure 4 may not be necessary in a relatively simple deployment. However, more complex, larger-scale projects most likely require all of the elements illustrated above.

In addition:

- Selected technologies/custom code may be new, upgraded, updated, or include added components.
- Technologies may include hardware, software, peripherals, or network components. Custom code is code developed for a specific project.
- Training applies to everyone who will be using or supporting the solution that is to be deployed.
- Documentation refers to all the information needed to install, maintain, support, and use the solution.
- Support processes include the procedures necessary to perform backups, restorations, disaster recovery, troubleshooting, and Help desk functions.
- External communications involve keeping external stakeholders apprised of the progress of the deployment and the ways in which the solution will affect them.
- Deployment processes include installation/un-installation procedures for deploying hardware and software, automated deployment tools, and procedures for emergency rollback.

Baselining

In the MSF process model, a baseline is a measurement or known state by which something is measured or compared. Establishing baselines is a recurring theme in MSF. Source code, server configurations, schedules, specifications, user manuals, and budgets are just some examples of deliverables that are baselined in MSF. Without baselines, it is impossible to manage change.

Scope

Scope is the sum of deliverables and services to be provided in the project. The scope defines what must be done to support the shared vision. It integrates the shared vision, mapped against reality, and reflects what the customer deems essential for success of the release. As a part of defining the scope, less urgent functionality is moved to future projects.

The benefits of defining the scope are:

- Dividing a long-term vision into achievable chunks.
- Defining the features that will be in each release.
- Allowing flexibility for change.
- Providing a baseline for trade-offs.

The scope of a solution's features must be defined and managed as well as the scope of work and services being provided by the project team.

The term "scope" has two aspects: the solution scope and the project scope. While there is a correlation between these two, they are not the same. Understanding this distinction helps teams manage the schedule and cost of their projects.

The *solution scope* describes the solution's features and the deliverables, including non-code deliverables. A feature is a desirable or notable aspect of an application or piece of hardware. For example, the ability to preview before printing is a feature of a word processing application; the ability to encrypt e-mail messages before sending is a feature of a messaging application. The accompanying user manual, online Help files, operations guides, and training are also features of the overall solution.

The *project scope* describes the work to be performed by the team in order to deliver each item described in the solution scope. Some organizations define project scope as a statement of work (SOW) to be performed.

Clarifying the project scope includes the following benefits:

- Focuses the team on identifying what work must be done.
- Facilitates breaking down large, vague tasks into smaller, understandable ones.
- Identifies specific project work that is not clearly associated with any specific feature, such as preparing status reports.
- Facilitates subdividing the work among subcontractors or partners on the team.
- Clarifies those parts of the solution that the team is responsible for as well as the ones for which it is not responsible.
- Ensures that all parts of the solution have a clear owner responsible for building or maintaining it. For large solutions especially, features are part of the solution, but not part of the project team's deliverables. For example, a team may be building a corporate procurement solution that interacts with a company's enterprise resource management (ERP) system. The integration is part of the overall solution scope, but not necessarily part of the project scope for that team.

Managing Tradeoffs

Managing scope is critical for project success. Many IT projects fail, are completed late or go dramatically over-budget due to poorly managed scope. Managing scope includes clarifying the scope early and good project tracking and change control.

Due to the inherent uncertainty and risk involved with IT projects, making effective trade-offs is key to success.

The Tradeoff Triangle

In projects, there is a well-known relationship between the project variables of resources (people and money), schedule (time), and features (scope). These variables exist in a triangular relationship as shown in Figure 5.

After you have established the triangle, any change to one of its sides requires a correction on one or both of the other sides to maintain project balance. This includes, potentially, the same side on which the change first occurred.

The key to deploying a solution that matches the customer's needs when they need it is to find the right balance between resources, deployment date, and features.

Customers are sometimes reluctant to cut favorite features. The tradeoff triangle helps to explain the constraints and present tradeoff options.

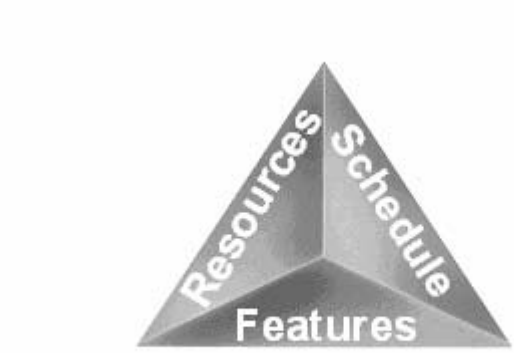

Figure 5 – Tradeoff Triangle

Features have a fixed level of quality that is presumed to be non-negotiable. You can view quality as a fourth dimension which would transform the triangle into a tetrahedron (or three-sided pyramid). Although lowering the quality bar results in simultaneously reducing resources, shortening schedule, and increasing features, it is obviously a recipe for failure.

Project Tradeoff Matrix

Another powerful tool to manage tradeoffs is the project tradeoff matrix, shown in Figure 6. This is an agreement between the team and customer, made early in the project, regarding the default priorities when making tradeoff decisions. There can be exceptions to the default priorities, if necessary. But the main benefit of establishing default priorities is to help make the tradeoffs less contentious.

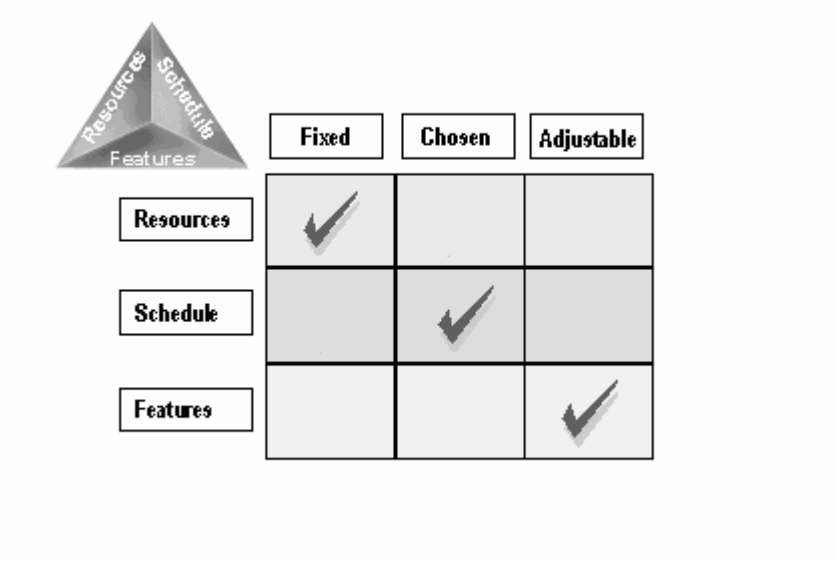

Figure 6 – Tradeoff Matrix

Figure 6 shows the typical tradeoff matrix used by Microsoft product teams. This matrix helps identify project constraints that are essentially unchangeable (represented by the Fixed column), constraints that are desired priorities (represented by the Chosen column), and constraints (represented by the Adjustable column) that can be adjusted to accommodate those constraints that are Fixed and Chosen.

Features are not cut casually. Both the team and the customer must review all project constraints carefully and be prepared to make difficult choices.

To understand how the tradeoff matrix works, resource, schedule, and feature variables can be inserted in the blanks of the following sentence:

Given fixed _____, we will choose a _____ and adjust _____ as necessary.

Logical sentence possibilities are:

- Given fixed resources, we will choose a schedule and adjust the feature set as necessary.
- Given fixed resources, we will choose a feature set and adjust the schedule as necessary.
- Given a fixed feature set, we will choose a level of resources and adjust schedule as necessary.
- Given a fixed feature set, we will choose a schedule and adjust resources as necessary.
- Given a fixed schedule, we will choose a level of resources and adjust the features set as necessary.
- Given a fixed schedule, we will choose a feature set and adjust resources as necessary.

It is essential that the team and the customer are completely clear on the tradeoff matrix for the project.

Characteristics of the Process Model

The three distinctive features of the MSF process are:

- A phase and milestone-based approach.
- An iterative approach.
- An integrated approach to building and deploying solutions.

A Milestone-Based Approach

Characteristics of the Milestone-Based Approach

Milestones, a central theme in MSF, are used to plan and monitor project progress.

Major Milestones and Interim Milestones

MSF distinguishes between two types of milestones: Major milestones and interim milestones. Features of major and interim milestones are:

- Major milestones serve to transition from one phase to another and to transition responsibility across roles.
- MSF defines specific major milestones that are generic enough for any type of IT project.
- Interim milestones serve as early progress indicators and segment large work efforts into workable pieces.
- Interim milestones vary depending on the type of project. MSF provides a set of suggested interim milestones, but teams define specific interim milestones that make sense for their projects.

Milestones as Synchronization Points

The major milestones are points in the project life cycle when the entire team synchronizes the milestone's deliverables with each other and with customer expectations. At this time, project deliverables are formally reviewed by the customer, the stakeholders, and the team. Successful achievement of a major milestone represents team and customer agreement to proceed with the project.

Although it is possible to have a completely predictable project by picking an exceptionally late release date, this is costly and doesn't meet business needs. The milestones allow the customer and the team to either reconfirm the project scope or adjust the scope of the project to reflect changing customer requirements or to react to risks.

Milestone-Driven Accountability

Although the program management role orchestrates the overall process within each phase, the successful achievement of each milestone requires special leadership and accountability from each of the other team roles. As a project moves sequentially through each phase, the level of effort for each of the roles varies. The use of milestones helps to manage this ebb and flow of involvement in the project.

Different Roles Drive Different Phases

The alignment of team roles with each of the five external milestones clarifies which role is primarily responsible for achieving each milestone. This creates clear accountability. When the project moves to a different phase, part of the process often includes transitioning responsibility to other roles.

The chart below shows the roles which drive each milestone. Although the completion of each milestone is driven by one or two roles, all roles participate throughout the project life cycle.

Milestone	Primary driver
Vision/Scope Approved	Product Management
Project Plans Approved	Program Management
Scope Complete	Development and User Experience
Release Readiness Approved	Testing and Release Management
Deployment Complete	Release Management

Post-Milestone Reviews

Each major milestone provides an opportunity for learning and reflection on the progress of the phase just completed. Post-milestone reviews provide a good forum for this reflection. These are different in purpose from milestone review meetings, which are conducted with the customer and other stakeholders to evaluate milestone deliverables. The final post-milestone review occurs at the end of the project. Some organizations call this a postmortem.

An Iterative Approach

Characteristics of an Iterative Approach

The practice of iterative development is a recurrent theme in MSF. Code, documents, designs, plans, and other deliverables are developed in an iterative fashion.

Versioned releases

MSF recommends that solutions be developed by building, testing and deploying core functionality. Later sets of features are added. This is known as a version release strategy. It is true that some small projects may only need one version. Nevertheless, it is a recommended practice to look for opportunities to break a solution into a multiple versions. Figure 7 shows how functionality develops over many versions.

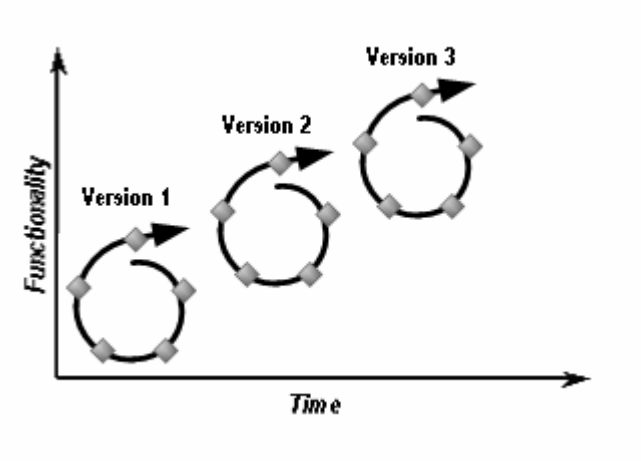

Figure 7 – Using Versioned Releases

Versioned releases do not necessarily occur sequentially. Mature software products often are developed by multiple version teams working with overlapping release cycles. The time between versions varies on the size and type of project, as well as customer needs and strategy.

Create Living Documents

To avoid spiraling out of control, iterative development requires documentation that changes as the project changes. These "living documents" are maintained in a different way than they are with a waterfall approach, where no development begins until all requirements and specifications are complete and locked down.

MSF project documents are developed iteratively, much like code. Planning documents often start out as a high-level "approach." These are circulated for review by the team and stakeholders during the envisioning phase. As the project moves into the planning phase, these are developed into detailed plans. Again these are reviewed and modified iteratively. The types and number of these plans vary with the size of the project.

To avoid confusion, planning documents that are started during the envisioning phase are referred to as "approaches." For example, a brief test approach can be written during envisioning that evolves into a test plan in later phases.

Baseline Early, Freeze Late

By creating and baselining project documents early in the process, team members are empowered to begin development work without the delays that may be incurred in excessive planning. By making the documents flexible by freezing them late within their corresponding phases, changes can be accommodated during development. This flexibility requires careful attention to the change control process. It is essential to track changes and ensure that no unauthorized changes occur.

Daily builds

MSF advocates preparing frequent builds of all the components of the solution for testing and review. This approach is recommended for developing code as well as for "builds" of hardware and software components. This approach enables the stability of the total solution to be well-understood, with ample test data, before the solution is released into production.

Larger, complex projects are often split into multiple segments, each of which is developed and tested by separate sub teams or feature teams, then consolidated into the whole. In projects of this type, typical in Microsoft product development, the "daily build" approach is a fundamental part of the process. Core functionality of the solution or product is completed first, and then additional features are added. Development and testing occur continuously and simultaneously in parallel tracks. The daily build provides validation that all of the code is compatible, and allows the various sub teams to continue their development and testing iterations.

Note that these iterative builds are not deployed in the live production environment. Only when the builds are well-tested and stable are they ready for a limited pilot (or beta) release to a subset of the production environment.

Rigorous configuration management is essential to keeping builds in synch.

Configuration Management

Configuration management is the *formalized* tracking and control of the state of various project elements. These elements include version control for code, documentation, user manuals and Help files, schedules, and plans. It also includes tracking the state of hardware, network, and software settings of a solution. The team must be able to reproduce or "roll back" to an earlier configuration of the entire solution if this is needed.

Configuration management is often confused with project change control which is discussed below. The two are interrelated, but not the same. Configuration management is the tracking of the state of project deliverables and documents. Change control is the process used to review and approve changes. Configuration management provides the baseline data that the team needs in order to make effective change control decisions.

For example, a team is working on an electronic healthcare claims system for a chain of hospitals. They record the settings selected on a Microsoft® BizTalk® server and track changes as they are made during development and testing. This is an example of configuration management. To conform to new government regulations, someone has proposed adding a new EDI mapping schema. Key team members meet with the manager funding the project and members of the operations staff to review the proposed change, its technical risk, and impact to cost and schedule. This is an example of change control.

For organizations using MOF, configuration management for the project can adapt many of the configuration management processes used for operations.

Guidelines for Versioned Releases

Versioned releases improve the team's relationship with the customer and ensure that the best ideas are reflected in the solution. Customers will be more receptive to deferring features until a later release if they trust the team to deliver the initial and subsequent solution releases in a timely fashion. Guidelines facilitating the adoption of versioned releases are:

- Create a multi-release plan.
- Deliver core functionality first.
- Cycle through iterations rapidly.
- Establish change control.
- Stop creating new versions when they no longer add value.

Create a Multi-Release Plan

Thinking beyond the current version enhances a team's ability to make good decisions about what to build now and what to defer. By providing a time table for future feature development, the team is able to make the best use of available resources and schedule constraints, as well as to prevent unwanted scope expansion.

Deliver Core Functionality First

A basic, solid and usable solution in the customer's hands is of more immediate value than a deluxe version that won't be available for weeks, months, or years. By delivering core functionality first, developers have a solid foundation upon which to build, and benefit from customer feedback that will help drive feature development in subsequent iterations.

Prioritize Using Risk-Driven Scheduling

Risk assessment by the team identifies which features are riskiest. For more information on risk, see the MSF Risk Management Discipline whitepaper. Schedule the riskiest features for completion first. Problems requiring major changes to the architecture can be handled earlier in the project, thereby minimizing the impact to schedule and budget.

Cycle through Iterations Rapidly

A significant benefit of versioning is that it delivers usable solutions to the customer expediently, and improves them incrementally. If this process stalls, customer expectations for continual product improvement suffer. Maintain a manageable scope so that iterations are achievable within acceptable time frames.

Establish Change Control

Once the specifications are baselined, all of the features and functionality of the solution should be considered to be under change control. It is essential that the entire team and customer understands what this means and understands the change control process.

MSF does not prescribe a specific set of change control procedures. These can be simple or very elaborate, depending on the size and nature of project. However, effective change control must have the following elements:

- Features are not added or changed without review and approval by both team and customer.
- To facilitate review, requests to change features are submitted in writing. This allows tracking of groups of change requests. At Microsoft, these are known as design change requests (DCRs).
- Analyze each feature request for impact, feasibility and priority. Consider dependencies with other features, including user and operational documentation, training materials, and the operating environment.
- Estimate the impact to cost and schedule for each change request (see the Bottom-Up Estimating section for more details).
- Specify individuals (including the customer, program management, and some combination of stakeholders and other team members) to serve on a change control board to authorize changes. Such a group can take many forms, as long as it is authorized to approve changes to cost, schedule, and functionality.
- Track changes and make them easy to access. For example, it is a good practice to maintain a change log section in functional specifications and other important documents.
- Change control requires effective configuration management to be effective.

An Integrated View of Development and Deployment

As stated previously, a solution does not provide value until it is fully deployed into live production. It is for this reason that the MSF process model follows the trajectory of a solution until the point at which it begins delivering value—when deployment is complete.

Benefits of an Integrated Process Model

A process model that integrates application development and deployment provides the following benefits.

Focused on Enterprise Needs

Enterprises (especially business decision makers) generally perceive the building and deployment of a solution as a single consolidated undertaking. Even if a solution is developed successfully, business decision makers do not see return on investment until it is deployed to the enterprise.

Enhanced Support for Traditional Web Development

Web development teams today build and deploy (host) Web sites as a single planned, coordinated effort.

Enhanced Support for Web Services

Web services must be designed and built for immediate deployment to their hosting environment. As Web services become a more frequently-used channel for software delivery, even commercial software vendors will find it makes sense to consider deployment as an integral part of their product lifecycle.

Removes "Over-the-Wall" Handoffs to Operations

It is common for development teams to build solutions without taking sufficient account of operational requirements. This results in applications with poor performance, availability, and manageability. MSF's integrated process model transitions ownership from development to operations teams over a series of interim milestones, not in one "cold" handoff.

Notes for Using the Integrated Process Model

Phases Not Equal in Duration

While the process model graphic shows equal sized phases, this is not to imply that each phase takes similar amounts of time. Depending on the project, the amount of time in each phase can vary dramatically.

Activities Often Span Phases

New practitioners of MSF may think that the activities associated with a phase are only done during that phase. This is not the case. For example, planning does not only occur during the planning phase, testing occurs outside of the stabilizing phase, and development can be ongoing outside of the developing phase. Phases are characterized by the goals and deliverables and, to a lesser extent, by the typical activities that the team is focused on at various times.

Creating, updating, and refining plans continues throughout the project. However, the bulk of planning occurs during the planning phase and key plan deliverables get a full review during the planning phase.

"Pure" Application Development and Infrastructure Deployment Projects

Some projects do not involve both building *and* deploying solutions. Commercial software vendors building "shrink wrap" products obviously do not deploy that which they build for their customers, although they need to thoroughly understand what is involved. Likewise, teams on infrastructure deployment projects are not creating the technologies they are deploying, although development activities must take place, such as building automated installation scripts.

Teams on pure application development or pure infrastructure deployment projects may simply skip over references and interim milestones that do not apply to their type of project.

Process Model Phases and Milestones

MSF version 3.0 integrates the two former MSF process models, application development (AD) and infrastructure deployment (ID). This single model follows the development of a solution from its inception to full deployment. By doing so, the previous four-phased pattern has expanded to five phases. Each phase culminates in an externally visible milestone. Figure 8 illustrates the MSF process model phases and milestones. Although the appearance of the graphic shown in Figure 8 may surprise some MSF practitioners, the change is much less drastic than it might look. This is because none of the important substance of the original two models has been lost. The best details of both have simply been merged together in sequence. See Appendix A for the background and rationale of the changes made in MSF v.3.0.

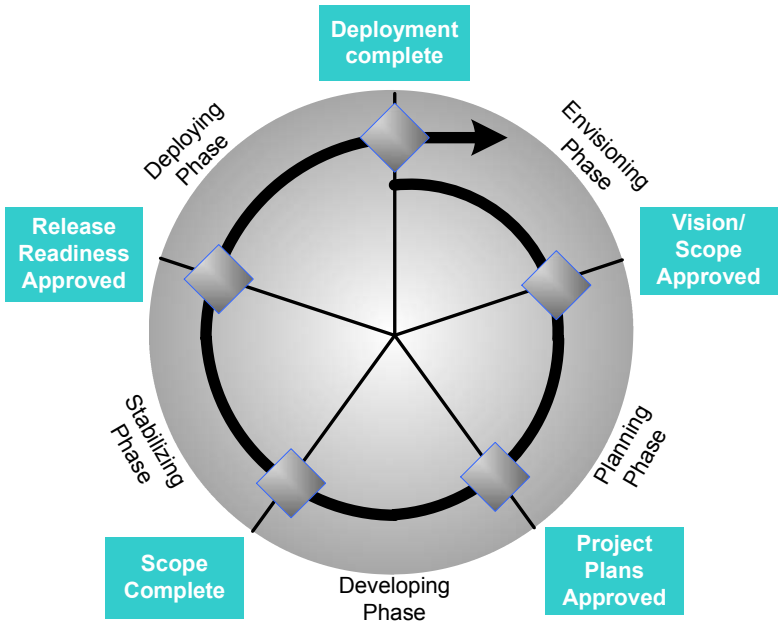

Figure 8 – MSF Process Model Phases and Milestones

Envisioning Phase

Overview

The envisioning phase addresses one of the most fundamental requirements for project success—unification of the project team behind a common vision. The team must have a clear vision of what it wants to accomplish for the customer and be able to state it in terms that will motivate the entire team and the customer. Envisioning, by creating a high-level view of the project's goals and constraints, can serve as an early form of planning; it sets the stage for the more formal planning process that will take place during the project's planning phase.

The primary activities accomplished during envisioning are the formation of the core team (described below) and the preparation and delivery of a vision/scope document. The delineation of the project vision and the identification of the project scope are distinct activities; both are required for a successful project. *Vision* is an unbounded view of what a solution may be. *Scope* identifies the part(s) of the vision can be accomplished within the project constraints.

Risk management is a recurring process that continues throughout the project. During the envisioning phase, the team prepares a risk document and presents the top risks along with the vision/scope document. For more information, see the MSF Risk Management Discipline white paper.

During the envisioning phase, business requirements must be identified and analyzed. These are refined more rigorously during the planning phase.

The primary team role driving the envisioning phase is the product management role.

Vision/Scope Approved Milestone

The vision/scope approved milestone culminates the envisioning phase. At this point, the project team and the customer have agreed on the overall direction for the project, as well as which features the solution will and will not include, and a general timetable for delivery.

Deliverables

The deliverables for the envisioning phase are:

- Vision/scope document.
- Risk assessment document.
- Project structure document.

Team Focus during the Envisioning Phase

The following table describes the focus and responsibility areas of each team role during the envisioning phase.

Role	Focus
Product Management	Overall goals; identify customer needs, requirements; vision/scope document
Program Management	Design goals; solution concept; project structure
Development	Prototypes; development and technology options; feasibility analysis
User Experience	User performance needs and implications
Testing	Testing strategies; testing acceptance criteria; implications
Release Management	Deployment implications; operations management and supportability; operational acceptance criteria

Suggested Interim Milestones

Core Team Organized

This is the point at which key team members have been assigned to the project. Typically, the full team is not assembled yet. The initial team may often be playing multiple roles until all members are in place.

The project structure document includes information on how the team is organized and who plays which roles and has specific responsibilities. The project structure document also clarifies the chain of accountability to the customer and designated points of contact that the project team has with the customer. These can vary depending on the circumstances of the project.

Vision/Scope Drafted

At this interim milestone, the first draft of the vision/scope document has been completed and is circulated among the team, customer, and stakeholders for review. During the review cycle, the document undergoes iterations of feedback, discussion, and change.

Planning Phase

Overview

The planning phase is when the bulk of the planning for the project is completed. During this phase the team prepares the functional specification, works through the design process, and prepares work plans, cost estimates, and schedules for the various deliverables.

Early in the planning phase, the team analyzes and documents requirements in a list or tool. Requirements fall into four broad categories: business requirements, user requirements, operational requirements, and system requirements (those of the solution itself). As the team moves on to design the solution and create the functional specifications, it is important to maintain *traceability* between requirements and features. Traceability does not have to be on a one to one basis. Maintaining traceability serves as one way to check the correctness of design and to verify that the design meets the goals and requirements of the solution.

The design process gives the team a systematic way to work from abstract concepts down to specific technical detail. This begins with a systematic analysis of *user profiles* (also called "personas") which describe various types of users and their job functions (operations staff are users too). Much of this is often done during the envisioning phase. These are broken into a series of *usage scenarios*, where a particular type of user is attempting to complete a type of activity, such as front desk registration in a hotel or administering user passwords for a system administrator. Finally, each usage scenario is broken into a specific sequence of tasks, known as *use cases*, which the user performs to complete that activity. This is called "story-boarding."

There are three levels in the design process: conceptual design, logical design, and physical design. Each level is completed and baselined in a staggered sequence.

The results of the design process are documented in the *functional specification(s).* The functional specification describes in detail how each feature is to look and behave. It also describes the architecture and the design for all the features.

The functional specification serves multiple purposes, such as:

- Instructions to developers on what to build.
- Basis for estimating work.
- Agreement with customer on exactly what will be built.
- Point of synchronization for the whole team.

Once the functional spec is baselined, detailed planning can begin. Each team lead prepares a plan or plans for the deliverables that pertain to their role and participates in team planning sessions. Examples of such plans include a deployment plan, a test plan, an operations plan, a security plan, and/or a training plan. As a group, the team reviews and identifies dependencies among the plans.

All plans are synchronized and presented together as the master project plan, not to be confused with the Microsoft® Project® .mpp file. The number and types of subsidiary plans included in the master project plan will vary depending on the scope and type of project.

Team members representing each role generate time estimates and schedules for deliverables (see the Bottom-Up Estimating section for more details). The various schedules are then synchronized and integrated into a master project schedule.

At the culmination of the planning phase—the project plans approved milestone—customers and team members have agreed in detail on what is to be delivered and when. At the project plans approved milestone, the team re-assesses risk, updates priorities, and finalizes estimates for resources and schedule.

Project Plans Approved

At the project plans approved milestone, the project team and key project stakeholders agree that interim milestones have been met, that due dates are realistic, that project roles and responsibilities are well defined, and that mechanisms are in place for addressing areas of project risk. The functional specifications, master project plan, and master project schedule provide the basis for making future trade-off decisions.

After the team approves the specifications, plans, and schedules, the documents become the project baseline. The baseline takes into account the various decisions that are reached by consensus by applying the three project planning variables: resources, schedule, and features. After the baseline is completed and approved, the team transitions to the developing phase.

After the team defines a baseline, it is placed under change control. This does not mean that all decisions reached in the planning phase are final. But it does mean that as work progresses in the developing phase, the team should review and approve any suggested changes to the baseline.

For organizations using MOF, the team submits a Request for Change (RFC) to IT operations at this milestone.

Deliverables

The following deliverables are produced during the planning phase:

- Functional specification
- Risk management plan
- Master project plan and master project schedule

Team Focus during Planning

The following table describes the focus and responsibility areas of each team role during planning.

Role	Focus
Product Management	Conceptual design; business requirements analysis; communications plan
Program Management	Conceptual and logical design; functional specification; master project plan and master project schedule, budget
Development	Technology evaluation; logical and physical design; development plan/schedule; development estimates
User Experience	Usage scenarios/use cases, user requirements, localization/accessibility requirements; user documentation/training plan/schedule for usability testing, user documentation, training
Testing	Design evaluation; testing requirements; test plan/schedule
Release Management	Design evaluation; operations requirements; pilot and deployment plan/schedule

Suggested Interim Milestones

Technology Validation

During technology validation, the team evaluates the products or technologies that will be used to build or deploy the solution to ensure that they work according to vendor's specifications. This is the initial iteration of an effort that later produces a proof of concept and, ultimately, the development of the solution itself.

Often, technology validation involves competitive evaluations (sometimes called "shoot outs") between rival technologies or suppliers.

Another activity that must be completed at this milestone is baselining the customer environment. The team conducts an audit (also known as "discovery") of the "as is" production environment the solution will be operating in. This includes server configurations, network, desktop software, and all relevant hardware.

Functional Specification Baselined

At this milestone, the functional specification is complete enough for customer and stakeholder review. At this point the team baselines the specification and begins formally tracking changes.

The functional specification is the basis for building the master project plan and schedule. The functional specification is maintained as a detailed description, as viewed from the user perspective, of what the solution will look like and how it will behave. The functional specification can only be changed with customer approval.

The results of the design process are often documented in a design document that is separate from the functional specification. The design document is focused on describing the internal workings of the solution. The design document can be kept internal to the team and can be changed without burdening the customer with technical issues.

Master Plan Baselined

In MSF, the master project plan is a collection (or “roll up”) of plans from the various roles. It is not an independent plan of its own. Depending on the type and size of project, there will be various types of plans that are merged into the master project plan. Some of these plans are shown in Figure 9.

Figure 9 – Master Project Plan

The benefits of having a plan made up of smaller plans are that it facilitates concurrent planning by various team roles and provides for clear accountability because specific roles are responsible for specific plans.

The benefits of presenting these plans as one are that they facilitate synchronization into a single schedule, facilitate reviews and approvals, and help to identify gaps and inconsistencies.

Master Schedule Baselined

The master project schedule includes all of the detailed project schedules, including the release date. Like the master project plan, the master project schedule combines and integrates all the schedules from each team lead. The team determines the release date after negotiating the functional specification draft and reviewing the master project plan draft. Often, the team will modify some of the functional specification and/or master project plan to meet a required release date. Although features, resources, and release date may vary, a fixed release date will cause the team to prioritize features, assess risks, and plan adequately.

Development and Test Environment Set Up

A working development environment allows proper development and testing of the solution so that it has no negative impact on production systems. It is generally a good idea to set up separate development servers that developers can use. The entire team should be informed that anything on such servers could become unstable and require re-installation.

This is also the environment where infrastructure components are developed, such as server configurations, deployment automation tools and hardware.

In order to avoid delay, the development and testing environment should be set up even as plans are being finalized and reviewed. This includes development workstations, servers, and tools. The backup system should be established if it is not already in place. CD-ROM images of standard server configurations are often used as machines are often “wiped” or reformatted.

If the organization does not already have a suitable test lab in place, the team must build one. The test environment must be as close a simulation to the live environment as is feasible. While this can be expensive, it is very important. Otherwise, certain bugs may go undetected until the solution is deployed “live” to production. Organizations using MOF can take advantage of information contained in the enterprise Configuration Management Database (CMDB) as a kind of bill of materials for replicating the production environment.

Developing Phase

Overview

During the developing phase the team accomplishes most of the building of solution components (documentation as well as code). However, some development work may continue into the stabilization phase in response to testing.

The developing phase involves more than code development and software developers. The infrastructure is also developed during this phase and all roles are active in building and testing deliverables.

Scope Complete Milestone

The developing phase culminates in the scope complete milestone. At this milestone, all features are complete and the solution is ready for external testing and stabilization. This milestone is the opportunity for customers and users, operations and support personnel, and key project stakeholders to evaluate the solution and identify any remaining issues that must be addressed before the solution is released.

Deliverables

The deliverables of the developing phase are:

- Source code and executables
- Installation scripts and configuration settings for deployment
- Frozen functional specification
- Performance support elements
- Test specifications and test cases

Team Focus during Developing

The following table describes the focus and responsibility areas of each team role during developing.

Role	Focus
Product Management	Customer expectations
Program Management	Functional specification management; project tracking; updating plans
Development	Code development; infrastructure development; configuration documentation
User Experience	Training; updated training plan; usability testing; graphic design
Testing	Functional testing; issues identification; documentation testing; updated test plan
Release Management	Rollout checklists, updated rollout and pilot plans; site preparation checklists

Recommended Interim Milestones

Proof of Concept Complete

The proof of concept tests key elements of the solution on a non-production simulation of the existing environment. The team walks operations staff and users through the solution to validate their requirements.

Internal Build n Complete, Internal Build n+1 Complete

Because the developing phase focuses on building the solution, the project needs interim milestones that can help the team measure build progress.

Developing is done in parallel and in segments, so the team needs a way to measure progress as a whole. Internal builds accomplish this by forcing the team to synchronize pieces at a solution level.

How many builds and how often they occur will depend on the size and duration of the project.

Often it makes sense to set interim milestones to achieve a visual design freeze and a database freeze because of the many dependencies on these. For example, the screens that are needed to create documentation and the database schema form a deep part of the overall architecture.

Stabilizing Phase

Overview

The stabilizing phase conducts testing on a solution whose features are complete. Testing during this phase emphasizes usage and operation under realistic environmental conditions. The team focuses on resolving and triaging (prioritizing) bugs and preparing the solution for release.

Early during this phase it is common for testing to report bugs at a rate faster than developers can fix them. There is no way to tell how many bugs there will be or how long it will take to fix them. There are, however, a couple of statistical signposts known as bug convergence and zero-bug bounce that helps the team project when the solution will reach stability. These signposts are described below.

MSF avoids the terms “alpha” and “beta” to describe the state of IT projects. These terms are widely used, but are interpreted in too many ways to be meaningful in industry. Teams can use these terms if desired, as long as they are defined clearly and the definitions understood among the team, customer, and stakeholders.

Once a build has been deemed stable enough to be a release candidate, the solution is deployed to a pilot group.

The stabilizing phase culminates in the release readiness milestone. Once reviewed and approved, the solution is ready for full deployment to the live production environment.

Release Readiness Milestone

The release readiness milestone occurs at the point when the team has addressed all outstanding issues and has released the solution or placed it in service. At the release milestone, responsibility for ongoing management and support of the solution officially transfers from the project team to the operations and support teams.

Deliverables

The deliverables of the stabilizing phase are:

- Golden release
- Release notes
- Performance support elements
- Test results and testing tools
- Source code and executables
- Project documents
- Milestone review

Team Focus during Stabilizing

The following describes the focus and responsibility areas of each team role during the stabilizing phase.

Role	Focus
Product Management	Communications plan execution; launch planning
Program Management	Project tracking; bug triage
Development	Bug resolution; code optimization
User Experience	Stabilization of user performance materials; training materials
Testing	Testing; bug reporting and status; configuration testing
Release Management	Pilot setup and support; deployment planning; operations and support training

Recommended Interim Milestones

Bug Convergence

Bug convergence is the point at which the team makes visible progress against the active bug count. That is, the rate of bugs resolved exceeds the rate of bugs found. Figure 10 illustrates bug convergence.

Because the bug rate will still go up and down—even after it starts its overall decline—bug convergence usually manifests itself as a trend rather than a fixed point in time. After bug convergence, the number of bugs should continue to decrease until zero-bug release. Bug convergence tells the team that the end is actually within reach.

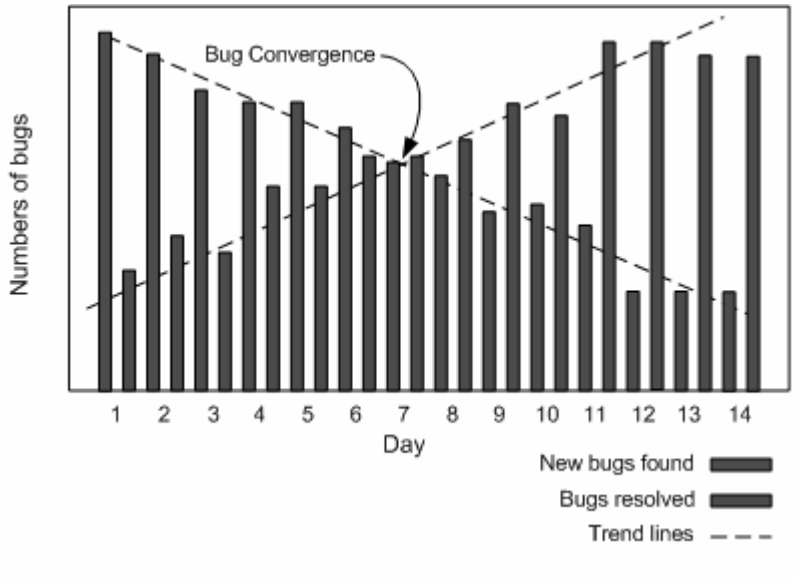

Figure 10 – Bug Convergence

Zero Bug Bounce

Zero-bug bounce is the point in the project when development finally catches up to testing and there are no active bugs—for the moment. Figure 11 illustrates a zero-bug bounce. After zero-bug bounce, the bug peaks should become noticeably smaller and should continue to decrease until the solution is stable enough for the team to build the first release candidate.

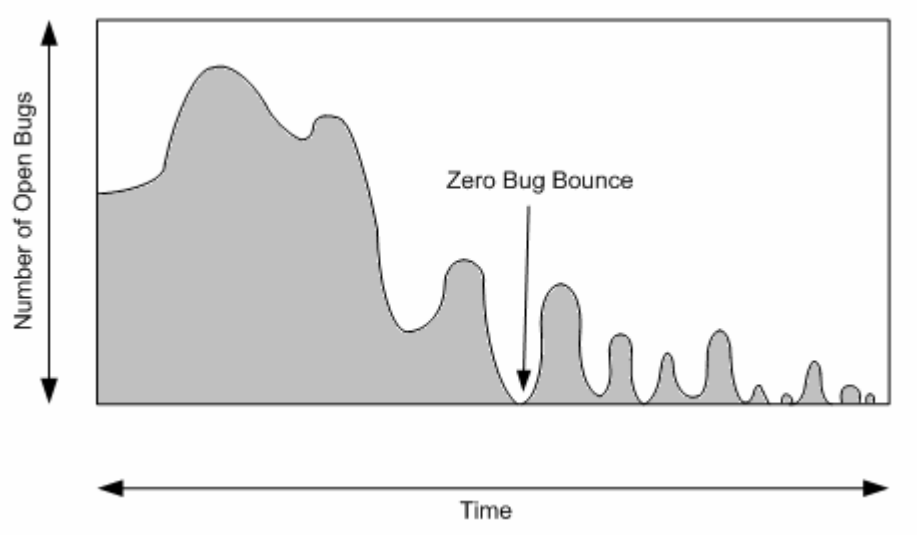

Figure 11 – Zero Bug Bounce

Careful bug triaging is vital because every bug that is fixed risks the creation of a new bug. Achieving zero-bug bounce is a clear sign that the team is in the endgame as it drives to a stable release candidate.

Note that new bugs will certainly be found after this milestone is reached. However, zero-bug bounce marks the first time when the team can honestly report that that there are no active bugs—even if it is only for the moment—and it focuses the team on working to stay at that point.

Release Candidates

A series of release candidates are prepared and released to the pilot group. Each release candidate is an interim milestone. Other features of a release candidate are:

- Each release candidate has all the elements it needs to be released to production.
- Building a release candidate tests its fitness for release, that is, whether all necessary pieces are present.
- The test period that follows generation of a release candidate determines whether a release candidate is ready to release to production or whether the team must generate a new release candidate with the appropriate fixes.
- Testing release candidates, which is done internally by the team, requires highly focused and intensive efforts, and focuses heavily on flushing out showstopper bugs.
- Testing requires a triage process for resolving any newly discovered bugs.
- It is unlikely that the first release candidate will be the one that is released. Typically, show stopping bugs will be found during the intensive testing of a release candidate.

Pre-Production Test Complete

The focus of this interim milestone is to prepare for a pilot release. This interim milestone is important because the solution is about to “touch” the live production environment. For this reason the team must test as much of the entire solution as possible before the pilot test begins.

Activities that must be completed during this interim milestone are:

- Evaluate test results against success criteria.
- Complete site preparation checklist and procedures.
- Complete implementation procedures, scripts, and load sets.
- Complete training material.
- Resolve support issues.
- Complete and test the rollback plan.

The pre-production test complete interim milestone is not complete until the team ensures that everything developed to deploy the solution is fully tested and ready.

User Acceptance Testing Complete

User acceptance testing and usability studies begin during the development phase and continue during stabilization. These are conducted to ensure that the new system is able to successfully meet user and business needs. This is not to be confused with *customer acceptance*, which occurs at the end of the project.

When this milestone has been achieved, users have tested and accepted the release in a *non-production environment* and verified that the system integrates with existing business applications and the IT production environment. The rollout and backout procedures should also be confirmed during this period.

Upon approval of release management, software developed in-house and any purchased applications are migrated from secure storage to a pristine archive location. In MOF, this is known as the Definitive Software Library (DSL). Release management is responsible for building releases (assembling the release components) in the test environment from the applications stored in the DSL.

User acceptance testing gives support personnel and users the opportunity to understand and practice the new technology through hands-on training. The process helps to identify areas where users have trouble understanding, learning, and using the solution. Release testing also gives release management the opportunity to identify issues that could prevent successful implementation.

Pilot Complete

During this interim milestone, the team will test as much of the entire solution in as true a production environment as possible. In MSF, a pilot release is a deployment to a subset of the live production environment or user group. Depending on the context of the project, a pilot release can take the following forms:

- In an enterprise, a pilot can be a group of users or a set of servers in a data center.
- In Web development, pilot release takes the form of hosting site files on staging server(s) or folders that are live on the Internet, only with a test Web address.
- Commercial software vendors, such as Microsoft, often release products to a special group of early adopters prior to final release.

What all these forms of piloting have in common is that they are instances of testing under live conditions.

The pilot complete interim milestone is not complete until the team ensures that the proposed solution is viable in the production environment and every component of the solution is ready for deployment. In addition, the following actions must be followed:

- Prior to beginning a pilot, the team and the pilot participants must clearly identify and agree upon the success criteria of the pilot. These should map back to the success criteria for the development effort.
- Any issues identified during the pilot must be resolved either by further development, by documenting resolutions and work-arounds for the installation teams and production support staff, or by incorporating them as supplemental material in training courses.
- Before the pilot is started, a support structure and issue-resolution process must be in place. This may require that support staff be trained. The procedures used for issue resolution during a pilot may vary significantly from those used during deployment and when the solution is in full production.
- In order to determine if the deployment process will work, it is necessary to implement a trial run or a rehearsal of all the elements of the deployment so that you may identify issues prior to the actual deployment.

Once enough pilot data has been collected and evaluated, the team is at a point of decision. It is at this point that one of the following strategies must be selected:

- Stagger forward—Deploy a new release to the pilot group.
- Roll back—The roll-back plan is executed and the pilot group is reverted back to the previous configuration state they had before the pilot (as closely as feasible). The pilot is tried again with a more stable release.
- Suspend—Suspend the entire pilot.
- Patch and continue—the pilot group is issued a “patch,” a fix to existing code.
- Proceed to deploying phase.

Deploying Phase

Overview

During this phase, the team deploys the core technology and site components, stabilizes the deployment, transitions the project to operations and support, and obtains final customer approval of the project. After the deployment, the team conducts a project review and a customer satisfaction survey.

Stabilizing activities may continue during this period as the project components are transferred from a test environment to a production environment.

Deployment Complete Milestone

The deployment complete milestone culminates the deploying phase. By this time, the deployed solution should be providing the expected business value to the customer and the team should have effectively terminated the processes and activities it employed to reach this goal.

The customer must agree that the team has met its objectives before it can declare the solution to be in production and close out the project. This requires a stable solution, as well as clearly stated success criteria. In order for the solution to be considered stable, appropriate operations and support systems must be in place.

Deliverables

Deliverables include:

- Operation and support information systems
- Procedures and processes
- Knowledge base, reports, logbooks
- Documentation repository for all versions of documents, load sets, and code developed during the project
- Project close-out report
- Final versions of all project documents
- Customer/user satisfaction data
- Definition of next steps

Team Focus during Deploying

The following describes the focus and responsibility areas of each team role during the deploying phase.

Role	Focus
Product Management	Customer feedback, assessment, sign-off
Program Management	Solution/scope comparison; stabilization management
Development	Problem resolution; escalation support
User Experience	Training; training schedule management
Testing	Performance testing; problem
Release Management	Site deployment management; change approval

Recommended Interim Milestones

Core Components Deployed

Most infrastructure solutions include a number of components that provide the framework or backbone for the entire solution. These components do not represent the solution from the perspective of a specific set of users or a specific site. However, the deployment of sites or users generally depends on this framework. In addition:

- Components are the enabling technology of the enterprise solution. Examples include domain controllers, mail routers, remote access servers, database servers.
- Site deployments depend on this technology.
- Depending on the solution, the core technology may need to be deployed before or in parallel with site deployments.
- To avoid delays, core components may be reviewed and approved for deployment in advance of other parts of the solution still being stabilized. The operations staff must feel confident making this commitment before the whole solution has been proved to be stable.

Site Deployments Complete Interim Milestone

At the completion of this milestone, all targeted users have access to the solution. Each site owner has signed off that their site is operating, though there may be some issues.

Customer and user feedback might reveal some problems. The training may not have gone well, or a part of the solution may have malfunctioned after the team departed the site. Some sites may need to be revisited based on feedback from site satisfaction surveys.

At this point, the team makes a concentrated effort to finish deployment activities and close out the project.

Many projects, notably in web development, do not involve client-side deployments and therefore this milestone is not applicable.

Deployment Stable Interim Milestone

At the deployment stable interim milestone, the customer and team agree that the sites are operating satisfactorily. However, it is to be expected that some issues will arise with the various site deployments. These continue to be tracked and resolved.

It can be difficult to determine when a deployment is “complete” and the team can disengage. Newly deployed systems are often in a constant state of flux, with a continuous process of identifying and managing production support issues. The team can find it difficult to close out the project because of the ongoing issues that will surface after deployment. For this reason, the team will need to clearly define a completion milestone for the deployment rather than attempt to reach a point of absolute finality.

If the customer expects members of the project team to be involved in ongoing maintenance and support, those resources should transition into a new role as part of the operations and support structure after project close-out.

At this late stage, team members and external stakeholders will likely begin to transition out of the project.

Part of disengaging from the project includes transitioning operations and support functions to permanent staff. In many cases, the resources to manage the new systems will already exist. In other cases, it may be necessary to design new support systems. Given the scope of the latter case, it may be wise to consider that as a separate project.

The period between the deployment stable and deployment complete milestones is sometimes referred to as a “quiet period.” Although the team is no longer active, team resources will respond to issues that are escalated to them. Typical quiet periods are 15 to 30 days long.

The purpose of the quiet period is to measure how well the solution is working in normal operation and to establish a baseline for understanding how much maintenance will be required to run the solution. Organizations using MOF will measure the number of incidents, the amount of downtime, and collect performance metrics of the solution. This data will help form the assumptions used by the operations Service Level Agreement (SLA) on expected yearly levels of service and performance. See the MOF Operations Guide for Service Level Management for more information on SLAs.

Recommended Practices for the MSF Process Model

The following supporting practices help teams apply the MSF process model to their project.

Focus Creativity by Evolving Features and Constraining Resources

Microsoft's general development approach is to constrain development resources and budget, which focuses creativity, forces decision-making, and optimizes the release date.

Establish Fixed Schedules

Internal time limits (a technique known as "time-boxing") keep pressure on the project team to prioritize features and activities.

Schedule for an Uncertain Future

Add buffer (additional) time to project schedules to permit the team to accommodate unexpected problems and changes. The amount of buffer to apply depends on the amount of risk. By assessing risks early in the project, the likeliest risks can be evaluated for their impact on the schedule and compensated for by adding buffer time to the project schedule.

One way to think of buffer time is as an estimate for unknown tasks and events. No matter how experienced the team, not all project tasks can be known and estimated in advance. Yet, be assured that some project risks occur and impact the project. The corrective actions required to respond to these risks will take time.

Recommended guidelines for using buffer time are:

- Buffer time should not be added by padding estimates for individual tasks. Since work expands to fill the time scheduled to do it (Parkinson's Law), the buffer will be absorbed by planned tasks, not unplanned events.
- Buffer time should be scheduled as if it were another task. Typically, buffer is allocated immediately before major milestones, especially the later ones. It always should lie on the project's critical path. The critical path is the longest chain of dependent tasks in a project and directly determines the duration of the project.
- As buffer time is expended over the course of the project, the remaining amount should be carefully tracked and conserved.
- If a feature is added, or resources removed from the project, do not compensate by using buffer time. If you do, your ability to compensate for risk has been correspondingly reduced. Negotiate features, resources, and schedule using the tradeoff triangle shown in Figure 5.
- If all of the buffer time has been used, make the whole team aware that any disruption or delay is very likely to have a "knock on" effect and jeopardize the end date.

Use Small Teams, Working in Parallel with Frequent Synchronization Points

Even a large and complex project may be divided into smaller, more efficient teams that work in parallel, if the teams periodically synchronize their activities and deliverables. This maintains a focus on consistent quality across the project, helps the program manager in charting overall progress, and emphasizes accountability within each of the teams.

Break Large Projects into Manageable Parts

A fundamental development strategy at Microsoft is to divide large projects into multiple versioned releases, with little or no separate maintenance phase.

Apply No-Blame Milestone Reviews

At each major milestone the team, customer and key stakeholders meet to review the deliverables for that milestone and assess the overall progress of the project. For large projects, this is also done at selected interim milestones as well.

After these meetings, the team conducts an internal team-facing review to evaluate team project performance. This review should be considered a Quality Assurance activity that can in turn trigger changes in how the project is being conducted.

The composition of individual team members often changes over the course of the project. Be sure to capture the input and learning of departing team members at major milestones before they move on.

Use Prototyping

Prototyping allows pre-development testing from many perspectives, especially usability, and helps create a better understanding of user interaction. It also leads to improved product specifications.

Use Frequent Builds and Quick Tests

Regular builds of the solution are the most reliable indicator available that the project is on track with development and that the team is functioning well together. Within the deployment phase, pilot testing cycles serve a similar purpose.

Cycle Rapidly

Enterprise solutions must emphasize business agility. To do this they must accommodate continuous change in customer needs. Rapid development and deployment cycles will facilitate the creation of versioned releases, which allow the evolving solution to respond to changing needs and requirements.

Avoid Scope Creep

Use the vision statement and specifications to maintain focus on the stated business goals and to trace critical features back to the original requirements. Apply the vision statement and specifications as filters to identify, discuss, and remove additional features that may have been added without proper consideration after the project had been defined.

Bottom-Up Estimating

Estimates for IT projects should be made by those who will do the work. Bottom up estimating provides the following benefits:

Better accuracy. Estimates made by those who will do the work are more accurate because the person making the estimates has had experience executing similar work.

Accountability. Those who develop their own work estimates feel more accountable for their work. They also feel more accountable for success in meeting the estimates they have made.

Team empowerment. Having team-developed dates as opposed to management-dictated dates empowers the team because the schedule is built on estimates that team members can accept as realistic.

Integrating Team Estimates

Each team lead is responsible for preparing time estimates needed to complete the deliverables their role is responsible for. (The development lead prepares estimates for developers; the user experience lead prepares estimates for UE deliverables, and so on.).

The program management role coordinates the team estimation process and integrates (“rolls up”) all the estimates into a master schedule and budget.

Appendix A

Changes from Previous Versions of MSF

Previous versions of MSF comprised two process models, each comprised of four phases and four major milestones. Depending on whether the project objective was application development (building) or infrastructure deployment (deploying), the phases and milestones had different names and definitions. These two complementary models have been merged into a single process model for MSF v.3.0. Merging the two process models has resulted in the addition of one phase to the application development process model, to accommodate activities that signify the end of the development phase and the beginning of the deployment phase.

The process model for application development (AD) was developed first. Its purpose was to consolidate the best of the culture and processes used by Microsoft's internal product teams, to build software, and to offer this software to customers and partners.

Later, Microsoft customers requested a similar process to guide the large-scale deployment of products and servers within enterprises. To answer this need, the application development model was adapted to create a process model for infrastructure deployment (ID).

Although these models have demonstrated their effectiveness for several years, the need for a single, integrated process model became evident for the following reasons:

- To clarify how the AD and ID models interact.
- To better support teams working on enterprise custom solutions and traditional web development, where building and deployment is typically a single consolidated undertaking.
- To better support the emergence of web services, and Microsoft's .NET strategy. As these become a more frequently-used channel for software delivery, even commercial software vendors will find it makes sense to consider deployment as an integral part of their product lifecycle.
- To better facilitate the handoff of solutions from development to operations teams, especially those teams using MOF.

Endnotes

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.