

---

# Microsoft Solutions Framework and the Capability Maturity Model

---

## Contents

Abstract.....	1
Establishing the Context.....	2
Searching for Best Practices.....	4
Applying Microsoft Solutions Framework (MSF).....	5
Understanding the Capability Maturity Model.....	7
Breaking CMM into Fundamental Themes.....	11
Getting Started from Ground Zero.....	13
Evolving to Higher Maturity with MSF.....	23
Discovering CMM and MSF Elements.....	31
Conclusions.....	33
References.....	33

## Abstract

### **MSF as a Vehicle for Developing Capability Maturity**

Information systems development groups, as well as many other software organizations, are increasingly concerned with the time and effort it takes them to develop useful software for their customers. In an attempt to do their work better, they seek guidance from leaders in the industry.

This paper describes two models of best practices and how they relate to each other—Microsoft Solutions Framework (MSF) and the Software Engineering Institute’s Capability Maturity Model (CMM). Each captures practices that the other does not, but their content overlaps, and MSF provides a very useful set of concepts for organizations that are evolving their capability maturity to meet the intent of the CMM.

This document was written by Dr. Joyce Statz of TeraQuest Metrics, Inc.

## Establishing the Context

*Why is building software so hard?*

*Why should developers care about having a process?*

### Working the Same Old Way

The phrase “software crisis” was invented at least 20 years ago, and the industry still behaves as if it were in crisis. Is this really true? Or is building software today more difficult than it was 20 years ago, while we are still using the same approaches? Jim McCarthy in *Dynamics of Software Development* asserts that “more people have ascended bodily into heaven than have shipped great software on time.” Why is this true?

### Finding Complex Solutions in a Short Time

In general, today’s software problems require more complex solutions than in the past. Applications must run on multiple platforms, communicate with users with several levels of sophistication, provide multiple forms of interfaces to people and other applications, and provide a broad variety of functions. The solutions must accommodate current user needs with a product lifetime that may be as short as a few months. A worst case was a financial application that had a lifetime of exactly one day because the laws changed and made the instrument no longer viable after one day in service.

### Keeping Development Agile

To meet these needs, developers must be agile. Competition in today’s telecommunications world, for example, requires concept-to-deployment cycles of no more than 12 weeks. In general, delivery cycles must fall within 6 to 9 months for most users. This dictates a need for incremental delivery of products, with the current version addressing the most important of a potentially long list of user requirements.

### Using Simple, Concise Processes

To respond to user needs quickly, developers must focus on the product content and rely on well-understood, simple, resilient processes. Unless the product is extremely simple and can be built by one person, it will require a product team and a coordinated work effort. Teams no longer have the time to deal with uncoordinated efforts. If each person on the team follows his or her own way of doing things, there is at least a lack of efficiency in addressing the tasks and, at worst, a general sense of chaos with rework to make interfaces function and to ensure product consistency. Thus, teams are

searching for concise, repeatable processes that ensure a quality result on time.

## Searching for Best Practices

*How do successful companies do this?  
What models does the industry provide?*

### Microsoft

In the last several years, Microsoft has been asked by its customers, “How does Microsoft build software? What can we leverage from what you do?” In response, several books and articles have been written about the Microsoft approach (see References at the end of this document).

The Microsoft Solutions Framework (MSF) team has gathered concepts and processes based on best practices from Microsoft product groups, Microsoft Consulting Services, the internal information systems group, customers, and the industry. The following sections describe this collection.

### Microsoft Best Practices

The industry’s tribal knowledge about Microsoft includes several best practices that have received widespread coverage:

- **Build it and it will ship.** Perform daily builds of software under development.
- **Use buddy testers.** Have a companion tester who works side by side with each developer throughout the process of building the product.
- **Form feature teams.** For a given product, break into small teams that build features for the upcoming release.
- **Use multiple cycles, with buffers.** Break a large effort into several smaller efforts and provide each with some buffer time to deal with unknowns.
- **Establish fixed ship date.** To ensure that the team focuses on building the essential features (and no more), constrain resources and schedule at the start of a project.
- **Grow the software.** Gradually build up full capability by prototyping and evolving a feature, rather than attempting to design completely at the start.
- **Share lessons learned.** Use project postmortems, process audits by experienced development managers, retreats on specific topics, and cross-group sharing by managers to leverage the best practices that evolve.

### Industry Best Practices

A number of groups throughout the world have gathered best practices in the industry. These models include the Capability

Maturity Model (CMM) for Software and the ISO 9000 standards and guidelines. Each brings together elements considered essential in a sound process to produce quality results.

Several of these models capture practices into layers of evolving organizational maturity, tailored by organizations of different sizes building different types of software. This document discusses one in detail—the CMM, developed by more than 500 industry professionals working with the process teams at the Software Engineering Institute (SEI) at Carnegie Mellon University.

## Applying Microsoft Solutions Framework (MSF)

*What is the MSF?*

*What are the MSF components?*

### Development Concepts and Principles

MSF is a collection of Microsoft's development concepts and best practices representing elements from teams executing a hundred or so development projects at any given time at Microsoft. The company does not enforce a uniform development process throughout the organization. Instead, each product team evolves what it considers to be a sound process for its work, learning from teams around it, and leveraging the experience of others in the industry. Because the teams constantly evolve their practices, the materials gathered into MSF are continually changing.

### Components

MSF consists of a collection of concepts and principles in four major areas, currently represented by four courses: *Principles of Enterprise Architecture*, *Principles of Application Development*, *Principles of Component Design*, and *Principles of Infrastructure Deployment*. Two of those courses—Principles of Application Development and Principles of Component Design—are revised versions of existing courses, while the other two are new courses.

*Principles of Enterprise Architecture* describes a structure for guiding the information systems development in an organization, composed of these related elements:

**Business architecture.** Mission, core business processes, organization structure, and roles and responsibilities.

**Application architecture.** Application models and the portfolio of applications needed to support the business architecture as well as enterprise standards for building application components.

**Technology architecture.** Criteria for selecting technologies, core technologies for the enterprise, and technology migration strategy supporting business goals.

**Information architecture.** Data origin and ownership policies, enterprise rules for data integrity, and strategies for handling and reporting data.

*Principles of Application Development* describes a framework for organizing the work of a team that is building a product. It consists of concepts and best practices in these areas that are tailored to each project:

**MSF team model.** A small, skilled, efficient team with clearly defined roles and responsibilities for building a system.

**MSF process model.** A milestone-based process for building a system, with crisp goals for each milestone, along with recommended types of deliverables and team member responsibilities.

**MSF risk management.** A proactive approach to risk management that relies on efforts to prevent a risk from occurring, or minimize its impact—mitigation—and plans for what to do if the risk does occur—contingency planning.

*Principles of Component Design* approaches design in the context of the Enterprise Architecture, with a user-centric approach and three perspectives:

**Conceptual design.** Usage scenarios to understand the business problem and derive a solution that meets user needs.

**Logical design.** Business objects and services derived from the usage scenarios.

**Physical design.** Business objects and services mapped to implementation components, thereby leveraging existing infrastructure and technologies.

*Principles of Infrastructure Deployment* describes how to apply (MSF) principles and models (specifically the team and process models) to technology infrastructure deployment projects.

**MSF team model.** A small, skilled, efficient team with clearly defined roles and responsibilities for engaging in an infrastructure deployment. This model is essentially the same as the version used in *Principles of Application Development*.

**MSF process model.** A milestone-based process for building a system, with crisp goals for each milestone, along with recommended

types of deliverables and team member responsibilities. This model is similar to the one used in *Principles of Application Development*, but with different milestone names for the last two phases.

**MSF risk management.** A proactive approach to risk management that relies on efforts to prevent a risk from occurring, or minimize its impact—mitigation—and plans for what to do if the risk does occur—contingency planning. This approach is the same as the one described in *Principles of Application Development*.

## Understanding the Capability Maturity Model

*What is the CMM?*

*How was it created?*

*How do organizations use it to improve their results?*

### The Capability Maturity Model (CMM)

The Capability Maturity Model for Software is one of several Capability Maturity Models (CMMs) developed at the Software Engineering Institute at Carnegie Mellon University since the mid-1980s. Other models include CMMs for managing people, acquiring software, personal software process, and systems engineering. These models share some features, although content and intended audience vary.

Each model provides a structured view of its focus area, generally in a five-layer model of increasingly sophisticated practices for those working in the area. With the exception of the personal software process, each is intended to be used by an organization to improve its overall capability in an incremental way. Each layer of the model provides a basis for continuous improvement in the practices already established, as well as the basis for the next layer of practices.

The CMM for software is composed of the five layers described below.

Five-level CMM for Software

Level	Focus	Key process areas
<b>5: Optimizing</b>	<b>Continuous process improvement</b>	<b>Defect prevention</b> <b>Technology change management</b> <b>Process change management</b>
<b>4: Managed</b>	<b>Product and process quality</b>	<b>Quantitative process management</b> <b>Software quality management</b>
<b>3: Defined</b>	<b>Defined engineering process</b>	<b>Organizational process focus</b> <b>Organizational process definition</b> <b>Integrated software management</b> <b>Software product engineering</b> <b>Intergroup coordination</b> <b>Training program</b> <b>Peer reviews</b>
<b>2: Repeatable</b>	<b>Project management and commitment process</b>	<b>Requirements management</b> <b>Project planning</b> <b>Project tracking and oversight</b> <b>Software subcontract management</b> <b>Software quality assurance</b> <b>Software configuration management</b>
<b>1: Initial</b>	<b>Heroes</b>	

## Working at Level 1

### Reinventing Processes as They Work

Most organizations today function at level 1, a more or less controlled chaos. Although it might have used some discipline in the past when developing legacy systems, most information systems organizations lose that culture as they move to client/server systems or as their applications become more complex and distributed across more platforms. Then delivery time, quality, and cost suffer.

In level 1 projects, teams reinvent their processes with each project or don't consider how to function as a team. Each individual produces his or her best effort, hoping that something good will come out of the effort—after all, this is a group of professionals.

### Avoiding Uncontrolled Commitments

The key stumbling block for most level 1 organizations is uncontrolled commitments. Project teams do little estimating and planning and have no data to tell them what it will take to get their jobs done. Meanwhile, management believes software is a simple matter and people at all levels of the organization make unrealistic commitment of resources. Any sound practices used by individuals or teams are likely to be abandoned in the heat of battle to meet the latest schedule crunch. Although some see this adrenaline-laced environment as an opportunity for creativity, it is actually quite a

drain of energy into late nights and last-minute crises to meet unreasonable deadlines.

## **Moving to Level 2**

### **Controlling the Requirements**

Gaining control of the fundamental day-to-day management of the work elements within a project constitutes getting to level 2 of the CMM. Here, the development team works with its customers to establish system requirements and ensures that it manages any changes that occur during the development cycle. (Change cannot be avoided, but the impact to the schedule and plan must be comprehended and managed.)

### **Managing Daily Project Activities**

The team develops project plans with input from all involved parties and monitors them to completion. If problems occur, they review the plan. Whatever the impact, the project team and its customer must face the reality of any change—from a revised schedule to new product content.

### **Using Configuration Management and Quality Assurance**

To manage its work well, the team uses a software configuration management process to control the state of its work products, to establish sound build and release processes, and to manage change to the work products throughout the development cycle. It uses quality assurance to ensure that the process is followed and if deviations occur, it makes conscious decisions about the next steps and their consequences.

### **Managing Vendor Work**

If part of a project is subcontracted, the subcontractor acts as the development team. Project management should provide a sound requirement set, require and monitor a realistic project plan, and ensure that configuration management and quality assurance are in place.

### **Tracking to a Sound Plan**

Using these practices, the team gradually builds up its understanding of how much effort it takes to do its work, using the plan to record its agreement on what to do and how to do it and to negotiate changes during development. When customers, management, and the development team work together with as much information as possible, everyone can openly make commitments, and teams have a better chance of meeting those commitments.

## Advancing to Level 3

### Leveraging and Tailoring Organizational Best Practices

Although most level 2 organizational discipline is seen at a project level, members of an individual project can usually find good practices that could be useful in another project. Leveraging those best practices across the organization and defining processes that can be tailored to each project is the heart of level 3 of the CMM. Project teams generally accumulate a rich process history as they complete their project. They should keep this data in a repository for use by other project teams on future projects. The organization will develop an appreciation for common processes and advisors who can help tailor those processes.

### Using Software Process Engineers

To transfer best practices across the organization and to manage evolving processes, organizations often establish a team responsible for this work at level 3. Often known as the Software Engineering Process Group (SEPG), this team performs for software organizations what process engineers have done for years in hardware and manufacturing organizations—help development teams select the best practices for their project. In addition, they oversee the evolution of the organization's overall process, describing a common process that encompasses the best practices and developing an overall process improvement plan for the organization.

### Defining Development Processes and Roles

At this level, the organization captures and leverages practices in all areas of software development, including efficient project management, requirements development, design, implementation, testing, peer reviews, and coordination across the different groups involved in building the products.

The organization helps prepare each member through an effective training program that covers the technologies, processes, and domain areas of its particular projects. With the common processes available to all teams, the organization can deploy its personnel where they are needed most, and individuals will know their roles and responsibilities and channel their energies into creative project solutions rather than developing yet another process to cope with problem situations.

## Evolving to Level 4

### Managing Processes to the Measures

With processes now defined and repeatable, organizations can focus on managing their work to specific quality targets. They can instrument their processes and use the process data to systematically

improve the quality of the results and the predictability of their work. Although measures are important at all levels of the CMM, at level 4 the focus should be on using the measures to remove variances in the processes and improve the performance of the processes.

### Setting Product Quality Targets

Teams in level 4 organizations are able to set project targets for defect levels they can attain with their current processes, and they are able to monitor their specific processes to ensure progress toward those targets during development.

### Improving at Level 5

Quantitative control of the processes allows everyone in the organization to participate in evaluating and improving the processes. This is the essence of the level 5 organization. Team members can prototype process innovations and technology innovations and use data to determine which elements should be included in the organization's work standards. When a defect arises, they will examine work products and processes and eliminate the causes of the defects—in that work product, in other work products that may have the same defect, in that process, and in review processes that failed to catch the defect.

Overall, the organization reflects the common view that everyone expects continuous improvement and seeks ways to make the processes work better. Only then will the organization be able to provide high-quality software delivered on time and within budget.

## Breaking CMM into Fundamental Themes

*How does the CMM support good business?  
What are the common themes?*

### Achieving Continuous Improvement

To compete effectively in a world of increasing customer demands with ever shorter cycle times, organizations must continuously improve their business processes. In general, this means paying more careful attention to the way work is done (the process) than has been true in the past. David A. Garvin in the *Harvard Business Review* notes that “many organizations are finding a competitive advantage from being process-focused in tandem with understanding their product domain.”

The focus on improvement is captured in the Plan-Do-Check-Act (PDCA) model. The CMM helps organizations evolve in their ability to use PDCA for organization improvement. Establishing a defined

process enables an organization to measure and find ways to improve the process.

### **Leveraging Organizational Learning**

To evolve the organization, learning must be a part of the work of individuals, teams, and managers. Becoming a process-focused organization requires a change in mind-set, possible only through shared experiences with the new processes.

### **Valuing Executive Leadership**

Executives should set the vision for the organization as a continuously improving, process-driven organization. They provide visible leadership and reward good role models. They ensure policies are defined, showing the organizational commitment and intended use of the processes.

### **Involving Management**

Senior management must express its commitment to the evolution and use of sound processes, and middle management must expect, encourage, and enforce the processes.

### **Measuring To Improve**

People perform against goals and measures that are set for and by them. Thus, when setting up new processes, team must measure the business impact of those processes as well as compliance with those processes. In general, we know that what gets measured gets improved, so it is very important to attach the right measures to the processes we really care about. Processes that are not useful should be revised, based on what was learned from these measures.

### **Providing Infrastructure Support**

The organization must provide training on the processes and practices, as well as the tools and personnel to enable everyone in the organization to do their work efficiently.

## Making a Culture Change

Moving from a chaotic approach to a disciplined approach requires a culture change, and organizations must address that reality with the appropriate preparation and reward structure. A key change for many organizations is to move from “rewarding the firefighter” to “rewarding the fire marshal.”

## Using Effective Process as a Competitive Advantage

Although industries such as pharmaceuticals and petroleum processing have used their processes as competitive weapons for years, only now are information-intensive businesses such as software development seeing that possibility. An organization can create a competitive advantage by evolving a suite of best practices into processes that can be tailored.

## Tailoring the Processes

Project teams must be able to select practices appropriate to the size and style of product they are building. Not all projects need the same level of rigor or communication as others. The team should tailor the CMM to fit the size and style of its project, not allow it to dictate a single approach to building software.

Leveraging these themes of the CMM and using the guidance of the key practices of the CMM, an organization can evolve a sound process for its projects and discover ways to continuously improve its quality.

## Getting Started from Ground Zero

*What does it take to move out of chaos?*

*Why is this so hard?*

*What can MSF provide to help an organization move to level 1?*

## Building a Bridge to Level 1

Those who guide software organizations to use the CMM find it very difficult to make the leap from no sense of discipline to projects that function as level 2 projects. The organization perceives that it has nothing to build on and makes numerous requests for a definition of practices of a level 1 company. Both parties feel a need for something between “not practiced at all” and “repeatable on similar projects.”

## Using Solutions Development Discipline (SDD) Concepts

Using the SDD as a starting point, an organization can leverage the following fundamental concepts:

**Team of peers.** A project is the responsibility of a team of peers, with members representing the key roles required to effectively define and build the product. Each member of the team plays a particular role, but all are responsible for delivering a product that meets customer needs.

**Small teams.** Effective teams tend to be small and focused.

**Milestone-driven process model.** The team uses a simple process model, with phases that are terminated by key milestones.

**Individual commitments.** Each member of the team makes an estimate for the work he or she is doing and the estimates are integrated into the overall plan.

**Deliverables for communication.** Milestones are points of agreement and commitment, with deliverables that are used to communicate the team's understanding at this point. They may evolve in the next phase if necessary.

**Risk-driven scheduling.** Content of releases and the decisions to go to a next phase are driven by a careful analysis of risk factors. Plan revisions are based on understanding the risks of doing and not doing any suggested revision.

**Versioned releases.** Product capability is divided into incremental versions so that users can get needed functionality as soon as possible.

**Multiple interim releases.** Although key milestones are committed externally, the development team arranges its work internally with interim releases that allow it to continuously build the product on known, stable software.

**Iterative analysis and design.** Design is done in progressive layers of detail.

**Specification before code.** Before development proceeds, the customer and the team must agree on a functional specification.

**Bottom-up estimating.** Contributors to the product are responsible for estimating their part of the effort, which will be rolled up into the overall plan.

**Zero-defect mind-set.** The team must make a commitment to quality, enabled by ensuring that what is being developed can be built every day and by eliminating problems early.

**Fixed ship date mind-set.** To help with trade-offs among cost, features, and schedule, the team sets one as most important. Most project teams choose the schedule and make trade-offs between the other dimensions.

## Becoming Teams

### Moving Beyond One-Person Projects

Because most level 1 organizations function as a collection of individuals (not as a team), with everyone doing his or her best, they tend to obtain the best results from one-person projects. One person can be efficient, with quality depending on that individual's ability. Thus, organizations at level 1 who can hire enough heroes to do one-person projects will be successful. But business generally demands projects that are more complicated than one person can build.

When several-person projects are attempted:

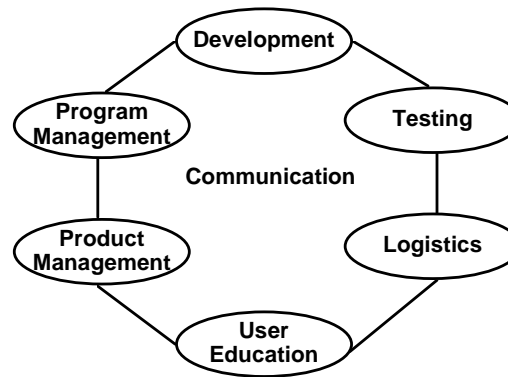
- Components have inconsistent or unusual interfaces.
- Components can be very different in substance and style because of several, rather than one, vision of the work in progress.
- Elements must be done more than once because individuals fail to communicate their intent and don't have defined roles.
- Quality is only as good as the weakest member of the project.
- Estimates of effort are useless because they are based on individual work.
- Plans are sketchy because none were needed for single-person projects.

### Working as a Team

How can the group change this? The key issue is that they work like a group, not like a team. They must adopt a team model that helps them define their roles and responsibilities. They need a way to create and use a common vision of their product. They need ways to communicate with each other about their work, before disagreements and broken interfaces develop.

### Adopting the Team of Peers

The MSF team-of-peers-model provides one good example of fundamental team practices for such a level 1 organization. As the project team evolves, it may modify the model, but as a start, the group can adopt the MSF team roles and responsibilities to create the first semblance of discipline in their work.



The team model has six different, cooperating roles:

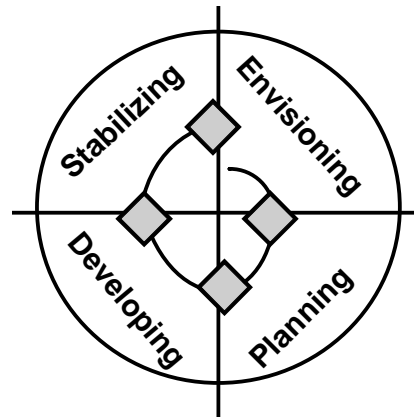
- The **product manager** ensures that the team understands business expectations and that customer requirements and expectations have been defined, manages customer expectations, and manages product launch.
- The **program manager** drives the technical specification of the product and coordinates its development.
- **Developers** design and deliver a product to meet the specification.
- **Testers** independently verify that the product meets requirements and is viable.
- **User education** designs and develops the user documentation and courseware for the product.
- **Logistics management** establishes the environment into which the product will be rolled out and installed and ensures effective migration to operations and support.

### Evolving into a Team

Putting the team in place initially is likely to require fairly strong leadership by one or more team members because the organization has probably had little teaming effort in the past. The team must evolve into a team of peers whose members feel responsible for the project and who help each other deal with meeting team commitments.

Making this move from a directive leadership to a peer relationship is fundamental, not only to MSF but also in other models, such as the People-CMM, Cleanroom Engineering methods, and most total quality models used in today's businesses. In general, having professionals work in teams of peers opens up the greatest amount of productivity and creativity, minimizes hierarchy and structure, and frees individuals to focus on customer needs and building solutions.

## Sharing a Process



Each member of a team needs a simple, shared understanding of what each role contributes to project activities. The process need not be extensive, but it must be communicated. MSF provides a very simple four-milestone development model that can be used by a team to take a project from initial definition to release.

**Activities and milestones of the MSF process model**

Major milestones	Phase of model	Key activities
Vision/Scope Approved	Envisioning	Establishing project scope and user requirements
Project Plan Approved	Planning	Developing functional specification and development plan Drafting design Setting release date
Scope (all deliverables) Complete/First Use	Developing	Completing the design Implementing and testing code Developing documentation Developing training Preparing for beta test
Release	Stabilizing	Completing system testing Completing rollout preparation

### Activities of the Envisioning Phase

Fundamental to the team's success is a shared vision of the product they are building. They should capture this in a set of goals or in a product description in the envisioning phase. Whatever the level of formality, the shared understanding is what drives the team to succeed. In the level 1 organization, this is likely to evolve from simple statements of goals to more carefully crafted descriptions of what the customer needs.

The first step is for the team to realize a common vision shared by all members. Each role has its own special perspective, but everyone on

the team must understand the perspective of the others throughout the development cycle.

- The product manager focuses on matching product results to expressed customer requirements.
- The program manager ensures that the product meets the agreed-on time, cost, and performance parameters.
- Developers work to build what is being asked for so that it is correct and reliable.
- User education develops documentation and training to be sure that the customer can use the product as it is intended.
- Testers ensure that the product does what was designed and is documented.
- Logistics management verifies a smooth product rollout.

Although all team members are involved in creating or reviewing the work products, one role is primarily responsible for that item. For example, the product manager is responsible for writing the vision document, but everyone provides input to the document.

## **Activities of the Planning Phase**

In the planning phase, MSF provides a good basis for developing commitment. Individuals in the development team perform bottom-up estimating and participate in creating the schedule for the next phase to design and develop the product.

### **Writing the Specifications Before the Code**

As the functional specification reaches approval, the customer and team members agree on the deliverables, priorities, and expectations. The functional specification becomes a contract between all parties and each role shares the same vision of the resulting product. The specification is negotiated so that all involved agree on the content and the delivery time frame. Individuals in each role understand what they are to do to develop the product.

### **Incremental Planning to a Fixed Ship Date**

A project team functioning at level 2 is able to build a project plan that covers its full project and incorporates estimates of effort, cost, and schedule; a description of the development process; a full work breakdown structure; a risk management plan; plans for training and equipment for the development team; and details of its implementation approach.

A team functioning at level 1 is unlikely to be able to build that plan without some experience in smaller chunks. MSF encourages incremental planning by the development team, having them focus on

planning the next phase while in the current phase, keeping in mind a simple constraint about the overall schedule—the fixed ship date.

### **Applying Risk-Driven Scheduling**

As the team makes plans for each successive phase, it considers the set of risks ahead. They complete plans to address the risks and revise the plan to reflect those actions. Where feature changes stress the fixed ship date, they make a conscious decision about what features to include and whether to change the ship date. They adjust plans accordingly. They address highest risk items early.

### **Using Versioned Releases**

Many project teams using MSF set a goal of completion in 6 to 9 months because customers require quick delivery on their requests. This generally means that a product will be delivered in increments, with the most important features to the customer being delivered in the current increment. To plan for the next increment, the team asks the customer again to rank requirements so that the most important will be addressed in the next increment.

This approach to incremental development is becoming very common in the industry as a way to meet customer needs on a timely basis and to control the barrage of requirements changes. While the current version is under development, customer needs may change for what has to be in the next version. Waiting until it is nearly time to begin that version to set the content allows the development team to match customer needs more precisely than agreeing on a set of requirements several years in advance of delivery.

Establishing the time frame for the full process helps the project team limit how much it tries to accomplish in a given version. It will take several cycles of development using the MSF process model before a team gets good at such estimation. At first, the team will seek advice from others (or use their own history on related projects) to guess at how much functionality can be adequately planned, developed, and stabilized in six or nine months.

### **Activities of the Developing Phase**

During the developing phase, the team continues the design work started in the prior phase and carries all deliverables to completion. At the scope complete milestone, the software and documentation and training materials are ready for beta test by a customer.

### **Using Iterative Analysis and Design**

Product development generally proceeds through the three phases of design recommended by MSF designing solutions: conceptual design, logical design, and physical design. The team first addresses customer

concerns by creating scenarios and discussing them with the customer. This may happen in the envisioning phase or planning phase, with the scenarios evolving into the objects and services of the product during the developing phase. Finally, the team takes the design to implementation, leveraging components that exist and building new pieces as needed.

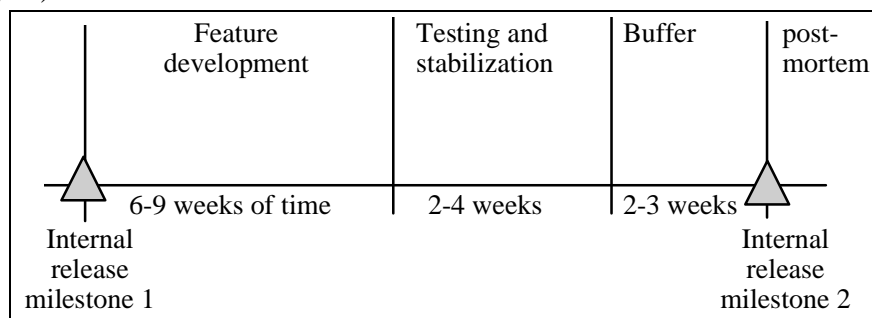
The development may iterate through prototypes that show the intended functionality, causing revisions to one or more layers of the design, before the team establishes the final form of the component.

### Communicating with Deliverables

The team uses the scenarios and other design descriptions throughout product development to communicate understanding of the product to be built. Although each of the phases of the MSF process model has a set of expected deliverables, the team should not freeze those deliverables at that point. Rather, they should attempt to make them as complete as possible, given the current understanding of the customer situation and the product proposed. In many cases, the deliverable will evolve further in the next phase, as the team understands more about the customer requirements or the implementation constraints for the product.

### Employing Interim Releases

The MSF process model advises the team to use internal releases as it develops a given version of a product. Thus, what may be several months of development work is broken into a series of several interim versions, each of which is subject to testing and stabilization. The project team generally builds its schedule by laying out these internal releases, each with a short buffer period (known only internally to the project) to cover the unknowns.



### Attaining a Zero-Defect Mind-Set

The team uses the testing and stabilization period to be sure that the portions of the release that were completed in this interim release work correctly. Because they must often deal with unknowns, they build in a buffer period to deal with those elements.

In addition to the explicit testing effort indicated in this interim cycle, the team consistently works toward a zero-defect status with reviews and unit testing during development, a daily build of the evolving

product, and a focus for all to eliminate problems as soon as they are discovered.

### **Activities of the Stabilizing Phase**

During the final phase—stabilizing—the team ensures that customer expectations for features and performance have been met and that the product is ready for release. Beta customers assist the team in wringing out problems and ensuring that all conditions are right for putting the product into operation and support.

The following table details the work of the team roles involved in each of the milestone-driven phases of the MSF process model.

Responsibilities of team roles in the MSF process model

Role	Vision Approved	Project Plan Approved	Scope Complete	Release
Product management	Writing vision document	Contributing to conceptual design	Managing customer expectations	Coordinating beta site and product launch
Program management	Developing design goals, success factors, metrics, and solutions concept Establishing project infrastructure	Drafting the functional specification Specifying logical design (in business terms) Building plan and schedule for the next phase Setting the ship date	Managing the specification Tracking project, communicating status Preparing the beta plan Coordinating usability testing	Tracking status and coordinating release schedule
Development	Consulting on other elements	Evaluating technologies Participating in physical design Developing proofs of concept or implementation prototypes Estimating effort on tasks and defining schedule	Developing product elements, demos, and operational prototypes Building internal releases Optimizing code Contributing to usability testing	Repairing defects Cleaning up the paperwork (getting specifications and design documents up to the level of product released)
User education	Establishing strategy for performance system, visual design, and training	Evaluating design for performance system Planning and scheduling documentation and online delivery work	Building and reviewing documentation, graphics, and course materials Contributing to usability testing	Delivering training Managing performance system baseline
Testing	Evaluating vision/scope statements	Evaluating design Planning testing and developing schedule for next phase Evaluating development plan and schedule for testing	Executing tests and reporting results Verifying bug fixes Contributing to usability testing Verifying performance of team against schedule	Testing beta release and final release candidates for readiness Performing configuration testing
Logistics management	Identifying deployment concerns	Evaluating design Developing plan and schedule for rollout	Building operations and support guidelines Building final release schedule	Supporting beta sites and managing rollout Managing release process
All members	Defining risks	Updating risks	Updating risks	Updating risks

## Evolving to Higher Maturity with MSF

*How do we move to level 2?*

*What can we leverage to get to level 3?*

### Moving from Level 2 to Level 3

Many elements of MSF can support organizations evolving to CMM levels 2 and 3. The ability of organizations to use those elements may be limited if they are starting from an environment essentially free of discipline. The previous section addressed how to get started in an essentially level 1 organization and to evolve practices that make it a level 2 organization.

Getting to level 2 requires that project teams control their commitments, build sound plans, and follow those plans. In particular, project teams in a level 2 organization must meet the goals described in the following paragraphs. Support from application development principles is useful, as indicated in the table later in this section.

#### Requirements Management

A project team must establish and maintain a documented baseline of user requirements against which the project plans and products will be built. Throughout the development process, the team should be able to show how the work in progress maps to the user requirements, perhaps with a trace matrix. When requests are made to include new features or change elements of the requirements, the team must examine them in light of the impact on the plan and schedule, making changes only if it can reach an agreement on the impact of the change.

Deliverables of the MSF process model for application development that support requirements management include the vision document, the functional specification, and test plans. Completing the functional specification includes initial agreement on functionality with the customer and calls for plan revision (with agreement of the customer) when changes are necessary. The team can use test results to show that requirements are met, along with tracing forward each element of the functional specification to its final implementation.

#### Project Planning

An adequate project plan for a project team is one that provides it with a roadmap for doing the work of the project. The team must estimate the project effort, size, and cost, and document all project activities and commitments to customers and other external organizations. All of those contributing to the project must understand and agree to their commitments.

The team generally captures this information in a project plan, which all team members participate in creating and keeping up to date. The approach in the MSF is an incremental one, having team members establish portions of the plans for which they have the relevant data as the project proceeds. This approach implies that the final agreement on content and the plan for delivering the product may not be known until the developing phase, although a high-level plan built during the planning phase allows the team to commit to a ship date. If this meets customer needs, the approach is practical, encouraging the team to make the best use of the information it has. If a customer must have guarantees of certain deliverables by a certain date for legal or other reasons, the team might need to be more rigorous about having a very defined schedule in the planning phase. Such an approach requires that the requirements be extremely well defined as well.

### **Project Tracking and Oversight**

As the project proceeds, the team must track its ability to meet commitments and to execute the planned activities. If it deviates significantly from the plans, it must take action to change the plans and gain agreement on the changes from all who are involved in the project, including the customer.

The team needs regular progress reviews with both project management and quality assurance. These reviews can be more or less formal, but they should be conducted regularly throughout the project cycle so that any problems are handled as soon as possible. MSF proposes that there be regular team status meetings or reports as well as progress reviews at the milestones to monitor progress. The MSF focus on risk-driven scheduling provides a careful focus for the team to analyze its prospects for the next part of the work and to review what has happened in the most recent effort. This requires the team to think about its performance level thus far and to be honest about the most likely performance for the next work segment. In addition to schedule and plan modifications, the post-implementation reviews at each milestone advised by MSF also prompt the team to consider alternate ways of doing its work, which may affect the plan and schedule.

### **Software Configuration Management**

The team must plan activities for software configuration management at the start of the project as well as processes that ensure it can identify and control work products. It must define a process for managing and controlling changes to requirements and work products and informing people affected by the project work about the status and content of the product baselines.

The work products that are covered by configuration management include specifications and plans, as well as all software and documentation and training materials. Each of these must be in a known state, with a record of approved changes when the item has been baselined. The team must carefully track problem reports, change requests, agreements to make changes, and results of making changes so that what is in the product is what the team intended to build.

When the team establishes a baseline, it must inform those involved in using it. Such occasions include the agreement on a functional specification or agreements on changes to that requirements baseline as well as baselines of product components being passed on to an internal organization for integration test or to a customer for beta test. Configuration management means the team has control over its work products, knowing that what is built is using the intended components, knowing what has changed at what point for what reason, and being able to track that it has addressed all known problems.

A team using the MSF process model is advised about source control, tracking defects, and building the product, but the team must also develop a set of plans for configuration management and specify its change control process.

### **Software Quality (Process) Assurance**

To be sure it is following its process and producing the intended product, the project team needs someone to play a role of process assurance. The work includes establishing a plan for quality assurance activities, ensuring that the project follows its plan (and standards and procedures cited there), ensuring that the product has the planned features and level of quality specified, and raising issues of noncompliance in any of these areas with the team (and, if needed, with management).

Process assurance can be done by an independent group, by a member of another project team, or by a member of this project team, depending on the organizational culture and customer needs. MSF does not advise how to do quality assurance, so the project team using the MSF process model must determine how to ensure the intended work is being done, to build a quality product of the level intended.

### **Software Subcontract (Vendor) Management**

Project teams that outsource some of their development must manage vendor work as carefully as they manage their own work. They must ensure that the vendor is qualified to do the subcontracted work, that they have established requirements and made commitments and

documented them in a project plan, that they continue ongoing review and communication during product development, and that they evaluate results of the vendor against the commitments.

MSF is silent on subcontracting, so project teams that vend part of their work must develop a process for managing subcontracts.

### **Leveraging MSF for Level 3**

An organization that has used elements of MSF as a means for moving to level 2 is likely to have different kinds of work being done in different ways in its projects. Moving to level 3 implies that the organization has adopted a set of best practices as its preferred processes. It expects each project team to make use of those best practices, tailoring them as appropriate. The practices include evolving the project management practices of level 2, as well as product development practices.

#### **Organization Process Focus**

To do an effective job of identifying and using best practices, organizations must establish a group with that responsibility and build a plan for how the organization will improve its processes. Such a plan should include periodic assessment of the organization's process maturity, leading to plans for improvement in capability. In most organizations, process engineering is done by a centrally supported Software Engineering Process Group (SEPG), which looks out for the interests of every project in the organization.

MSF focuses at a project level and does not directly deal with the organization process focus key process area.

#### **Organization Process Definition**

The SEPG, working with individuals in the organization, defines a standard approach and one or more life cycles for developing and managing software in the organization. It bases this definition on the best practices that have evolved in the organization's projects, supplemented by industry best practices. It makes these descriptions, along with supporting templates, checklists, and other job aids, available to all projects in the organization. Users of the materials are invited to provide feedback on improvements they make or on changes they think necessary. In addition, software quality assurance personnel monitor the usefulness of the organization's processes and ensure that shortcomings are addressed.

In addition to the processes definitions, the SEPG also establishes and maintains a database of information and examples about the organization's use of the processes. Thus, a new team can review records of project estimates and their actual performance when it is estimating its work, can examine project plans on past projects for

ways to do the plan, and can study previous methods of designing or implementing a product for appropriate techniques. Teams can leverage lessons learned on previous projects when those projects are at a similar stage.

An SEPG that is building up a defined process can leverage MSF, as well as templates and examples of work products provided with MSF. The work products of teams that follow SDD provide good material for the process use database.

### **Integrated Software Management**

A project uses the organization's defined processes as the basis for building its project plans and defining its activities. It tailors the organization process according to its needs, and then follows that process for managing the project work.

The philosophy of MSF is that project teams tailor a process for their projects, based on the principles and concepts of the MSF process model. This is very much the philosophy of the CMM for this key process area. Assuming that the organization has gathered the best level 2 practices into its defined process, each team is required to start from that definition, tailoring to its specific project needs and objectives. Software quality assurance (SQA) personnel (or the SEPG) should advise the team on questions or issues they may have with adapting the process.

If the team can't find an easy way to tailor the process, it may need to develop its own extensions or process, with the collaboration of SQA or the SEPG. If the work is representative of other projects that will appear in the organization, such a new definition may become an alternative defined process for the organization.

### **Software Product Engineering**

Like its project management processes, the organization documents one or more product development processes. Project teams then use those development processes to establish the approach they will take for creating their product. The approach must lead to effective requirements analysis, design, implementation, and testing of the product elements. Work products must be kept consistent with each other as the project proceeds so that changes in one are comprehended in others that are affected.

MSF recommends a collection of product development concepts and practices that can form a foundation for a defined development process. To develop specifications, the team can analyze methods using scenarios. Many development teams can adopt the three-level design approach. The MSF process model for application

development advises the types of testing and test planning. MSF recommends using such methods as prototyping, usability analysis, and code reviews, each of which can be useful in developing the organization's preferred development practices.

Integrating user documentation and training development into the cycle of product development ensures that work products in these areas will be synchronized with the software.

Each organization using MSF is likely to have its own collection of best practices for developing products that will be included in the organization's defined development process. The organization should include those that contribute to quality results and make the selections for a specific project after considering the needs of that project. It is unlikely that one single process will apply in all cases; it is more likely that the team will draw from a set of practices to tailor a practical approach. Again, SQA personnel or the SEPG can advise a team on which practices are likely to be most useful to its project.

### **Intergroup Coordination**

When the project effort involves more than one development group, all groups should collaborate effectively on the project. This requires that all groups know and agree to the customer requirements for the project, that the groups document and agree to their commitments to the project, and that the groups have a process for working together. This process provides ways for them to plan their work, communicate about progress, and identify and resolve issues.

MSF does not explicitly consider projects that involve more than one group, so organizations cannot leverage practices from here, other than generalizing the methods for working with one team.

### **Peer Reviews**

When building its project plan, the team must decide what types of formal work product reviews will be done and include time for those reviews in its plan. Then, throughout the project, the team must follow those plans, identifying and removing defects in the work products.

The particular types of peer reviews used can vary widely from organization to organization and from project to project. Most project teams find it useful to do reviews on work products throughout the cycle, especially focusing on areas that are complex or confusing. Although code reviews are a common practice, even more can be gained by reviewing work products early in the life cycle. Errors in the functional specification or in design can be very expensive to fix if they are not found until code is developed or implemented.

The MSF process model recommends reviewing work products in each phase, especially focusing on the vision document, the functional specification, and code. Organizations are advised to use methods of review that have reviewers examine the materials before gathering as a team to identify problems. In this way, team time becomes much more productive than if the team gathers to see the work product for the first time as a group.

### Training Program

A level-3 organization provides training for its members, so that everyone can effectively play the roles they assume. It devises an organizational plan, as well as project-level plans, for how to deploy training to grow the skills and abilities of members of the organization. It provides training for developers and managers so that they understand the processes, technologies, and domain of the work they do. It keeps records of the training taken and of effectiveness of provided training, to ensure that programs are useful.

MSF does not explicitly address training, but the concepts and practices of MSF can form part of the training program to establish its development and management processes. In general, the organization should assess the competencies needed for its business, determine what skills and abilities must be developed in its people, and acquire or develop the required training.

Support for CMM levels 2 and 3 from MSF

CMM Level	Key Process Area	Application Development Phase	Activity
2	Requirements management	Envisioning	Establishing understanding of scope with customer
		Planning	Tracing features in functional specification to user activities and tasks
		Developing	Modifying product vision and functional specification if requirements change or are better defined
		Stabilizing	Checking that requirements are met in beta test
2	Project planning	Envisioning	Identifying risks Establishing trade-offs (resources, schedule, and features) Establishing the project structure Outlining the solutions concept (project success factors, operational scenarios, deliverables, and acceptance criteria) Setting level of effort and schedule for next phase
		Planning	Estimating effort for development Building schedule and plan for development and setting ship date Building schedule and plan for user education

			elements <b>Building schedule and plan for testing effort</b>
		<b>Developing</b>	<b>Building beta test plan</b>
		<b>Stabilizing</b>	<b>Planning hand-off to operations and support team</b>
<b>2</b>	<b>Project tracking and oversight</b>	<b>Envisioning</b>	<b>Establishing status and team meetings</b>
		<b>Planning</b>	◇
		<b>Developing</b>	<b>Gathering progress information</b> <b>Analyzing status information</b> <b>Conducting reviews at milestones</b> <b>Reviewing and updating schedule based on risks and known schedule variances</b> <b>Doing post-implementation reviews on internal releases</b>
		<b>Stabilizing</b>	<b>Monitoring progress to plans</b> <b>Conducting project post-implementation review</b>
<b>2</b>	<b>Software configuration management</b>	<b>Envisioning</b>	<b>Establishing a formal location for the functional specification</b> <b>Establishing source control procedures</b>
		<b>Planning</b>	<b>Identifying build process and interim releases</b> <b>Identifying source library management mechanisms</b>
		<b>Developing</b>	<b>Building interim releases</b> <b>Maintaining and managing database of defects</b>
		<b>Stabilizing</b>	<b>Establishing golden release for manufacturing</b>
<b>2</b>	<b>Software quality assurance</b>	<b>Envisioning</b>	<b>Defining a desired level of quality (basis for a quality plan)</b>
		<b>Planning</b>	◇
		<b>Developing</b>	◇
		<b>Stabilizing</b>	◇
<b>2</b>	<b>Software subcontract (vendor) management</b>	<b>(all phases)</b>	<b>N/A</b>
<b>3</b>	<b>Organization process focus</b>	<b>(all phases)</b>	<b>N/A</b>
<b>3</b>	<b>Organization process definition</b>	<b>Developing, stabilizing</b>	<b>Identifying issues during post-implementation reviews that can affect the next project</b>
<b>3</b>	<b>Integrated software management</b>	<b>Envisioning</b>	<b>Tailoring the application development principles to this project</b> <b>Documenting risk assessment</b>
		<b>Planning</b>	<b>Documenting risk assessment and risk mitigation approach</b>
		<b>Developing</b>	<b>Risking assessment and mitigation done at each interim release</b>
		<b>Stabilizing</b>	◇

3	Software product engineering	Envisioning	Requirements. Identify user profile
		Planning	Requirements. Analyzing user needs and developing functional specification Design. Reviewing implementation options and drafting design in three levels (conceptual, logical, and physical) Test. Developing test plans, test cases, test data, and beta criteria Test. Identifying environments and tools Documentation. Planning user documentation
		Developing	Design. Completing three-level design into program design Code. Developing the product code and performing unit test Test. Performing integration and other tests Documentation. Developing and reviewing help, documentation, and courseware
		Stabilizing	Testing all elements of the product, documenting problems, and repairing defects Conducting beta test Finalizing internal product documentation (getting specifications and design documents to level of released product)
3	Intergroup coordination	(all phases)	With team of peers, planning and developing the product
3	Peer reviews	Envisioning	Customer and team reviewing vision statement
		Planning	Customer and team reviewing the functional specification
		Developing	Reviewing code
		Stabilizing	<>
3	Training program	(all phases)	<>

Table Notes: N/A = Element of the CMM not applicable to development model like SDD

<> = No element in principles of application development for this KPA in this phase

## Discovering CMM and MSF Elements

*How does MSF augment the CMM?*

*How does the CMM augment MSF?*

### MSF Elements Outside the CMM

The MSF process model includes some activities that are outside the CMM, as do many development processes. The CMM does not explicitly consider developing user requirements, establishing a business case, developing marketing materials, or deploying the

solution. MSF covers all of these, with a concerted effort on preparing the logistics and plan for rollout of the product into operation.

### **CMM Elements Not in MSF**

The CMM includes many best practices that are not a part of MSF. This is to be expected because MSF is an evolving collection, gathered from organizations gradually increasing their process maturity. MSF is a collection of principles and concepts that represent what is known and done at Microsoft today, and this collection continues to grow as the organization matures and learns from its own experience and that of others. In addition, because the *Principles of Application Development* course focuses on a process for running a project, it does not address organizational improvement or establishing organizational processes, as the CMM does.

## Conclusions

*What have we covered?*

*Where do we go next?*

In this document, we have examined the practices and concepts of two models of best practices—the SEI CMM and Microsoft’s MSF. As organizations attempt to move from a state of software crisis, they can leverage material from both of these models to establish sound development processes. The models offer complementary support, with MSF being especially useful to give an organization foundation concepts for moving to level 2 in process maturity. That foundation, in turn, provides excellent support for level 3 and beyond.

For further information on the CMM, consult the book by Paulk et. al cited in the references. Contact the Microsoft Solutions Framework team for further information about MSF.

## References

Cusumano, Michael A., and Richard W. Selby. *Microsoft Secrets*. NY: The Free Press, 1995. This book includes two chapters on product definition and development at Microsoft, a summary based on interviews between March 1993 and September 1994 with managers and key developers of both application and systems products at Microsoft.

Garvin, David A. “Leveraging Processes for Strategic Advantage,” *Harvard Business Review*, September-October 1995, 77–90.

Maguire, Steve. *Debugging the Development Process*. Redmond, WA: Microsoft Press, 1994. In this book, Maguire describes techniques and strategies for developers and team leaders to organize and run projects. This is essentially a process like the *Principles of Application Development*, with tips and stories about how to keep the process lean and aimed at the project goals.

McCarthy, Jim. *Dynamics of Software Development*. Redmond, WA: Microsoft Press, 1995. In this book, McCarthy provides 54 rules of thumb for shipping great software, examining the life cycle from understanding the market to shipping the product. The overriding concern is building and leading an intellectually engaged development team.

“Managing Software Milestones at Microsoft,” *American Programmer*, Volume 8, Number 2, February 1995, 28–37. This article, an extract from his book, describes characteristics of milestones (phase or intermediate segments of the developing phase in *Principles of Application Development* terms). He cites six events that occur within most such milestones.

Paulk, Mark C., Charles V. Weber, Bill Curtis, and Mary Beth Chrissis. *The Capability Maturity Model, Guidelines for Improving the Software Process*. Reading, MA: Addison-Wesley, 1995.

Sherman, Roger W. "A View of Development and Testing at Microsoft," *American Programmer*, Volume 8, Number 2, February 1995, 15–21. With a focus on the testing activities, this article describes the development process (essentially *Principles of Application Development*) and the primary deliverables of each phase. Controlled work on interim releases is accented as a way to ensure reliability of the product, along with a range of testing approaches (tools, automated testers, regression suites, technical beta tests) and careful tracking of defects.

Smith, Stanley A., and Michael A. Cusumano. "Beyond the Software Factory: A Comparison of 'Classic' and PC Software Developers," *MIT Sloan School WP#3607-93\BPS*. September 1, 1993. This paper examines development practices at IBM, Fujitsu, Hewlett-Packard, Microsoft, and Lotus. In the case of Microsoft, it discusses the challenge of implementing more structured development processes while maintaining the creative technical edge needed in the PC software industry.

© 2001 Microsoft Corporation. All rights reserved.

The information contained in this document represents the current view of Microsoft Corporation on the issues discussed as of the date of publication. Because Microsoft must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information presented after the date of publication.

This document is for informational purposes only. **MICROSOFT MAKES NO WARRANTIES, EXPRESS OR IMPLIED, IN THIS DOCUMENT.**

Microsoft and MS are either registered trademarks or trademarks of Microsoft in the United States and/or other countries.