Microsoft
Solutions Framework

Team Model for Application Development

CloverLink Systems Inc. utilizes a Team-Oriented Development Methodology, providing expertise in each area of application development. From Product and Project Management to Architecture and Design Decisions, we empower and train our Clients through teamwork to ensure the success of both the development effort and the long-term maintenance of the solution.

CloverLink SYSTEMS, INC.

Your Link to Tomorrow's Business

1486 Sunshine Drive • Glendale, CA • 91208 • www.CloverLink.com
e-mail: Sales@CloverLink.com
800.378.8348

MSF Team Model for Application Development

Contents

Introduction

Microsoft Solutions Framework (MSF) is a framework rather than a methodology that can be adapted to suit the particular needs of an organization. The team model for application development is one aspect of this framework. It describes how teams should structure themselves and what principles they should uphold in order to be successful at developing software.

The team model is specific in nature, but as part of a framework it should be viewed as a starting point. Individual project teams may implement aspects differently, depending on the scope of the project, the size of the team, and the skills of the team members.

Quality Goals of an Effective Team

Overview

MSF is based on the belief that six key quality goals must be achieved in order for a project to be considered successful. These goals drive the team and define the team model:

- Delivery within project constraints.
- Satisfied customers.
- Delivery to specifications based on user requirements.
- Release after knowing and addressing all issues.

- Enhanced user performance.
- Smooth deployment and ongoing management.

Delivery Within Project Constraints

A key goal for all teams is to deliver within project constraints. The fundamental constraints of any project include those of budget and schedule. Most projects measure success using “on time, on budget” metrics.

Satisfied Customers

Projects must meet the needs of customers and users in order to be successful. It is possible to meet budget and time goals but still be unsuccessful if customer needs have not been met.

Delivery to Specifications Based on User Requirements

The product specification describes in detail the deliverable to be provided by the team to the customer. It is important for the team to deliver in accordance with the specification as accurately as possible because it represents an agreement between the team and the customer as to what will be built.

Release After Knowing and Addressing All Issues

All software is delivered with defects. A key goal is to ensure those defects are identified and addressed prior to releasing the product. Addressing can involve everything from fixing the defect in question to documenting work-around solutions. Delivering a known defect that has been addressed along with a work-around solution is preferable to delivering a product containing unidentified defects that may surprise the team and customer later.

Enhanced User Performance

In order for a product to be successful, it must enhance the way that users work and perform. Delivering a product that is rich in features and content but is not usable by its designated end user is considered a failure.

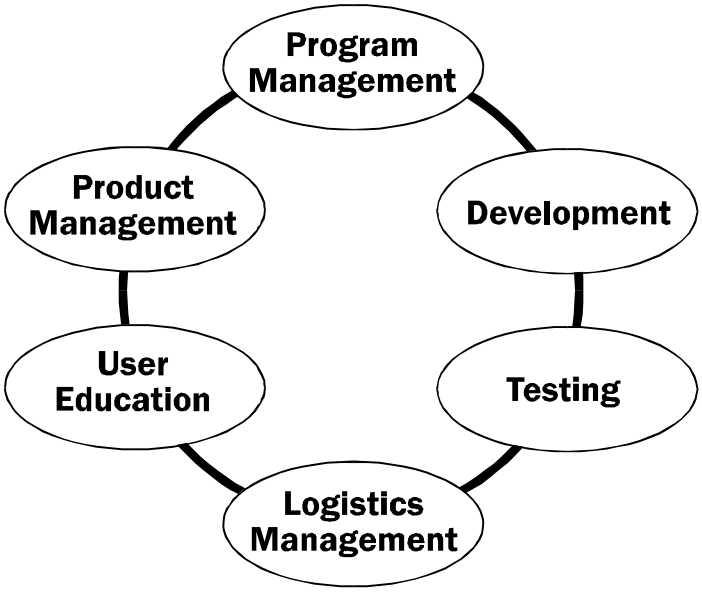
Smooth Deployment and Ongoing Management

Sometimes the need for a smooth deployment is overlooked. The perception of a deployment is carried over to the product itself, rightly or wrongly. For example, a faulty installation program may lead users to assume that the installed application is similarly faulty, even when this may not be true. Consequently, the team must do more than simply deploy; it must strive for a smooth deployment and prepare for the support and management of the product. This can include ensuring that training, infrastructure, and support are in place prior to deployment.

Team Model to Achieve Quality Goals

MSF Team Model

The MSF team model for application development is defined as a small team of peers working in interdependent multidisciplinary roles. The following is a depiction of the team model:



MSF Team Model for Application Development

The team model addresses the need to achieve the key goals outlined in the previous section. Each goal requires a different discipline and each discipline is embodied by a team role. The people who fill these roles bring a unique perspective to the requirements needed to meet each goal.

The six roles of the team model correspond directly with the six key quality goals for an effective project team. They map as follows:

Goal	Team Role
Satisfied customers	Product management
Delivery within project constraints	Program management
Delivery to product specifications	Development
Release after addressing all issues	Testing
Enhanced user performance	User education
Smooth deployment and ongoing management	Logistics management

Having a clearly defined role and owning a clearly defined goal increases understanding of responsibilities and encourages ownership by the project team, which ultimately results in a better product. Since each goal is critical to the success of a project, the roles that represent these goals are seen as peers with equal say in decisions.

The MSF Roles

Product Management Role

The goal of product management is customer satisfaction. The product management role is positioned to achieve this by acting as the customer advocate to the team and as the team advocate to the customer. It is important to distinguish between the customer and the end user—the customer is the one who *pays* for the product while the end user is the one who *uses* the product.

As the customer advocate to the team, product management is responsible for understanding customer requirements, creating the business case, establishing the shared project vision between team and customer, and ensuring that any solution that the team develops meets the needs of the customer by solving their particular business problem.

As the team advocate to the customer, product management is responsible for high-level communications and managing customer expectations. High-level communications include public relations, briefings to senior management/customers, marketing to users, demonstrations, and product launches. Managing expectations is the key role of product management once the vision is set. It is considered to be a primary role because it can determine the difference between success and failure.

The importance of effectively managing expectations can be illustrated with an example involving the anticipated delivery of 10 product features from a team to a customer by a certain date. If the team delivers only two features when the customer expects the delivery of all 10, the project will be deemed a failure both by the customer and by the team.

If, however, product management maintains constant two-way communication with the customer during the feature development and production period, changes can be made with regard to customer expectations that can ensure success. Product management can include the customer in the tradeoff decision-making process and inform them of changing risks and other challenges. Unlike the previous scenario, the customer can assess the situation and agree with the team that delivery of all 10 features within the specified time frame is unrealistic and that delivery of only two is acceptable. In this scenario, the delivery of two features matches the expectations of delivering only two, and both parties will consider the project a success.

Program Management Role

The role and focus of program management is to meet the quality goal of delivering the product within project constraints. To meet this goal, program management owns and drives the schedule, the features, and the budget for the project. Program management ensures that the right product is delivered at the right time.

As the owner of the schedule, program management collects all team schedules, validates them, and integrates them into a master schedule that is tracked and reported to the team and to stakeholders.
As the owner of the features, program management plays a role in defining which features will be delivered in order to meet the requirements outlined by product management. Program management owns the functional specification (what is to be built) and the master project plan (how it is going to be built) and facilitates their creation by getting input from each of the roles on the team. It is essential that each role contributes input, perspective, and sign-off on the functional specification and project plan. The features are then tracked against the functional specification and their status is reported to the team and to stakeholders. As the owner of the budget, program management facilitates the creation of the planned cost by gathering resource requirements from all of the roles on the team. Program management must understand and agree with all resource decisions (hardware, software, people) and must track the budget actuals against the plan. The team and key stakeholders receive status reports.
In addition, program management coordinates resources, facilitates team communication, and drives critical decisions where consensus cannot be achieved.

Development Role

To succeed in meeting its quality goal, the role of development is to build a product that meets the specification and customer expectations. It is important that development focus not only on coding to the functional specification but also on meeting customer expectations. This is because functional specifications are written before any significant development or building take place, leaving them inherently incomplete. Therefore, development must innovate, but only to solve the customer's problem, not just for the sake of implementing interesting features.
Development serves the team as technology consultants and as product builders. As technology consultants, development must provide input into high-level designs, evaluate technologies, and develop proof-of-concept prototypes to validate potential solutions and to mitigate development risks early in the development process. As builders, development provides low-level product and feature design, estimates the effort required to deliver on that design, and then builds the product.
Development estimates its own scheduling because it works daily with all developmental contingency factors. MSF refers to this concept as bottom-up estimating. Its goal is to achieve a higher quality of schedule and to increase accountability of the estimates and of the work.

Testing Role

The goal of testing is to make sure that all issues are known and addressed prior to releasing the product. An issue is anything that

prevents the product from meeting its requirements. This could be a fault in the code that development writes, otherwise known as a bug, a deviation in the specification that program management owns, or a defect with the documentation that user education produces.

To achieve their quality goal, testing is responsible for "reality induction," as Jim McCarthy says in *Dynamics of Software Development*. The testing role must be able to clearly articulate what is currently wrong with the product and what is currently right with it so that the status of product development is accurately portrayed. To do this, testing must have a very good grasp of the needs of the users and a clear understanding of what the product will do to meet those needs.

To facilitate the testing process, testing develops test strategies, plans, schedules, and scripts. This helps ensure the team's understanding of what, how, and when something is going to be tested.

It is important to distinguish between testing and quality assurance (QA). Testing has a project focus and involves detailed technical work. QA, on the other hand, is often a corporate function organized under a director of quality, whose responsibility is process compliance with corporate or regulatory standards. QA may also be responsible for sharing best practices across project teams.

User Education Role

User education focuses on enhancing user performance so that users are as productive as possible with the product. To accomplish this, user education acts as the advocate for the end user of the product, much like product management acts as the customer advocate to the team.

As the user advocate to the team, user education participates in the design process to deliver a product that is useful, usable, and in need of as little performance support material as possible. This also lowers the costs of supporting the product in the operations/delivery channel. Where user performance support materials are still required, user education designs, builds, and tests materials that will enable easier use. These materials can include reference cards, keyboard templates, user manuals, on-line help, wizards, and even full-featured courseware.

User education also is responsible for usability testing, tracking usability issues, and ensuring that those issues are addressed in the product design. User education must also ensure that changes in scope and design are well known and reflected in any of the relevant performance support materials.

Logistics Management Role

Logistics management serves as the advocate for the operations, product support, help desk, and other delivery channel organizations in its focus on smooth deployment and ongoing management.

As the operations advocate to the team, logistics management participates in the design process to help ensure that the product is

deployable, manageable, and supportable. Logistics management also is responsible for understanding the product's infrastructure and support requirements and ensuring that installation sites have met those requirements prior to deployment. Typically, this is facilitated through the creation and implementation of rollout, installation, and support plans.

Other logistics management activities include supporting the logistical requirements of the team, supporting the product throughout the beta testing process, and providing training to the operations and help desk personnel.

The MSF Team Model Is NOT an Organization Chart

One question that often arises when applying the MSF team model is: "Who is in charge?" An organization chart describes who is in charge and who reports to whom. In contrast, the MSF team model describes key roles and responsibilities for a project team, but does not define the management structure of the team from a personnel administration perspective. In many cases, the project team includes members from several different organizations, some of whom may report administratively to a different manager.

Team Principles

Overview

There is more to having a successful team than just defining roles and responsibilities. Along with team structure, there must also be some underlying practices and principles that help to ensure the success of the team. The following is a list of best practices and principles that have helped to make the team model a success internally at Microsoft and with customers and partners who have implemented MSF. These include but are not limited to:

- Shared project vision
- Team of peers
- Product mindset
- Zero-defect mindset
- Understanding the business
- Overlapping roles and shared responsibilities
- Total participation in design
- Learning from current and past projects
- Team members working together at one site

Shared Project Vision

Fundamental to the success of a project is that team members and the customer have a shared project vision—that is, a clear understanding as to what the goals and objectives are for the project. Team members and customers all bring with them assumptions as to what the project

and the product are going to do for the organization. The vision brings those assumptions to light and ensures that all project participants are working to accomplish the same goal.
The vision should be formalized into a vision statement that forms the basis for the first project deliverable. An effective vision statement:

- Describes not only what the product is but also what the product is not.
- Enables decisions (features are put in or left out based upon the vision).
- Motivates the team to achieve it.
- Is achievable so that the team and customer can commit to achieving it.

Without a shared vision, team members will have competing visions, making it much more difficult to deliver as a cohesive group. And if the team does deliver, members will have difficulty determining their success because it depends on which vision they measure it by.

Team of Peers

The team of peers concept places equal value on each role. This enables unrestricted communication between the roles, increases team accountability, and reinforces the concept that all six quality goals are equally important and all must be achieved. To be successful with the team of peers, all roles must have ownership of the product's quality, must act as customer advocates, and must understand what business problem they are trying to solve.
While each role has an equal value on the team, the team of peers exists between roles and should not be confused with consensus-driven decision making. Each role requires some form of hierarchy for the purposes of distributing work and managing resources. Team role leads for each role are responsible for managing, guiding, and coordinating the team while team members focus on meeting their individual goals.

Product Mindset

The product mindset is not about whether you ship commercial software products like Microsoft or develop applications for internal customers. It is about treating the results of your labor as a product. The first step to achieving a product mindset is to look at the work that you are doing as either a project by itself or contributing to a larger project. In fact, MSF advocates the creation of project identities so that team members see themselves less as individuals and more as members of a project team. One technique Microsoft uses to accomplish this is to give projects code names (such as "Chicago" for Windows® 95). This helps to clearly identify the project, clearly identify the team, raise the sense of accountability, and serve as a mechanism for increasing team morale. Printing the team project code

name on T-shirts, coffee mugs, and other group gift items are ways to create and reinforce team identity and spirit.

Once you understand that you work on projects, it's just a matter of understanding that whatever the final deliverable is, it should be considered a product. Principles and techniques that apply to creating products, like those advocated in MSF, can be used to help ensure your project's successful delivery.

Having a product mindset also means being more focused on execution and what is being delivered at the end of the project and less focused on the process of getting there. That doesn't mean process is bad or unimportant, just that it should be used to accomplish the end goal and not just for the sake of using process. With the adoption of the product mindset, everyone on the team should feel responsible for the delivery of that product.

Former Microsoft program manager Chris Peters describes the product mindset as applied to software development in the following excerpt from a 1991 presentation:

"Everybody … has exactly the same job. They have exactly the same job description. And that is to ship products. Your job is not to write code. Your job is not to test. Your job is not to write specs. Your job is to ship products. That's what a product development group does.

"Your role as a developer or as a tester is secondary. I'm not saying it's unimportant—it's clearly not unimportant—but it's secondary to your real job, which is to ship a product.

"When you wake up in the morning and you come in to work, you say, 'What is the focus—are we trying to ship or are we trying to write code?' The answer is, we are trying to ship. You're not trying to write code, you're trying not to write code."

Zero-Defect Mindset

In a successful team, every member feels responsible for the quality of the product. Responsibility for quality cannot be delegated from one team member to another team member or function. Similarly, every team member must be a customer advocate.

Zero-defect mindset is a commitment to quality. It means that the goal of the team is to perform their work at the highest quality possible, so that if they have to deliver tomorrow, they can deliver something. It's the idea of having a nearly shippable product every day. It does not mean delivering code with no defects; It means that the product meets or exceeds the quality bar that was set by the team during envisioning.

The analogy that best describes this concept is that of the automobile assembly line. Traditionally, workers put cars together from individual parts and were responsible for their own quality. When the car rolled off the line, an inspector checked it to see if its quality was high enough to sell. But the end of the process is an expensive time to find all of the problems because corrections are very costly at this point. Also, since the quality was not very predictable, the amount of

time required at the end to determine if it was sellable was not predictable either.

More recently in car manufacturing, quality has become "job one." That means that as work is being done (such as attaching a door or installing a radio), an inspector checks the work to make sure that it meets the quality standards that are defined for that particular car. As long as this level of quality continues throughout the assembly process, then much less time and fewer resources are required at the end to ensure that the car is of acceptable quality. This makes the process much more predictable because the inspector needs to check only the integration of the parts, and not the individual work.

Understanding the Business

It is not enough for a team to be technically competent in the ways of software development to be successful. Many a "great" product did not meet the business objectives of the customer and, though well written and technically sound, was shelved. To prevent this, team members need a clear understanding of the business problem that they are trying to solve.

One way to accomplish this is to have active customer participation and feedback throughout the development process. This includes their participation in establishing the product vision, signing-off on what is going to be built, taking part in trade-off decisions, and adding feedback through usability studies and test beta releases.

Another method is to continually ensure that every feature in the product design is traceable back to a requirement outlined by the customer or user. If a feature does not trace back, then the team should question whether to spend development time on it.

Overlapping Roles and Shared Responsibilities

As Chris Peters humorously put it in a 1991 presentation:

"It's extremely important to move responsibility very low in the organization. Your goal is not to be working on a project where you can't sleep at night. Your goal isn't to have it so that the project leads can't sleep at night. Your goal is so that nobody sleeps at night. And when nobody is sleeping at night, you have pushed responsibility to the proper level."

To encourage team members to work closely with each other, they are given broad, interdependent, and overlapping roles, and all share responsibility for shipping the right product at the right time. This approach discourages specialization among team members, which often leads to isolated, rather than collaborative, effort.

Total Participation in Design

Former Microsoft program management team director Jim McCarthy summarizes the concept in *Dynamics of Software Design*:

"The goal in any product design is to have the best ideas become the basis of the product. Everybody on the team should be working to accomplish that goal."

Each role participates in the product specification because each role has a unique perspective of the design as they try to ensure that the design meets their individual objectives, as well as the team's objectives.

Learning from Current and Past Projects

For project success to be more than just luck, a work environment must encourage teams to make a deliberate effort to learn from current and past project successes and failures. Creating a learning organization is fundamental to ongoing improvement and the continued success of projects.

One method for institutionalizing this type of behavior is to use post-milestone reviews, or postmortems. After a team reaches a milestone, it reviews the lessons it learned. This allows teams to make midcourse corrections to avoid repeating mistakes and to highlight what went well so that best practices can be created. It is very important that these best practices be shared within and across teams throughout the organization.

Team Members Working Together at One Site

One of the goals of the team model is to lower communication overhead so teams have fewer obstacles to effective communication. Besides team structure, the location of the team plays a major role in how effective a team can be with its internal and external communication.

In their study of Microsoft, *Microsoft Secrets*, Michael A. Cusumano and Richard W. Selby state:

"… Single-site development allows project personnel to get together physically on a regular basis and explore ideas interactively. Frequent and easy communication can prevent major problems from getting worse."

Having teams work together at a single site also helps to enforce the sense of team identity and unity.

Scaling the Team Model for Large Projects

Overview

In his book *Rapid Development*, former Microsoft software developer Steve McConnell states:

"Large projects call for organizational practices that formalize and streamline communication. … All the ways to streamline communication rely on creating some kind of hierarchy, that is, creating small groups, which function as teams, and then appointing representatives from those groups to interact with each other and with management."

The MSF team model advocates breaking down large teams (those greater than 10 people) into small, multidisciplinary feature teams. These small teams work in parallel, with frequent opportunities to synchronize their efforts.

In addition, function teams may be used where multiple resources are required to meet the needs of a particular role and are grouped accordingly within that role.

Feature Teams

As defined earlier, each role in the team model is comprised of one or more resources organized in a hierarchical structure (although as flat as possible). For example, testers report to a test manager or lead. On top of this structure are feature teams. These are smaller sub-teams that organize one or more members from each role into a matrix organization. These teams are then assigned a particular feature set and are responsible for all aspects of that feature set, including its design and schedule. For example, a feature team might be dedicated to printing.

Steve McConnell writes in *Rapid Development:*

"Feature teams have the advantages of empowerment, accountability, and balance. The team can sensibly be empowered because it contains representatives … from each of the concerned parties. The team will consider all necessary viewpoints in its decisions and thus there will hardly ever be a basis for overriding it decisions.

"For the same reason, the team becomes accountable. They have access to all the people they need to make good decisions. If they don't make good decisions, they have no one to blame but themselves. The team is balanced. You wouldn't want development, marketing, or quality assurance alone to have ultimate say over a product's specification, but you can get balanced decisions from a group that includes representatives from each of those categories."

Function Teams

Function teams are teams that exist within a role. They are the result of a team or project being so large that it requires the people within a role to be grouped into teams based upon their functionality. For example, it is common at Microsoft for a product development team to have a product planner and a product marketer. Both jobs are an aspect of product management: One focuses on getting the features the customer really wants and the other focuses on communicating the benefits of the product to potential users.

This could also be true for development, where developers may be grouped by the service layer they work on: user, business, or data. It is also common for developers to be grouped based on whether they are solution builders or component builders. Component builders are usually low-level C developers who create reusable components that can be leveraged by the enterprise. Solution builders build enterprise applications by "gluing" these components together. Solution builders usually work with higher level languages like Visual Basic®.

Scaling the Team Model for Small Projects

Overview

Even though the team model consists of six roles, a team doesn't need a minimum of six people. It also doesn't require one person per role. The key point is that six goals have to be represented on every team. Typically, having at least one person per role helps to ensure that someone looks after the interests of each role. But not all projects have the benefit of filling each role in that fashion. Often, team members must share roles.

Sharing Roles

On smaller teams, roles have to be shared across team members. Two principles guide role sharing. The first is that development team members never share a role. Developers are the builders and they should not be distracted from their main task. To give additional roles to the development team only makes it more likely that schedules will slip due to these other responsibilities.

The second guiding principle is to try not to combine roles that have intrinsic conflicts of interest. For example, product management and program management have conflicting interests and should not be combined. Product management wants to satisfy the customer whereas program management wants to deliver on time and on budget. If these roles were combined and the customer were to request a change, the risk is that either the change will not get the consideration it deserves to maintain customer satisfaction or that it will be accepted without understanding the impact to the project. Having different team members represent these roles helps to ensure that each perspective receives equal weight. This is also true if trying to combine testing and development.

The following table illustrates risky (as indicated by "No" or "Unlikely" symbols) and synergistic (as indicated by "Possible" symbols) combinations of roles, but as with any teaming exercise, successful role sharing comes down to the actual members themselves and what experience and skills they bring with them.

	Product Mgmt	Program Mgmt	Development	Testing	User Ed	Logistics Mgmt
Product Mgmt		N	N	P	P	U
Program Mgmt	N		N	U	U	P
Development	N	N		N	N	N
Testing	P	U	N		P	P
User Ed	P	U	N	P		U
Logistics Mgmt	U	P	N	P	U	

Legend: **P** = Possible **U** = Unlikely **N** = No

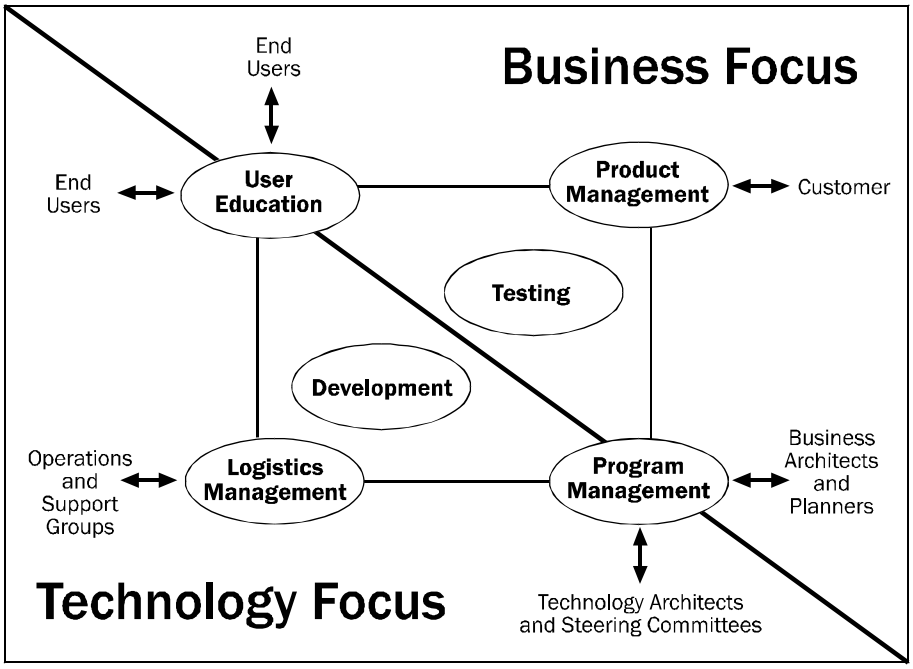
Combining Roles for Small Projects

The row column intersections with the N indicate that these roles should not be combined because of conflicting interests.
The row column intersections with the U indicate that it is unlikely that these roles would be combined, as the skills required for each role are quite different. For example, the skills and focus of product management vary greatly from those of logistics management.
The row column intersections with the P indicate that it is possible to combine these roles because they have compatible interests. For example, testing and user education both focus on users and try to ensure that the users' needs are met.
Of course, these are not absolutes. Some projects successfully share roles even though the chart indicates it's risky. The point is that if you are going to share roles, then you must keep in mind what the goals and focus of the roles are and try to reduce the amount of internal conflict that exists within a single individual. Otherwise, some aspect of the key quality goals might be overlooked, or risks might be mismanaged in some way.

Coordination with External Teams

Overview

In order for a team to be successful, it must interact, communicate, and coordinate with other external groups. These range from customers and end users to other product development teams.
The following diagram illustrates how coordination occurs either with a business focus or a technology focus. Program management, product management, user education, and logistics management are the primary facilitators. These roles are both internally and externally focused, whereas development and testing are internally focused and insulated from external communications.



Coordination with External Teams

The diagram represents a high-level perspective, as teams will typically have to coordinate with many more external groups, such as quality assurance, finance, and legal. It is important that the interfaces with any external groups be explicit and understood and that development and testing continue to be insulated so that they can work efficiently without unnecessary disruptions.

Conclusion

Overview

The MSF team model for application development is not a guarantee for project success. More factors than just team structure determine the success or failure of a project, but team structure is important. In *Rapid Development*, Steve McConnell illustrates this point by saying:

"Even when you have skilled, motivated, hard-working people, the wrong team structure can undercut their efforts instead of catapulting them to success. A poor team structure can increase development time, reduce quality, damage morale, increase turnover, and ultimately lead to project cancellation."

The MSF team model for application development is meant to address just that point. Proper team structure is fundamental to success, and implementing this model and using its underlying principles will help make teams more effective and therefore successful.