

## **IT Development**

# **Systems Development Methodology Handbook**

[Introduction](#)

[Iterative and Incremental Development](#)

[Glossary of Terms](#)

## **Phases**

### **Envisioning**

[Envisioning: Overview](#)

[Envisioning: Tasks](#)

[Envisioning: Artifacts](#)

### **Planning**

[Planning: Overview](#)

[Planning: Tasks](#)

[Planning: Artifacts](#)

### **Developing**

[Developing: Overview](#)

[Developing: Tasks](#)

[Developing: Artifacts](#)

### **Stabilizing**

[Stabilizing: Overview](#)

[Stabilizing: Tasks](#)

[Stabilizing: Artifacts](#)

### **Deploying**

[Deploying: Overview](#)

[Deploying: Tasks](#)

[Deploying: Artifacts](#)

[Appendix](#)

## IT Development

### Introduction

The IT Development and Engineering Systems Development Methodology (SDM) Handbook is a collection of tasks, artifacts, best practices, resources, and recommendations used internally to create information systems for Liberty University.

### How the methodology was created

Defining a systems development methodology for any development team is a tricky venture at best. Systems development is a rapidly changing and complex field. Compounding that is a sort of "methodology war" going on within the field between various industry and academic groups, each claiming that they know the best way to develop software and systems. "Just follow these three easy steps..." or "Employ these tools..." or "Organize yourself in this fashion...". How does one wade through all the books, articles, and slogans to employ an effective methodology? To compound this, developers tend to have very strong ideas about how they like to develop software and usually do not take well to anyone coming from on high giving them directives about a methodology.

Our programming department has also undergone much change in the past few years. From the late 1990's, when the department consisted of 2 people, to 2004, when it consisted of 12 people, and now as it consists of about 30 people, the size of the department has radically changed. From developing utilities and add-ons to be used with a third-party student information system to building full-fledged applications on that platform, as well as applications on the web and the University website, the nature of the software we are developing has changed (and is changing still with the advent of Banner). Also, we are developing more than just software now, with systems development, network engineering, and image development groups. From single-developer teams who work directly with one customer to multi-person teams, including developers, project managers, customers, operations, and support personnel, the way in which we are developing systems is changing as well. It became apparent that defining a standard methodology for the department was critical to managing the complexity and size of our undertaking.

In the Spring of 2004, Programming Services (our name at the time) at Liberty University formed an internal task force that was charged with creating a software development methodology for the department. That task force was chaired by Jonathan Minter (Director of Programming Services) and its members included Lori Baker (Project Manager), Darrell Hyatt (Developer), and Anderson Silva (Developer).

The task force decided that the best approach for developing the methodology would be to do it in an iterative, incremental fashion. Therefore, instead of retreating to a hole for 6 months and emerging with a large, complex document that is to be followed immediately, we would develop the methodology in small iterations. This has all the benefits of iterative and incremental software development, including rapid feedback, easier management, and delivery of early value.

## The basis for the methodology

Over the last several years, IT Development and Engineering had begun researching and utilizing components of the Microsoft Solutions Framework (MSF). This framework includes many models and bodies of knowledge that have been gleaned from Microsoft's internal development and services groups. Specifically, the we had begun to utilize the MSF Process Model, which includes a specific description of the Software Development Life Cycle (SDLC) and various principles for good systems development. Also, we began using the MSF Team Model, which includes a team structure and roles to properly construct a project team.

Therefore, this methodology utilizes the MSF as the basis for the methodology, including the Process and Team Models. Much of the content in the methodology is adapted directly from the white papers found at <http://www.microsoft.com/msf>

The MSF endorses iterative and incremental development. We are also incorporating agile development principles into our methodology. Therefore, we are valuing *individuals and interactions over processes and tools, working software over comprehensive documentation, customer collaboration over contract negotiation, and responding to change over following a plan* (The Agile Manifesto, <http://www.agilemanifesto.org>).

## The goals of the methodology

At the onset of the task force, we identified certain outcomes that the task force should achieve for IT Development and Engineering

- The methodology should define areas of expertise and knowledge in which the entire department should be proficient.
- The methodology should be able to be succinctly communicated inside and outside of IT Development and Engineering.
- The methodology should enable communication among IT Development and Engineering employees, customers, and other IS employees.
- The methodology should be sufficiently flexible to handle all projects, large and small, whether they are in house software development, implementation of a COTS product, systems projects, network projects, or any other IT project.
- The methodology should be sufficiently flexible to allow each project to tailor the methodology to suit the specific needs of the project.
- The methodology should support iterative and incremental development.

## The structure of the methodology

In defining the methodology, we chose to create one page per phase in the cycle. Each page has a similar structure.

### *Description of the Phase*

### *Tasks and Artifacts Table*

A table defining the various tasks that should be undertaken by the project team during this phase as well as the artifacts that should be present after the phase is complete. This table should serve as a quick reference guide for project managers each time they enter and exit a phase.

### *Questions to Answer*

A list of questions that the project team should be able to answer if they are done with the phase. This can also serve as a cheat sheet for the project manager.

### *Milestones*

The name and description of the milestone that will be hit when the phase is complete.

### *List of tasks and artifacts*

Each task or artifacts will include a description as well as resources that would help a team member better perform that task or create the artifact.

## **How to Change the Methodology**

We view this methodology as a living document, always open to change. We view the methodology as a tool to help us create better software, not something to be followed blindly. Therefore, we should be constantly evaluating our methodology to ensure that we are serving our purpose (creating software). Any areas of the methodology that are not bring value to that goal should be stripped out or refined.

Suggestions for changes in the methodology should be given to the Director or any person on the Methodology Task Force.

[Next: Iterative and Incremental Development](#)

[Back: Table of Contents](#)

## IT Development

### Iterative and Incremental Development

#### What is Iterative and Incremental Development?

The following discussion is adapted from "Agile and Iterative Development: A Managers Guide" by Craig Larman.

##### *Iterative Development*

An approach to building systems (or anything for that matter) in which the overall lifecycle is composed of several iterations in sequence. Each iteration is a self-contained mini-project composed of activities such as requirements analysis, design, programming, and test.

##### *Incremental Development*

An approach to building systems in which the final system is built by adding and releasing new features, iteration by iteration. In this approach, the team does not wait until the entire system is functionally complete to deploy the product to users. It is deployed little by little until the feature set is adequate or funding for new enhancements is removed.

#### How does that fit with our methodology?

This systems development methodology uses five primary phases (Envisioning through Deploying). It is quite possible that a development team could take those phases and take a waterfall approach to development (that is, a strict sequencing of phases that is focused on fully completing each phase before moving on to the next). In this scenario, if there are 12 months of features to be developed, the development team takes 2 months for Envisioning, 3 months for Planning, 6 months for Developing, and so on.

That approach is not espoused by this methodology. Instead, this methodology encourages splitting up the features into many iterations and deploying several times during that 12 month period. There are two main ways to structure the project using the MSF Process Model and iterative development:

##### **One Envisioning phase and iterate over Planning through Deploying**

With this approach, the project team will spend a large amount of time Envisioning the entire project (i.e. all 12 months of development). The team will structure the project, come up with a rough schedule, and plan out which features go in each iteration. Then, they will embark on many iterations, basically treating an iteration as Planning through Deploying. This has the benefits of iterative development, without all the overhead and the "fuzzy front end" delays of Envisioning.

##### **Iterate over Envisioning through Deploying each Iteration**

In this approach, the team treats each iteration as its own project. They cycle through the all the phases as if they were starting the project over again. In practice, the subsequent Envisioning phases will be shorter than the first one, but the activities will still be performed. The advantage of this approach is that the

team makes sure they are not assuming too much or getting too far off track from their original goals and structure of the project.

## How long should an iteration be?

Again from "Agile and Iterative Development: A Manager's Guide".

In modern iterative methods, the recommended length of one iteration is between one and six weeks. Each iteration includes production-quality development, not just requirements analysis, for example. And the system resulting from each iteration is not a prototype or proof of concept, but a subset of the final system.

This methodology espouses an iteration length of **3-6 weeks**. This has several benefits:

- *It provides early value to the users.* They are able to start using the system and getting the benefit of the system as soon as the features are developed.
- *It provides early feedback to the development team.* If the features developed somehow do not meet the needs of the users, the developers become aware of that shortly after developing the features. Those problems can be corrected before moving on too far into the rest of the project (possibly with bad assumptions). It ensures that the final system meets the users' true needs.
- *It gives the users confidence in the development team.* When the users see early, concrete features they can actually start using, it will inspire confidence in the team. Also, if they provide feedback to the team and the product is changed because of it (in the next iteration), they will feel that their voice has been heard and they will take greater ownership in the product.
- *It provides for easier implementations.* It is much easier to deploy a small set of features (including deploying the code, converting data, training, writing users manuals) than it is to do the entire system at once. If too many features are delivered at once, critical activities (like full testing, users manuals, training) will most likely be skipped because they are too labor-intensive.
- *It provides the University with flexibility for changing priorities.* If priorities change and resources must be diverted from a project, all the work done in the project will not be lost. The development team has at least provided some value to the users. Otherwise, 6 months could have been spent in development, with nothing to show for it.

## Are we agile?

Again, adapted from "Agile and Iterative Development" by Larman.

The hype around supposedly "agile" methodologies leave almost everyone claiming that they are "agile". What does this mean?

### *Agile Development Defined*

If agile methods have a motto, it is *embrace change*. If agile methods have a strategic point, it is *maneuverability*. It is not possible to exactly define agile methods, as specific practices vary. However, short timeboxed iterations with adaptive, evolutionary refinement of plans and goals is a basic practice various methods share. In addition, they promote practices and principles that reflect an agile sensibility of simplicity, lightness, communication, self-directed teams, programming over documenting, and more.

In short, this methodology espouses those principles, and the principles of the Agile Manifesto without adhering to a specific methodology that classifies itself as agile (XP,

SCRUM, DSDM, etc.)

### *The Agile Manifesto*

No, we aren't trying to overthrow any government or create a Communistic society. In 2001, a group of methodologists met to bring together the various "new" methodologies that had been developed over the previous decade or so. They boiled down the principles of those methodologies into a simple statement, now known as the Agile Manifesto:

"We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

Individuals and interactions over processes and tools  
Working software over comprehensive documentation  
Customer collaboration over contract negotiation  
Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more." ([Agile Manifesto](#))

[Next: Glossary of Terms](#)

[Back: Introduction](#)

## **Glossary of Terms**

This section serves a glossary of many terms that will be used throughout this handbook. It serves several purposes:

- Assist in understanding of various terms in the discipline
- Resolve discrepancies or ambiguity in terms used in different areas of the discipline
- Assist in creating a common language for the department
- Assist our customers in understanding the terms we use in our projects

**Artifact:** Any document (physical or electronic) that persists after the project is completed. Artifacts include items delivered to the customer (project plans, requirements, etc.) as well as items used internally (design documentation, etc.)

**Deliverable:** Items created in the development process that are built for the customer. Deliverables are defined by the team and are different for each team. Examples include source code (and associated features), users manuals, etc.

**Measurements:** (also known as metrics) The collection of numeric assessments of a project's success. Project success should be (if at all possible) stated in terms of objective, measurable goals that leave no room for interpretation. For instance, a project measurement could be the number of defects found after release of the product.

**Methodology:** A set of standards or protocols to follow when developing software. Typically, the methodology will include an SDLC and associated practices for each phase. Methodologies are also defined in terms of their "weight". That is, heavyweight methodologies require much documentation and ceremony. Whereas, lightweight methodologies require little documentation and ceremony outside of writing the code.

**Microsoft Solutions Framework (MSF):** A software development framework developed by Microsoft for their consumer product development, as well as internal and external business application development (see <http://www.microsoft.com/msf>). The MSF describes ways of organizing teams, structuring projects, and other effective practices for software development.

**MSF Process Model:** A subset of the Microsoft Solutions Framework that describes the life cycle of an MSF project. It describes the five phases of a project (Envisioning, Planning, Developing, Stabilizing, Deploying), as well as principles to be used in development.

**MSF Team Model:** A subset of the Microsoft Solutions Framework that describes best practices in organizing teams for software development. It describes 6 roles that should be filled and other principles and best practices of building teams.

**Risk Mitigation:** The process of risk assessment in software projects is the act of finding all possible things that could go wrong with a project during its development and deployment. The process of keeping these risks from occurring, or minimizing the impact of a risk that does occur is risk mitigation.



**Scope:** What features are in the project (as opposed to those that are out of the project). The scope defines the boundaries of the work that will be done on the project.

**Software Development Life Cycle (SDLC):** The phases a software development project will go through from inception of the project to completion.

**Stakeholders:** Members of the University community (or sometimes outside the community) that have a stake in the outcome of the project. Frequently stakeholders are senior staff members who are responsible for large parts of the university. Additionally, stakeholders might be those in departments who are affected by the work of the department the project is serving.

**Task:** In this methodology handbook, we are using the term task to denote anything the project team must do throughout the cycle of development that doesn't necessarily yield an artifact or deliverable. Sometimes these tasks become an input to an artifact (e.g. developing a Rough Draft of Schedule becomes an input to the Project Plan). Many times, it is something that just creates value for the team that doesn't need documented (e.g. Peer Reviews or User Department Observation)

**User Department:** The departments represented by the users of the system. The user department is classically labeled the "business side" (as opposed to the IT side)

**User Interface:** The part of the software system that the user interacts with. This could be a web page, an RPG program, or a windows application. The distinction is made between this user interface and the application logic that goes on "behind the scenes"

[Next: Envisioning - Overview](#)

[Back: Iterative and Incremental Development](#)

## IT Development

### Envisioning: Overview

The purpose of the Envisioning phase of a project is to allow the project team to determine the initial goals, scope and structure of the project.

#### Tasks and Artifacts

##### Tasks

Team Formation

Risk Assessment

Security Analysis

Preliminary Schedule Development

- Rough Draft of Scope
- Rough Draft of Schedule

User Department Observation

Measurements Development

##### Req'd Role Responsible

Yes Project Management

Yes Project Management

Yes Development

Yes Project Management

Product Management

Project Management

##### Artifacts

Project Plan

##### Req'd

Yes

##### Role Responsible

Project Management

#### Questions to Answer

- Have you identified team members for each role of the team?
- Does the team include appropriate customer representation, with the needed authority to make decisions about features, priorities, and resources?
- Does the team have adequate Information Services representation that can ensure a smooth implementation, training, and ongoing maintenance of the product?
- Have you identified the major features that will be included on this project?
- Have you identified the major risks associated with this project, along with mitigation strategies for those risks?
- Have you developed a high-level schedule for all phases of the project?
- Do the developers on the project have an adequate understanding of the business domain?
- Does the team have an understanding of the sensitivity of the data that will be handled through the system?
- Does the team know what measurements you are going to track throughout the project?

#### Milestone

Project Scope and Vision Approved

The Envisioning phase is complete when the entire team and stakeholders have come to an agreement regarding the scope and structure of the project.



## IT Development

### Envisioning: Tasks

#### Team Formation

##### Description

The team formation process is modeled after the Microsoft Solutions Framework (MSF) Team Model. This model identifies 6 roles for each project.

The goal of this task is to identify at least one person to be the lead on each role. Individuals may fulfill duties on more than one role. Also, there may be more than one person fulfilling one role.

For reference, a short description of each role is listed below.

##### *1. Product Management*

Satisfied Customers: This role is primarily responsible for giving requirements to the developers, prioritizing requirements within the project, and establishing the business case.

##### *2. Program (Project) Management*

Delivery Within Project Constraints: This role is primarily responsible for making sure the project meets the established goals of the team, making sure it is delivered on time and within budget, allocation resources, facilitating communication, and driving critical decisions.

##### *3. Development*

Delivery to Product Specifications: This role is primarily responsible for building the product such that it meets the needs of the customer, being a technological consultant, and creating and maintaining estimates.

##### *4. Testing*

Release After Addressing All Issues: This role is primarily responsible for making sure all issues are known and addressed before release.

##### *5. User Education*

Enhanced User Performance: This role is primarily responsible for making sure the users can be as productive as possible with the new (or revised) product, providing input on usability, and training the users on the product before and after release. For each project, a representative from DISC will fulfill this role.

##### *6. Logistics Management*

Smooth Deployment and Ongoing Management: This role is primarily responsible for ensuring that the deployment goes as smoothly as possible and that the product is set up for long-term maintenance.

#### Resources

[MSF Team Model Detailed Description](#) 

[Peopleware - DeMarco and Lister](#) 

[Getting Your Team Off on the Right Foot](#) 

### Risk Assessment

## Description

Risk Assessment is the process of perceiving, analyzing and preparing for (or dealing with) conditions or events which threaten the success of a project. The amount of effort needed to assess risk is usually proportional to the size and scope of the project. Since life changes as a project moves along, risk assessment must be done in all phases of the project.

During the envisioning phase of a project, most of the risk assessment effort will be directed towards predicting which conditions or events which are the most likely to threaten the success of the project. As such, your goals in risk assessment are:

- list as many potential dangers that can be reasonably foreseen
- determine the probability of occurrence for each danger listed
- determine the extent of potential loss for each danger listed

Once this is done, re-order the entire list from most dangerous to least dangerous. This will help keep things in perspective as the project wears on. (Remember to re-order the list again when re-assessing the dangers in later phases.)

## 5 Common Risks to Assess for the Project

- Are the stakeholders actively involved in their role in the project?
- Are new features requested after the requirements have been declared complete?
- Do any features or products utilized violate the provided Information Security Guidelines?
- Do the requirements affect a change in other products not previously included in this project?
- Does inadequate testing contribute to a larger number of defects in the deployed product?

## Resources

[Risk Management - Wikipedia](#) 

[Conducting a Project Risk Assessment](#) 

[Risk Assessment - 17 Steps to Success](#) 

[Risk Management - Tasmanian State Government Guide](#) 

[Project Management Framework - Risk Management](#) 

[Reducing Project Management Risk](#) 

[Project Health - Should We Keep Investing?](#) 

[When is a Risk Not a Risk?](#) 

[The NASA Guide to Effective Risk Management](#) 

[Take a Risk \(GanttHead.com article\)](#) 

## Security Analysis

### Description

The Security Analysis for the project should be incorporated into the Risk Assessment document. If security is not taken into consideration from the beginning of a project, it will become more and more expensive to maintain an application's security. We want to distance ourselves from the "penetrate and patch" process to a new "cradle-to-grave" mindset where security is taken into consideration in all phases of the SDM.

The developer should consider the nature of the data/application in deciding an appropriate level of security.

Most security measures are based on the following concepts (also known as the CIA model of security):

- Confidentiality
- Integrity
- Availability

## Security Levels

RED - where data would only be stored temporarily and after a task is completed, we delete part of the data. (i.e. credit card information).

ORANGE - where data is sensitive, and belongs to a person, but some other authorized people may have access as well. The data has importance outside the University as well. (i.e. SSN, Address)

YELLOW - where data is sensitive, but it is sensitive only to the university. (i.e. Grades, Financial Aid, etc). It is still only accessible by the owner or an authorized person.

GREEN - Data that identifies a person that could be displayed to the public. (i.e. Name)

BLUE - Stats info. (Public Data)

Some questions to consider during this phase are:

- How sensitive is this project?
- Have you picked the Security Level your project falls under?
- What metrics will be used throughout the project to monitor security development?
- Will it have public access? restricted access?
- Are there legal issues involved with this application? (e.g. privacy)
- Do you have enough resources in the team (including stakeholders) to analyze security risks?

## Rough Draft of Scope

### Description

The purpose of preliminary schedule development in the Envisioning phase is to define the broad milestones of the project.

In the MSF, a milestone is a synchronization point for the project. It represents a major event in the life of a project, such as the first working release, requirements freeze, project closure, etc. Milestones can also be defined using the transition from phase to phase in the project.

A list of milestones for the project, and corresponding target dates for those milestones should be present in the Vision and Structure document (see below)

### Resources

[MSF Process Model \(description of milestones\)](#) 

## Rough Draft of Schedule

### Description

The purpose of developing a rough draft of the scope is to identify the major features and goals that the new or modified product is to provide and to accomplish. When a project begins, each member of the team may have a different set of assumptions for what the project is to accomplish. This allows the team to identify the major differences in these assumptions.

At this level, we are most concerned about identifying the deliverables so that a project schedule can be drafted. Deliverables refer to the products that will be delivered to the customer, such as the features that will be developed, training manuals, etc.

### Resources

[Defining the Scope of a Project](#) 

## User Department Observation

### Description

The purpose of user department observation is to learn as much as possible about the business processes of the user. User observation assists in the development of the problem definition, business needs, business processes, and user interface design.

### *Methods of User Department Observation*

1. Observation - Take anywhere from a couple of hours to several days to observe users going through the business process that is being programmed for the project.
2. Training - Participate in any training classes or new employee training that the department uses. User manuals may also be a valuable tool for this task.

### Resources

[The I.T Inside the World's Biggest Company](#)  

## Measurement Development

### Description

A metric is simply a measure of some property or component of a project which is collected for the purpose of diagnosing a problem or evaluating the success of a product. Many project teams initially will go overboard and attempt to measure everything, hampering the pace of the project. Other teams, having been burned by over-metrification, abandon metrics altogether and lose out on the benefits that a precisely conceived metrics set could provide. The point is to measure only that which will yield quality information.

In the envisioning phase, we are only concerned with deciding which metrics are important and planning to collect data for those metrics. The exact method used for collecting each metric does not need to be decided at this time. One of the easiest ways to generate a set of metrics is to take each goal stated in the Vision and Structure Document and attempt to re-state it in a quantifiable manner. For some goals, this may mean conducting research into what the baseline measurement is. For example, if the goal is "to reduce account balance processing time by 50%", then you must know what the average account balance processing time (and generally the range of one standard deviation in either direction) to determine if you have met the goal.

Most of all, start off with a set of metrics you can manage. As we get used to determining, collecting, and analyzing the kind of metrics which matter to us, we can let the data tell us where we are missing measures. We can add the new metric to the set in the next version of project. It is better that we not have one or two measures that we would have liked than that we collect too much useless data, get frustrated and give up on metric collection altogether.

### 5 Common Measurements

- Number of changes to requirements
- Number of defects caught before deployment (after development is complete)
- Number of defects caught after deployment
- Average amount of reduced time for business processes
- What are the average amounts of time that it takes to train a person to use this product(s)?

### Resources

[TenStep - Managing Metrics](#)  

[Wikipedia - Software Metrics](#)  

[Software Metrics Primer](#)  

[Process and Project Metrics](#)  

[Software Productivity Center, Inc. - Metrics](#)  

[Spin.org - Customer Success Metrics](#)  

[← prev : Envisioning - Overview](#)

[Envisioning - Artifacts : next →](#)



## IT Development

### Envisioning: Artifacts

#### Project Plan

##### Description

The Project Plan combines the three main documents that must be completed for the Envisioning Phase. These documents are: The Vision Document, the Project Structure Document, and the Master Risk Assessment Document. The following is a description of the components of The Project Plan.

The official version of this document should be signed by the project team and stakeholders. The signed copy of the document should be provided to the IS Administration Office for filing.

##### Vision

*Vision* - Documents the description of the project's deliverable as the solution to the stated need.

*Need* - Why is this project needed? What is the problem definition?

*Goals and Objectives* - What are the broad goals and objectives for the project?

*Feasibility* - Has this project been determined as feasible for the University to complete?

*Constraints/Assumptions* - What are the project assumptions and constraints?

*Scope Statement* - What are the boundaries for the project? What is in and what is out?

*Schedule/Timeframe* - What is the high level estimate for the timeframe of the project?

*Performance Objectives* -

1. Broad Design Considerations
2. Measurable Goals - What are the team's measures of success for the project?

##### Structure

*The Project Team*

1. Team Organization - Who is the project team and what are their roles? See Team Formation for a description of the team roles.
2. Staffing Management Plan - Documents the required human resources and their approximate contribution time for the project.

*Stakeholders* - Who are the stakeholders?

*Communication Plan* -

1. Email List - Documents the members of the project team and the email distribution list that was created for the project.
2. Meeting Schedule - Documents when the team will meet and who is required to attend the meetings.
3. Reporting Plan - Documents the various reports and items that must be

communicated to keep the team and stakeholders thoroughly informed of the status of the project.

*Change Management Plan (optional)* -

1. How will change requests be handled?
2. How will changes be tracked?

*Quality Assurance Plan (optional)* - The process to evaluate the project's performance to ensure that the project will perform within quality standards.

### **Master Risk Assessment**

See the Risk Assessment Plan

### **Security Analysis**

What is the security level for the project? How will security be monitored throughout the project?

(See the Security Analysis task for more information)

### **Resources**

[DLP Contact System Vision and Structure Document](#)

[Guide to the Project Management Body of Knowledge](#) 

[Complete Idiot's Guide to Project Management](#) 

[Low Intensity Project Management \(Gantthead.com\)](#) 

[← prev : Envisioning - Tasks](#)

[Planning - Overview : next →](#)

## IT Development

### Envisioning: Artifacts

#### Project Plan

##### Description

The Project Plan combines the three main documents that must be completed for the Envisioning Phase. These documents are: The Vision Document, the Project Structure Document, and the Master Risk Assessment Document. The following is a description of the components of The Project Plan.

The official version of this document should be signed by the project team and stakeholders. The signed copy of the document should be provided to the IS Administration Office for filing.

##### Vision

*Vision* - Documents the description of the project's deliverable as the solution to the stated need.

*Need* - Why is this project needed? What is the problem definition?

*Goals and Objectives* - What are the broad goals and objectives for the project?

*Feasibility* - Has this project been determined as feasible for the University to complete?

*Constraints/Assumptions* - What are the project assumptions and constraints?

*Scope Statement* - What are the boundaries for the project? What is in and what is out?

*Schedule/Timeframe* - What is the high level estimate for the timeframe of the project?

*Performance Objectives* -

1. Broad Design Considerations
2. Measurable Goals - What are the team's measures of success for the project?

##### Structure

*The Project Team*

1. Team Organization - Who is the project team and what are their roles? See Team Formation for a description of the team roles.
2. Staffing Management Plan - Documents the required human resources and their approximate contribution time for the project.

*Stakeholders* - Who are the stakeholders?

*Communication Plan* -

1. Email List - Documents the members of the project team and the email distribution list that was created for the project.
2. Meeting Schedule - Documents when the team will meet and who is required to attend the meetings.
3. Reporting Plan - Documents the various reports and items that must be

communicated to keep the team and stakeholders thoroughly informed of the status of the project.

*Change Management Plan (optional)* -

1. How will change requests be handled?
2. How will changes be tracked?

*Quality Assurance Plan (optional)* - The process to evaluate the project's performance to ensure that the project will perform within quality standards.

### **Master Risk Assessment**

See the Risk Assessment Plan

### **Security Analysis**

What is the security level for the project? How will security be monitored throughout the project?

(See the Security Analysis task for more information)

### **Resources**

[DLP Contact System Vision and Structure Document](#)

[Guide to the Project Management Body of Knowledge](#) 

[Complete Idiot's Guide to Project Management](#) 

[Low Intensity Project Management \(Gantthead.com\)](#) 

[← prev : Envisioning - Tasks](#)

[Planning - Overview : next →](#)

## IT Development

### Planning: Overview

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

### Tasks and Artifacts

Tasks	Req'd	Role Responsible
Requirements Gathering	Yes	Project Management
Schedule Development	Yes	Product Management
Risk Reassessment	Yes	Project Management
Peer Reviews	Yes	Development
Measurements Collection		Project Management Development

Artifacts	Req'd	Role Responsible
Use Cases	Yes	Product Management
Design Documents		
<ul style="list-style-type: none"> <li>• User-Interface</li> <li>• Business-Logic</li> <li>• Database</li> </ul>	Yes	Development
Updated Project Plan	Yes	Project Management
Security Checklist	Yes	Development

### Questions to Answer

- Did the requirements gathering provide enough information to design the system?
- Does the project team have a rough idea of how long this iteration will take?
- Do the risks identified in Envisioning still apply or have new risks surfaced?
- Is the project team doing what they said they would do to keep the risks identified in Envisioning from occurring?
- Have the designs been reviewed and evaluated by someone outside the project team?
- Is the project team collecting the data they said they would in Envisioning?
- Does the team have a shared understanding of the business problems the system is designed to solve?
- Can the peer reviewers understand the intended design of the system through the design documents?
- Does the project plan accurately reflect changes made to requirements, schedule, resources, and risks?
- Has the developer constructed a security checklist appropriate for the environment and requirements of the project?

### Milestone

Project Plans Approved

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

[← prev : Envisioning - Overview](#)

[Developing - Overview : next →](#)

[Planning - Tasks : next →](#)

### Planning: Tasks

#### Requirements Gathering

##### Description

A requirement is a necessary attribute in a system, a statement that identifies a capability, characteristic, or quality of a system. The difficult part of requirements gathering is not documenting requirements; it is the effort of helping the customer figure out exactly what they need. A requirement specifies what will be provided, not how a requirement will be provided. The question of how is part of the design. (Source: Recommended Requirements Gathering Practices -Dr. Ralph R. Young)

There are three levels of requirements:

1. Business requirements: high-level needs that, when addressed will increase the value of the business.
2. User requirements: Bridge the requirements of the business and the requirements of the software.
3. Software Requirements: The doing parts are the functions that derive directly from your user requirements. The being parts are non-functional needs that must be addressed.

There are numerous requirements gathering techniques that can be adapted for the project. A few are listed below:

- Interview/Discussion: The most common way to gather requirements is through interviews and discussions. It is an effective way to formally or informally research what the necessary requirements are for the project.
- Requirements Workshops: A structured way to capture requirements from which commitment, teamwork and consensus is often found. (See Requirements by Collaboration: Workshops for Defining Needs - Ellen Gottesdiener for more information on workshops)
- Prototyping: See Design Documents: User interface for more information on prototyping
- Use Cases: See the Use Cases section below for more information on Use Cases
- Storyboards: A storyboard is a set of drawings depicting a set of user activities that occur in an existing or envisioned system or capability.

##### Resources

[Requirements by Collaboration](#) 

[Recommended Requirements Gathering Practices](#) 

#### Schedule Development

##### Steps to Building a Project Schedule:

1. Develop a task list

- *Developing a task list involves dividing the project into smaller, more manageable components and then specifying the order of completion. It answers what tasks need to be done to accomplish the projects objectives. The task list should not only include task development time, but other project management tasks such as training, testing, and implementation.*

2. Estimate Task Duration

- *5 options for estimating task duration:*

1. Ask the people who will be doing the work
1. Get an objective expert's opinion
2. Find a similar task in a completed project plan
3. Perform a test session of the task if time permits
2. Make your best educated guess

### 3. Document the Project Schedule

- There are numerous ways to document the project schedule. Some examples are MS Word, MS Excel, MS Project, Sticky Notes, etc.

#### Resources

[Schedule Checklist - GanttHead.com](#) 

## Risk Reassessment

Throughout the life of the project continual risk reassessment must be practiced in order to make certain that the success of the project is not threatened. During Risk Reassessment the results of the Risk Assessment task during the Envisioning Phase should be revisited. There are a few areas that must be addressed for Risk Reassessment:

- Do the risks identified during the Envisioning Phase still apply?
- Have any additional risks been discovered during the Planning Phase?
- Is the probability of occurrence for each risk still accurate?
- Are appropriate measures being taken to make sure that the risks do not occur?
- Are any risks that have occurred being properly handled as documented in the Project Plan?

## Peer Reviews

### Description

Peer Reviews provide a resource for teams to identify potential design and implementation flaws, and increase the probability of success. Applying this process early in the life cycle allows for maximum advantage in terms of resource efficiency as well as design confirmation and ultimate mission success.

Issues that should be addressed by the Peer Reviews in Planning:

- Requirements
- Security
- Design

Peer reviews will be performed throughout the lifecycle of the project. In Planning, the developer should identify someone in the department (outside the project team) that will be the reviewer for this particular project. This will enable someone to see the designs as they evolve and also review the code as it is written.

The developer and reviewer should work out the details of how they want to perform their reviews on a project by project basis.

#### Resources

[An Abridged Guide to Reviewing Code](#) 

[Part of Your Complete Breakfast: Code Reviews](#) 

## Measurements Collection

For Measurements Collection in Planning, you should refer to the measurements defined in Envisioning. Basically, whatever measurements you said you would track in Envisioning, you need to collect (where appropriate) in Planning.





### Planning: Artifacts

#### Use and Misuse Cases

##### Description

Use cases are a method of gathering and analyzing requirements. In essence, they are stories of using the system. Most often use cases are textual descriptions of some actor (some user) performing some action in the system. The use case then describes the steps by which the user and the system will interact. Also, the use case can document what will happen in certain exception conditions.

Misuse cases on the other hand are concerned with security requirements. Use cases typically concentrate on what the system should do, while misuse cases describe what the system should not do. Stakeholders do not normally think about what the system should not do, rather they are more concerned with what the system should look like and what it will do. A misuse case is a sequence of actions that will result in a loss to the organization. In order for us to be able to properly assess the security risks within our projects we must take time to also look into misuse cases when we are completing the Planning Artifact of Use Cases.

See the link below for example use cases. The body of knowledge for use cases is too broad for a full treatment in this handbook. See the other resources below and the books listed below for more information.

This handbook espouses the following principles in writing use cases:

1. *Use cases should not include any information about the user interface.* UI information only bogs down the use case and makes it more unmanageable. UI requirements should be documented in the User Interface Design document (below).
2. *Use cases should focus on the user's actions, not the system's actions.* Although this information is sometimes necessary to include, the use cases should not get bogged down in low-level system details. Use cases are for customers to express how they think the system should behave when they interact with it.
3. *We will err on the side of informality rather than formality and simplicity rather than complexity.* A simple list of steps for the use case and any alternate flows will be sufficient. The "fully dressed" format espoused by some is not necessary to provide value and only muddies the waters when getting user feedback on the use cases.
4. *Use case diagrams are not necessary.* These diagrams create little value and are just one more place to record changes in requirements.

##### Resources

[Use Cases: Robert C. Martin](#)  

[Use Cases, Ten Years Later: Alistair Cockburn](#) 

[Use and Abuse Cases: Martin Fowler](#)  

[Writing Effective Use Cases: Alistair Cockburn](#)

[How to Avoid Use Case Pitfalls](#) 

[Use Cases of Mass Destruction](#) 

[Use Case Examples](#)

[Templates for Misuse Case Description](#)  

## Design Documents: User Interface

### Description

Non-Function User Interface Prototyping is a method of usability testing that is useful for Web sites, Web applications, and conventional software. In a nutshell, you make screen shots, diagrams, hand written drafts of web screens, menus, image placement, forms, pop-up windows, etc of the project you are trying to build.

Non-Function User Interface should help the Project Manager and the Programmer answer some of the following questions:

*Concepts and terminology:* Do the target users understand the terms you've chosen? Are there key concepts they gloss over or misconstrue?

*Navigation/workflow:* If there's a process or sequence of steps, does it match what users expect? Do they have to keep flipping back and forth between screens? Does the interface ask for inputs that users don't have, or don't want to enter?

*Content:* Does the interface provide the right information for users to make decisions? Does it have extra information that they don't need, or that annoys them?

*Page layout:* Although your scribbled screens may not be pretty, you'll still get a sense of whether users can find the information they need. Do you have the fields in the order that users expect? Is the amount of information overwhelming, not enough, or about right?


*Functionality:* You may discover missing functionality that users need, or functionality you'd planned but users don't care about.

*Source: Paper Prototyping - Morgan Kaufmann*

### Resources

[Paper Prototyping - Carolyn Snyder](#) 

[IBM - Paper Prototyping](#) 

[Paper Prototyping: Getting User Data Before You Code](#) 

## Design Documents: Business-Logic

### Description

The Business Logic layer is the layer of code that sits between the Database layer (where information is actually stored) and the User-Interface layer (where information is presented to the user of the software in a way that he or she understands). The purpose of the Business Logic layer is, in a sense, purely transformative. That is, its sole reason for existing is to take the information stored in the database and to alter it into a format which the User-Interface layer can present to the product user.

The entire trouble with the Business Logic layer is that it is not usually implemented entirely separate of either of the other two layers. More often, it is either included in the user-interface (e.g. a Cold Fusion script or an RPG program which displayed "Freshmen" when given the value 1 from the database, "Sophomore" when 2, "Junior" when 3, etc.) or it is included in the database (e.g. through the use of table joins, views, or stored procedures). This means that the temptation arises for the analyst to say "oh, well, look THAT is just part of the user-interface/database". However, for documentation purposes, this temptation must be resisted because later programmers (or, indeed, you yourself) may need to alter the layer in which the business logic was placed (e.g. a stored procedure must be created to do the transformations because the script/program doing them now processes too slowly).

One simple way to track how the data is transformed is to create a spreadsheet that

details what database values resolve to what user-interface values (and, of course, vice versa). In the case where shortcuts are used (e.g. joining in a table of country names and abbreviations), documenting the way that table should be joined in is sufficient.

Another method is simply to describe in a text document what transformations will be done.

For more complex projects, the above suggestions may not work. If this is the case, the use of Data Flow Diagrams, flow charts, UML Sequence Diagrams or even some other method may be necessary. For these techniques, using drafting/drawing or some sort of presentation software (e.g. VISIO or PowerPoint) is preferable and generally easier.

### Resources

[The Data Flow Diagram Quick Reference](#) 

[How to Draw Data Flow Diagrams](#) 

[A Fresh Look at Flow Charting](#) 

[Flow Chart and Decision Process](#) 

[UML 2 Sequence Diagram Overview](#) 

[Modeling with UML](#) 

## Design Documents: Database

### Description

One of the most important features of database documentation is how the various tables relate to one another.

The easiest way to document these relationships, if you are creating the database using SQL Server, is to create all of your tables and relationships using the Diagrams tool in Enterprise Manager. This tool will document your database in a visual manner intuitive enough for other developers.

If you are creating the database on the AS/400, you have less luck since there is no native tool to document the relationships involved. However, since you are stuck doing your own documentation, it is recommended that you use a visual tool such as VISIO or PowerPoint to document the tables and their relationships.

### Resources

[Data Modeling](#) 

[Database Normalization for the Average Jane/Joe](#) 

[Wikipedia - Database Normalization](#) 

[Database Normalization and Design Techniques](#) 

[UT Austin - Database Management Principles](#) 

## Updated Project Plan

The Project Plan is a living document and must be updated as changes are made to the various areas of the project. The use of iterative and incremental development justifies updating the project plan. Iterative and incremental development allows for some changing of requirements and goals as the project goes through different iterations and phases. There are numerous areas in the Project Plan that could change throughout the life of the project. For example, the scope could be modified, a team member may be replaced, or the projects deliverables may change.

The preferred method for updating the Project Plan is to create a revision of the existing document. Previous versions of the Project Plan should be saved so that they can be referred to in the future if historical information is needed.

# Security Checklist

## Description

During the planning process, security planning is focused on identifying security risks for the application being developed.

The developer should construct a checklist that identifies all possible security risks that should be accounted for in the development process. This checklist will serve as a basic security requirements document. The items on the checklist will come from known vulnerabilities for the platforms on which the application is being developed, as well as special considerations based on the type of data manipulated by the application.

### *Vulnerabilities*

For web applications, see the OWASP Top Ten Web Application Vulnerabilities (linked below) as a start.

For AS/400 applications, see the AS/400 Top Ten (or so) Vulnerabilities (linked below) that was developed in-house as a start.

For systems implementations, see the SANS Top 20 vulnerabilities (linked below) as a start for the specific target platform.

### *Data*

Based on the security planning done in the Envisioning phase, you should be documenting the types of data that will be handled in the application. For instance, you might identify that you will be manipulating biographical information, credit card payments, usernames and passwords, etc. You should identify how (in general) each type of data will be stored. That is, you will specify that biographical information will be available only to an authorized user, credit card information will be destroyed after use, and usernames and passwords will be stored (encrypted) in a database file.

## Resources

[Developing a security checklist](#) 

[OWASP Guide to Building Secure Web Applications](#) 

[OWASP Top Ten Web Application Vulnerabilities](#) 

[SANS Top 20 Vulnerabilities](#) 

[← prev : Planning - Tasks](#)

[Developing - Overview : next →](#)

## IT Development

### Developing: Overview

During the developing phase, the team accomplishes most of the building of the system. However, some development work may continue into Stabilizing in response to testing. The developing phase involves more than code development and software developers. All roles are active in building and testing deliverables.

### Tasks and Artifacts

Tasks	Req'd	Role Responsible
Risk Reassessment	Yes	Project Management
Security Implementation	Yes	Development
Peer Review	Yes	Development
Unit Testing	Yes	Development
Test Planning	Yes	Testing
Usability Testing		User Education
Schedule Tracking	Yes	Project Management
Measurements Collection		Product Management
Coding Style	Yes	Development
Reusability Planning		Development

Artifacts	Req'd	Role Responsible
Updated Project Plan	Yes	Project Management
Source Code	Yes	Development
Test Plan	Yes	Testing
API for Reusable Components		Development

### Questions to Answer

- Has the entire scope of this iteration been developed?
- Do the risks identified in Envisioning and Planning still apply or have new risks surfaced?
- Is the project team doing what they said they would do to keep the risks identified in Envisioning and Planning from occurring?
- Has the developer followed the security checklist constructed in Planning?
- Is the peer reviewer satisfied that the quality and readability of the source code complies with this document?
- Has the developer certified that the system is ready for integration testing?
- Has the testing lead documented all test cases for the system?
- Is the user interface intuitive and easy to use?
- Has the project manager appropriately documented any changes to the estimated schedule?
- Is the project team collecting the data they said they would in Envisioning?
- Has the developer identified, implemented, and documented any potentially reusable components in the system being developed?
- Does the project plan accurately reflect changes made to requirements, schedule,

resources, and risks?

## Milestone

### Scope Complete Milestone

At the Scope Complete Milestone, all features are complete and the system is ready for external testing and stabilization. This milestone is the opportunity for customers and users, operations and support personnel, and key project stakeholders to evaluate the solution and identify any remaining issues that must be addressed before the system is released.

[← prev : Planning - Overview](#)

[Stabilizing - Overview : next →](#)

[Developing - Tasks : next →](#)

## IT Development

### Developing: Tasks

#### Risk Reassessment

During the Developing Phase, Risk Reassessment must once again be completed to make certain the success of the project. Since our projects have the possibility of continually changing it is imperative that risks be assessed throughout the life of the project. As with the Planning Phase, the following questions should be answered when completing the Risk Reassessment task during the Developing Phase.

- Do the risks identified still apply to the project?
- Have any additional risks been discovered during the Developing Phase?
- Is the probability of occurrence for each risk still accurate?
- Are appropriate measures being taken to guarantee that the risks do not occur?
- Are any risks that have occurred being properly handled as documented in the Project Plan?

#### Security Implementation

During this phase, the developer should be coding the product in a secure way. That is a very broad goal that is not always practically feasible. To obtain the most feasible level of security, the developer should be following the security checklist developed in Planning.

The only way to ensure that the security checklist is being implemented is to make the checklist an item to be reviewed by the peer reviewer identified in Planning. This also helps in the overall education of the department in developing secure software.

For more information, see the resources listed in Planning.

#### Peer Reviews

##### Description

A peer review (sometimes called a code inspection or a walkthrough) is simply a review of the source code by developers who are not a part of the project development team. The primary purpose of the peer review is to make sure that the code is readable and commented such that code should be easily maintainable and modifiable by others when the members of the development team go on to other projects.

In this task it is more important that the comments and code agree with each other than that either agree with the design documentation. Gaining assurance that the source code is in agreement with the design documentation is the primary task of the test planning task.

However, there may be some desire to establish pre-testing assurances of compliance to the design, making this a secondary purpose of the peer review. Even if this is the case, it is best that the first time reviewers see the code, they



see it without having previously seen the design. This will allow the reviewers to focus on what the code actually does and what the comments actually say, rather than on what they should do and say.

Other benefits of peer review include cross-training in a language or tool which may be unfamiliar and promoting code reusability. When a reviewer sees a piece of code that would serve the entire team better by being broken out on its own and more generalized, this is promoting code reusability.

### Resources

[Code Review Checklist](#) 

[SATC - Software Inspections](#)

[MFeSD - Peer Reviews](#)

[Macadamian - Single Committer Software Development](#)

[Addison Westey - A Little Help from Your Friends](#)

[Peer Reviews](#)

[The Peer Review Process](#) 

## Unit Testing

### Description

The objective of Unit Testing in Developing is to ensure that each component of the system works in isolation from other components. Naturally, during the development of the product, the developer will be testing the interaction between various system components. However, unit testing is only concerned with individual components.

A key method in unit testing is to test your code very quickly after it is written. Identify the conditions under which it succeeds and ruthlessly test. There should be a quick code-test cycle when in Developing.

Unit testing does not need to be a heavyweight, documentation centric activity. It is primarily for the developer's benefit and to ensure that integration testing in Stabilizing can actually occur. Another perspective on unit testing is to write your tests before you write your code. This is known as "Test Driven Development"

### Resources

[Proactive Testing](#)

[testdriven.com](http://testdriven.com)

[Unit Testing Databases](#)

[Unit Testing Cold Fusion Code](#)

## Test Planning

### Description

Test planning occurs after Unit Testing and before Usability Testing. This task exists to create a roadmap to test the entire product (and the interaction between products, when appropriate) to make sure that it complies with the expectations agreed to from the Requirements Gathering task of the Planning Phase. The Test plan should be built in collaboration between the Test and Verification Team and the developer(s).

See Test Plan Artifacts for more details.

## Usability Testing

### Description

Usability testing is a semi-structured way to get feedback on your design from

actual users, not other developers. It is semi-structured because it is more than just focused groups (you aren't just asking people what they "think" about a design) but it is not so rigid as a scientific experiment.

Generally, you should get 5-6 people together for each major part of the system. For instance, if you have a public interface and an administrative interface, you should probably test them separately. Then, you will come up with a test plan for them, giving them just enough instructions so they know what to do, but not too many instructions so they are left to experience the product as the average user would. It is best to be present with the user as they are testing the system and to have more than one person observing and recording notes about where the user trips up and where the design succeeds.

Usability testing should be done in an iterative way, not waiting until the product is completely done to start usability testing. Usually, the testing will uncover more than just cosmetic changes, but changes that sometimes require a major shift in architecture to support the users better. This allows for rapid feedback on design and the ability to fix major user interface design flaws before they become irreversible.

### **Resources**

[UseIt.com - Jakob Nielsen's website](#)

[Why you only need to test with 5 users](#)

[Usable Web - Portal for usability resources](#)

[Web Style Guide](#)

## **Measurements Collection**

During the Developing Phase the tracking of measurements continues as it did in the Planning Phase. Measurements for the project are decided upon during the Envisioning Phase and must be tracked throughout the life of the project.

## **Schedule Tracking**

Once the Project Schedule is completed in the Planning Phase it is now imperative that the Project Manager properly track the progress made based upon the schedule. The Project Schedule should be broken into specific tasks and have a task duration associated with the various tasks. The Project Manager should be involved with the progress that the Developer makes on the schedule. This would include everything from planning the tasks that should be done for a day or week, following up on the task, and finally confirming that a task has been completed. This process should not be one where the Project Manager hands the developer a task list and waits for it to be completed. Schedule tracking is an interactive process. The following is a suggested method for managing the schedule.

1. Meet with the developer to discuss the forecasted tasks for the day or week (whichever timeframe that you feel necessary). Find out if there are any areas that you, the Project Manager can help the developer out with.
2. Check up on the developer mid-way through the task(s) time frame and find out if everything is on track and if the developer needs any help.
3. Confirm that the task(s) have been completed and properly document the finished task.
4. Make any adjustments that are necessary to the task list.

Ultimately it is up to the Project Manager to decide with the Developer how communication will be handled for the tracking of the schedule.

Schedule Tracking is very beneficial to the project because it allows for Project status to be properly reported, it forecasts any anticipated changes in the schedule more appropriately and it allows for the customer to have a more accurate understanding of the status of the project.

## **Coding Style and Recommendations**

### **Description**

#### *Source Code Comments*

1. Every file that contains source code must be documented with an introductory comment that provides information on the file name and its contents.
2. All files must include copyright information.
3. All comments are to be written in English.
4. Write some descriptive comments before every code major section.
5. During Peer Review, if a developer who is familiar with the language in question does not understand your code, it means your code needs documentation.

#### *Naming Conventions (Files, Fields and Variables):*

This section will be implemented in the next iteration of the SDM.

#### *Language-Specific Recommendations:*

#### *Fusebox-based File Structure:*

The logic uses a "star" application flow where all branching starts in and returns to the index.cfm. This is usually accomplished by passing an action to the index.cfm which includes the correct act or dsp file.

#### *File Name Prefixes:*

- act – action template that executes some business logic on the server side. For example, add contact to a database.
- dsp – display templates, mostly html.
- qry – queries, stored procedure calls that populates the display templates.

## **Reusability Planning**

### **Description**

Reusability helps software development in maintainability, modularity, portability, productivity, interoperability, reconfigurability, commonality, adaptability, reliability and quality.

One of the best ways to plan reusability is to think of programming as a collection of components. A component is defined as "a nontrivial, nearly independent, and replaceable part of a system that fulfills a specific function within a system's architecture."

For components to be reusable they need to be clearly specified, portable and adaptable. To make a component truly reusable, it has to be designed in a way so as to allow adaptation. Adaptability is necessary because the developers of a component cannot predict the requirements for all the systems and scenarios in which it will be reused.

Repositories of such components need to be made available and easily accessible to developers.

[← prev : Developing - Overview](#)

[Developing - Artifacts : next →](#)

## IT Development

### Developing: Artifacts

#### Updated Project Plan

The Project Plan that was created in the Envisioning Phase and updated in the Planning Phase must once again be reexamined in the Developing Phase. One of the major areas that will continually need to be addressed is the Risk Assessment Plan that is document in the Project Plan. Risk Assessment is a task that must be completed throughout the life of the project and as changes are made they should be properly documented in an updated version of the Project Plan. Some other areas of the document that may be updated are the Project Team and the Security Information.

The preferred method for updating the Project Plan is to create a revision of the existing document. Previous versions of the Project Plan should be saved so that they can be referred to in the future if historical information is needed.

#### Source Code

##### Description

All source code (including SQL scripts, views, stored procedures, and CL scripts) should be backed up in one of the code managment systems (Source Safe or MKS Implementer).

All source code should be peer reviewed and tested before deployment.

For further guidance, see the Coding Style and Recommendations section of the Developing: Tasks page.

##### Resources

[Wikipedia - Source Code](#)

#### Test Plan

##### Description

The Test Plan is the set of documentation that results from the Test Planning task. It will contain testing scenarios which are used to identify discrepancies in the product. These scenarios should be reproducible by the developers and other testers. Where appropriate, the identified measurements should be used as the benchmarks for testing the product. Any problems encountered during testing should also be documented.

The general rule is, when in doubt, let nothing be assumed for someone trying to reproduce the discrepancy. So, document the tools you used, the computing environment you tested in, and anything else which might be useful to anyone intending to reproduce the problem.

##### Resources

[Test Plan Template](#) (Revision 1.1 - 05/04/2005)

#### API Documentation

Application programming interface (API) is a set of definitions of the ways in which one piece of computer software communicates with another. It is a method of achieving abstraction.

One of the primary purposes of an API is to provide a set of commonly-used functions—for example, to draw windows or icons on the screen. Programmers can then take advantage of the API by making use of its functionality, saving them the task of programming everything from scratch. (See Reusability Planning)

Everytime a developer writes a reusable component, there must be a API document that would allow other developers to reuse that component.

What should be documented on the API?

1. Description of what the component does
2. Usage notes (exceptions, flags, etc)
3. List of attributes, parameters, return values (if they are required or not, and default values)
4. Example of code usage

[← prev : Developing - Tasks](#)

[Stabilizing - Overview : next →](#)

## IT Development

### Stabilizing: Overview

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

### Tasks and Artifacts

Tasks	Req'd	Role Responsible
Establish target deployment date	Yes	Project Management
Deployment Planning	Yes	Testing
Training		User Education
User Acceptance	Yes	Project Management
Security Testing	Yes	Testing
Stakeholder Communication	Yes	Project Management
Risk Reassessment	Yes	Project Management
Schedule Tracking	Yes	Project Management
Measurements Collection	Yes	Project Management
Artifacts	Req'd	Role Responsible
Deployment Sign-off Document	Yes	Project Management
Deployment Planning Document	Yes	Testing
System Admin Handoff Document		Logistics Management

### Questions to Answer

- Do the risks identified in Envisioning, Planning, and Developing still apply or have new risks surfaced?
- Is the project team doing what they said they would do to keep the risks identified in Envisioning, Planning, and Stabilizing from occurring?
- Are all team members (as well as outside IS resources) aware of and ready for the deployment?
- Have the customers and stakeholders accepted the product as developed?
- Have the end users been trained on the new system (or the new features of an existing system)?
- Has the project team communicated with all stakeholders according to the communication plan generated in Envisioning?
- Has the testing team verified that the system is ready for deployment?
- Does the NOC have enough information about the system to be able to respond to crises and support the system in production?
- Does the project team have a repeatable and testable way to deploy the system into staging and production?
- Is the project team collecting the data they said they would in Envisioning?

### Milestone

## Release Readiness Milestone

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

[<< prev : Developing - Overview](#)

[Deploying - Overview : next >>](#)

[Stabilizing - Tasks : next >>](#)



## IT Development

### Stabilizing: Tasks

#### Establish Target Deployment Date

The project target deployment date should have been established in the planning stage of the project. At this point, a more accurate deployment date should be settled upon. This date should be coordinated with the following parties.

- The Verification and Testing Unit
- The NOC
- The developer
- The Director of IT Operations (if downtime is required)
- The customer

Additional time of two weeks minimum, should be built into the schedule to allow the Verification and Testing Team to examine the product for any defects or errors. This team should be made aware of the deployment schedule as determined in the planning stage. The size of the project will determine the length of time required by the Verification and Testing Team to complete testing.

If any down time is necessary, this downtime must be approved by Aaron Mathes, Chief Operations Officer for Information Services. Downtime notices should be placed on the University website 5-7 days before the downtime.

#### Deployment Planning

Verification and Testing Unit should work together with Developer, Project Manager and System Developers to create the Deployment Plan Document. Not all plans will require the involvement of all parties, it will mostly depend on the size of the project and its importance. Minor projects will be incorporated into the regular "defect" deployment that happens twice a week (see Web Patch Deployment Procedures on [molly/4200/Verification and Testing/Deployment Procedure.doc](#)), and may not need a Deployment Plan Document.

#### Training

After the system has been deployed in the staging environment, training should occur for all end users. This includes administrative offices, as well as students. Preferably, the training should be carried out through the trainers in DISC so they can add the new product or features to their portfolio of training. There might be instances, however, where the training will be carried out by the user departments. The project manager should work with the DISC trainers and the user department to determine the best course of action.

#### Security Testing

##### Description

The testing team should utilize the security checklist generated in Planning and test the application based on that checklist. Also, they should use any tools available to test the general security of the product. Any vulnerabilities identified should be identified and cataloged as a defect. The developer should then resolve the defect before deployment.

##### Resources

[OWASP Testing Project](#)

## **Risk Reassessment**

During Stabilizing, Risk Reassessment must once again be completed to make certain the success of the project. Since our projects have the possibility of continually changing it is imperative that risks be assessed throughout the life of the project. As with the Planning Phase, the following questions should be answered when completing the Risk Reassessment task during the Stabilizing Phase.

- Do the risks identified still apply to the project?
- Have any additional risks been discovered during the Stabilizing Phase (especially risks related to deployment)?
- Is the probability of occurrence for each risk still accurate?
- Are appropriate measures being taken to guarantee that the risks do not occur?
- Are any risks that have occurred being properly handled as documented in the Project Plan?

## **Schedule Tracking**

Now that the project is in Stabilizing, the completion date for the project should be more certain. At this point, a review should be done of the schedule as described at the beginning of Developing. If the project is running later than anticipated, this should be communicated to the project team and all stakeholders.

## **Measurements Collection**

During Stabilizing the tracking of measurements continues as it did in Planning and Developing. Measurements for the project are decided upon during the Envisioning Phase and must be tracked throughout the life of the project.

Most notably, the project team should be tracking the number of defects found during testing.

## **Stakeholder Communication**

An important task within every project is stakeholder communication. Every project has a number of stakeholders that will be affected by the release of the product. It is important that all stakeholders affected by the project are communicated to when a product is deployed. This communication can take place through email, phone conversations, or face-to-face meetings; whichever is more appropriate.

The project plan developed in Planning should have included a stakeholder communication plan. In this phase, it is important to make sure all communication was carried out as planned.

## **User Acceptance**

Throughout the Stabilizing Phase it is important that the main end user(s) is involved in the decision for the deployment of the product. The user should give his or her approval of the new features and changes to the product to confirm that the new development is satisfactory for the end user to complete his job or function. This communication to the end user can be done through group demos, one-on-one walkthroughs, and even the training that is completed during the Stabilizing Phase.



## IT Development

### Stabilizing: Artifacts

#### Deployment Sign-off Document

The deployment sign-off document is the agreement by the members of the team that they are ready for the deployment of the developed product. This document indicates the deployment date, what will be deployed, and includes signatures from the following individuals:

- Project Manager indicates that all team members have done what they need to do to deploy the product
- Product Owner indicates that they authorize changes to their system
- NOC Director indicates that they are ready to deploy the system
- IT Operations director indicates that campus stakeholders have approved downtime (if necessary) and that he/she approves the system to be deployed
- Verification and Testing Lead indicates that the product has been tested and ready to be deployed

#### Example Document

[Commencement Photo Deployment Approval](#)

#### Deployment Planning Document

Deployment will happen in two major stages:

##### Stage 1:

*Development Environment -> Testing Environment*

The Testing & Verification Unit will run tests on the deployed resources in the Testing Environment and communicate recommendations to the developers of what may need to be changed for a successful deployment into production.

Once the changes are made (if any) the process of deploying to the Testing Environment will be repeated where the Testing & Verification Unit will run the tests again, once the deployment to the Testing Environment is successful, Stage 2 can begin.

##### Stage 2:

*Development Environment -> Production Environment*

Once Stage 1 successfully completed, and the assumption that Testing Environment is mirrored from the Production Environment that means (theoretically) the patches created for Stage 1 should work on Stage 2. The Testing & Verification Unit should be able to run the same tests they ran on Stage 1 and get the same results. If the results cannot be duplicated that probably means that both Production and Testing Environment are out of synchronization and the issue should be taken care of appropriately.

## **Personnel Requirements**

Organize your deployment team and then assign specific roles to team members.

If the necessary resources cannot be offered for deployment it will be safer to delay the deployment until the resources are made available.

## **Organizing Your Deployment Teams**

Usually team members will include:

- \* Developer in charge of Application System
- \* At least one System Administrator that will be available to carry the deployment
- \* At least one Tester to verify and test the Application System after it has been deployed.

## **Current Computing Environment**

Everyone in the team should receive a short briefing from System Administrators of what the current server and network environments looks like and what risk the deployment in questions may pertain to the environment.

## **Application Requirements**

The bits and pieces and configuration files that need to be deployed. A list of all files, SQL scripts, Application Server configuration, server permissions, web applications configurations should be listed.

## **Establishing Standards and Guidelines**

The ultimate goal of creating a Deployment Planning Document is to automate any future deployment for a given Application. As Deployment Planning Documents and procedures and logic are generate for deployment they should be done following Department standards and guidelines so they can be duplicated with little effort as possible.

## **Deployment Design**

Currently Programming Services have a patch deployment design based on the deployment of patch and release files to a network share. These files are deployed to the Staging Server, tested, and once correctly working, the deployment takes place from this network share into the production server.

## **Risk Assessment**

Depending on the Current Computing Environment briefing, the Testing & Verification team will have to evaluate the risks with the System and Network Administrators, who have the ultimate power of veto on a deployment.

## **Problem Escalation Plan**

The escalation plan should be the set of tasks necessary to revert the deployment in case something unexpected happens. The original version of the Application being deployed should always be available for a "rollback" in case the Deployment is not successful.

## Communications Strategy

The communication strategy should be a simple common understanding of all parties involved in deployment on where, when and how they can be contacted during the deployment tasks.

## Testing & Verification Plan

See Testing & Verification Unit Documents.

## System Admin Handoff Document

To facilitate the handoff of new server systems from Development to Operations, a handoff document should be created. This should be created for all new or reconfigured servers. If the project is simply adding new functionality to an existing application that has already been deployed, or adding a new application to an existing web server, this is not necessary. The document should contain the following sections:

### General Configuration

This includes things like the OS, IP configuration, applications installed, service accounts and permissions, machine name, DNS entries, where it is racked, what systems it interfaces with, etc.

### Testing Plan

This should be a copy of the testing plan that was used during Stabilizing. The point here is that if a NOC employee has to do maintenance to the server, they can run through the testing plan to make sure it is still functioning correctly. Without this information, it is very difficult for them to know if the machine is truly "up" or not.

### Security Benchmarks

This section should include the initial security configuration of the machine. If the CIS benchmark scoring tool could be used, this should also include the initial score.

### Response Plan

This section should identify any possible problems the systems developer has encountered or thinks the NOC might encounter during maintenance of the machine and how to go about responding to the issue. This basically serves as the NOC's playbook that they can add to or modify as things change in production.

### Example Document

[Arial Campaign Enterprise Handoff](#)

[<< prev : Stabilizing - Tasks](#)

[Deploying - Overview : next >>](#)

## IT Development

### Deploying: Overview

During this phase, the team deploys the core technology and site components, stabilizes the deployment, transitions the project to operations and support, and obtains final customer approval of the project. After the deployment, the team formally closes the project.

#### Tasks and Artifacts

Tasks	Req'd	Role Responsible
Deployment	Yes	Logistics Management
Defect Resolution		Development
Test in Production	Yes	Testing

Artifacts	Req'd	Role Responsible
Project Closeout Document	Yes	Project Management

#### Questions to Answer

- Has the system been deployed to production successfully?
- Has the system been tested in production?
- Have all defects identified been resolved?
- Do the users understand how to get support on the product after the project has been closed?
- Does IT Operations have the necessary documentation and knowledge to fully support this system?

#### Milestone

##### Deployment Complete Milestone

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

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





## IT Development

### Deploying: Tasks

#### Deployment

This section describes Development and Engineering's responsibilities during the actual deployment of the system to production. This could include either the deployment of new files to a web server, or when a new server is first being used by new customers. This may vary depending on the size of the project. The project manager and other resources should use their discretion to determine how this should be carried out.

The project manager and user education lead should be heavily engaged with the users at this point in time. It might be useful to be physically present in their department at the time of deployment. This gives them the ability to watch the customers using the new system and react immediately to deployment issues or other defects.

The developer (either system or application) should probably be onsite with the NOC when the system is actually deployed and then should move to be onsite with the customers to see them use the system.

The developer should also be available for the Verification and Testing Unit if necessary.

#### Defect Resolution

In the deployment stage, there will be a period of time during which the number of defects identified in the product will sharply increase. The bulk of these defects should be identified and resolved before deployment to the production system, but it is possible that some defects may slip through due to configuration differences or limitations in testing resources available. When such defects are encountered immediately following a deployment, the fixing of such defects will be considered development's highest priority. No software maintenance requests (i.e. tickets) are needed from the customer in order to repair these defects, however development may choose to use such software maintenance requests to track the defects identified and their resolutions.

#### Testing in Production

The Verification and Testing Unit will run the tests created during the Stabilizing Phase, one last time on production to make sure that deployment was successful. If the deployment does not pass the test(s), Developer(s), Project Manager and System Developer may be contacted for generation of "defect resolution" patches under Emergency mode.



## IT Development

### Deploying: Artifacts

#### Project Closure Document

Administrative closeout involves gathering and centralizing project documents, obtaining signoffs, and communicating the finish of the project. It also provides a reference source that can be used to improve the success of future projects. A communication plan is created to inform all project stakeholders that the project has now been closed.

The document should include the following:

##### **Project Title Page**

- Project Name
- Department
- Product
- Prepared by
- Project Team/Roles

##### **Executive Summary**

- Project Overview
- What were the original goals and objectives
- Success criteria
- Were the goals and objectives met

##### **Project Goals**

- Review the milestones and success of the project
- Outstanding issues, risks, and recommendations
- Lessons Learned (which processes worked well; which did not work well)

##### **Why Project is being closed**

- Completion/Canceled
- Closing Activities

##### **Post-Project Tasks**

- What actions were not yet completed?
- What deliverables are not yet achieved?
- Which training requirements are still outstanding?

##### **Project Closure Recommendations**

Gain project closure approval from the Project Sponsor, including agreement that the project has fulfilled all of the requirements as documented and the Project Sponsor is satisfied that all outstanding items have been satisfactorily addressed.

##### **Project Approvals and Agreement**

Product Sponsor and team members sign name and date the sign-off document  
Service Level Agreement states:

*In IT Development and Engineering, our agreement to you, the customer, is to*

*maintain any software that we have released for the university. If a problem occurs with the current system, a Help Desk ticket should be submitted and a response will be received by the customer within 24 hours. For any new requests, a Project Request Form (located on the IT Development and Engineering web page) should be submitted and the request will be prioritized for the next version of the product or for a new product.*

[<< prev : Deploying - Tasks](#)