SourceForge Collaborative Development Process

SourceForge Collaborative Development Process
SourceForge Enterprise Edition
Part Number: 98-0056
Last Revision: April 2005

Restricted Rights

Feedback

Please send comments and questions to:

CDP Management
VA Software Corporation
46939 Bayside Parkway
Fremont, CA 94538 U.S.A.

cdpfeedback@vasoftware.com

Contents

CHAPTER 9 SourceForge Best Practices 245

CHAPTER 1
Introduction

This chapter includes the following information:

- Introduction
- Audience
- How to Use this Manual
- Conventions
- Phase Deliverables Diagrams
- Suggested Owners Notations
- CDP and the Sarbanes-Oxley Act of 2002
- SourceForge and Agile Development

Introduction

Today, many organizations are closely examining process methodology in an effort to improve their overall efficiency and productivity. The growing trend toward outsourcing application development also demands a strong process infrastructure. Organizations are looking to establish best practices for effectively managing the costs, schedule, and quality of both internal and outsourced development projects.

To that end, process improvement initiatives have become increasingly important and useful in today's business climate. In addition to improving operations, a mature process infrastructure can often open up new business opportunities and markets, and help reduce the hidden costs often associated with outsourcing development projects to offshore vendors. In many cases, however, organizations struggle with outdated, insufficient, or overly complicated processes that often create rather than eliminate barriers to efficiency.

Existing tools often do not lend themselves to supporting development process, and those that do often support only existing, predefined methodologies, and cannot be adapted to the specific requirements of an organization without expensive customization. The SourceForge® Collaborative Development Process™ (CDP™) provides a practical and flexible solution that enables process attainment and improvement, without the unnecessarily cumbersome overhead often associated with other process improvement methodologies.

This manual provides users with a broad range of content, covering a variety of process models, specific practices, and software products. While the content presented draws from a variety of sources, it all supports a common goal:

> ***Provide organizations with the tools, processes, and best practices needed to rapidly transform process theory into development reality.***

Listed below is a brief description of each of the major sources of materials presented in this manual and the relationships among them.

SourceForge Collaborative Development Process

The CDP is a comprehensive lifecycle model developed over many years by a highly skilled team of VA Software Engineering, Quality Assurance and Product Management professionals. The products, processes, and quality standards detailed in the CDP are drawn from a wealth of collective industry experience developing high quality software and leverage many of the best practices employed by other world-class software development organizations in their process methodologies.

The VA Software Product Development organization consistently follows the CDP model in the development of SourceForge® Enterprise Edition and other, internal software products. Offshore development partners to whom VA Software subcontracts development work also follow the CDP model, ensuring high visibility into progress, consistently high quality deliverables, and effective management of schedules and costs.

VA is continually analyzing and improving the CDP based on rigorous internal use, research into third party process improvement models, such as the Software Engineering Institute Capability Maturity Model®, and advances in the functionality of SourceForge and other software tools.

The CDP is a comprehensive, end-to-end lifecycle model, suitable for deployment in an organization beginning a process initiative from the ground up. However, it is also designed in a modular fashion, allowing organizations with a level of established process to select the areas that best supplement their existing process infrastructure.

SourceForge Enterprise Edition is used extensively throughout the CDP, although the extent of its required use is customizable based on the needs and existing infrastructure of an organization.

Software Engineering Institute Software Capability Maturity Model

The Software Engineering Institute (SEI) Capability Maturity Model for Software (SW-CMM®) is a framework that describes the key elements of an effective software process. Encompassing an evolutionary improvement path from ad hoc, immature processes to mature, disciplined processes, the SW-CMM defines five process maturity levels ranging from Level 1 to Level 5. Each maturity level is comprised of Key Process Areas (KPAs), defining a series of activities that, when performed consistently and effectively, help organizations meet established goals for process capability at a defined maturity level.

The key premise of the SW-CMM is that process maturity equates to the improved ability of an organization to meet cost, schedule, functionality, and product quality goals. ***While the SW-CMM framework provides a series of goals associated with each maturity level, it does not provide a methodology for achieving them.*** The CDP provides this methodology, and when combined with SourceForge, fully supports all of the goals associated with achieving a minimum of SW-CMM Maturity Level 2 or 3.

Each stage of the CDP defines one or more processes that equate to goals associated with one or more SW-CMM Level 2 or 3 KPAs. Chapter 2 of this manual provides an overview on using the CDP and SourceForge to achieve SW-CMM Level 2 or 3. *SourceForge and the Capability Maturity Model for Software, A Supplement to the SourceForge Collaborative Development Process* provides detailed mappings for using SourceForge and the CDP to fulfill each key practice (commitment, ability, activity, measurement, and verification.) Specific SW-CMM references are provided throughout the CDP chapters and checklists are included at the end of each phase to ensure that required activities are completed.

It is important to note that the SW-CMM references are not a required element of the CDP, nor is the CDP based exclusively on the SW-CMM. The references are provided for the benefit of users preparing for a formal SW-CMM appraisal, and for those interested in using the SW-CMM framework as a basis for or supplement to an internal process improvement initiative.

SourceForge Enterprise Edition

SourceForge Enterprise Edition is VA Software's flagship collaborative software development product. SourceForge combines collaborative development tools with real-time metrics and reporting. It provides process-enabling features such as centralized document management, fully traceable requirements and task management, archive capabilites for all project-related communication and extensive management visibility into, and control of, project performance. SourceForge is a key element of the CDP. Using SourceForge is also a critical factor when following the CDP guidelines for achieving SW-CMM goals.

Recommended uses for SourceForge are included throughout this manual, as are extensive guided best practices for using SourceForge to achieve specific CDP and SW-CMM process objectives.

SourceForge provides a complete toolset to support the CDP and achieving SW-CMM Level 2 or 3. However, in some cases, non-SourceForge tools may be substituted as alternatives to the exclusive use of SourceForge. For example, an organization may have legacy or commercial document management or bug tracking systems that represent a significant investment, or have internally developed systems that meet a company specific need not addressed by SourceForge. In such cases, the specific processes described in the CDP may be modified as needed. Furthermore, SourceForge is engineered as an extensible system. SourceForge integration with third party or in-house developed tools can be achieved by you alone or in partnership with the VA Software Professional Services organization.

Audience

The CDP is designed for software development professionals performing a wide variety of functions. The CDP processes can be applied to in-house as well as outsourced development projects. Users who will benefit most from the material presented are those working as software developers, testers, system engineers, quality assurance staff, product managers, business analysts, process improvement specialists, information technology (IT) managers responsible for compliance with the Sarbanes-Oxley Act of 2002, and those with a management or oversight responsibility for in-house or outsourced software product development and associated functions. Professionals in other, related fields such as product marketing, technical support, training, sales engineering, and professional services will also benefit.

Software is developed by different types of organizations for a variety of purposes and customers. The processes defined by the CDP are intended to assist software development professionals working in Product Development, Software Engineering, System Engineering, IT, Research and Development, and similar organizations engaged in developing software products. It is important to note that references to the “product” refer not only to products that will be packaged or otherwise offered directly for sale, but to any software work product that will be used by customers internal or external to your organization.

References to the “customer” refer to the intended end user of your completed work product, whether or not that work product represents the product in its final form. For example, your completed work product may be intended for use only as a component in a larger product. This is often the case with the development of embedded software systems where the "customer" may be another team in your company responsible for integrating software and dedicated hardware devices. The customer of your product may range from a single person or group within your organization to millions of anonymous users worldwide.

Each phase of the CDP describes considerations for producing various types of products for different types of customers. Descriptions of activities and deliverables are generally written for a broad audience and propose suggested content options for users developing products with specific goals. Some activities or deliverables may not be applicable to the type of products you are developing. If so, you may wish to modify certain areas of the CDP to best suit your specific needs.

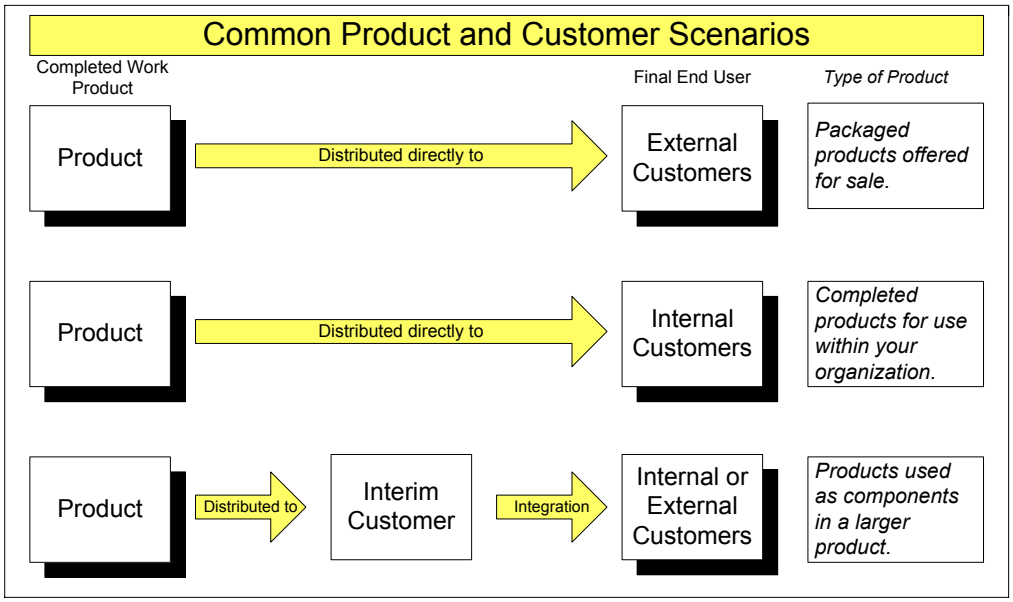


Figure 1. Common Product and Customer Scenarios

The diagram above represents several of the common product and customer scenarios described in the CDP. Software products may be completed and distributed directly to their final internal or external customers. They may also be completed and released to another interim customer, who may be internal or external to your organization. This interim customer is then responsible for integrating your product with others to form a larger product, or integrating your product with existing systems and infrastructure, prior to making it available to its final internal or external customers.

How to Use this Manual

This manual is organized according to the phases of a typical software product lifecycle. Each phase consists of a series of activities and deliverables that when completed, represent the achievement of a major milestone in the product's development.

The phases defined in the SourceForge Collaborative Development Process (CDP) are:

1. Product Definition
2. Design
3. Development
4. Release Preparation
5. Release
6. Sustaining Engineering

A Cross-Lifecycle Principles chapter is also included that describes the concepts and activities you will be working with throughout all phases of the CDP.

As indicated by the following diagram, the CDP is a continuous process, with the data and knowledge collected during the post-release analysis of one development project providing input into the next. In this way, a persistent pattern of continuous process improvement is achieved.

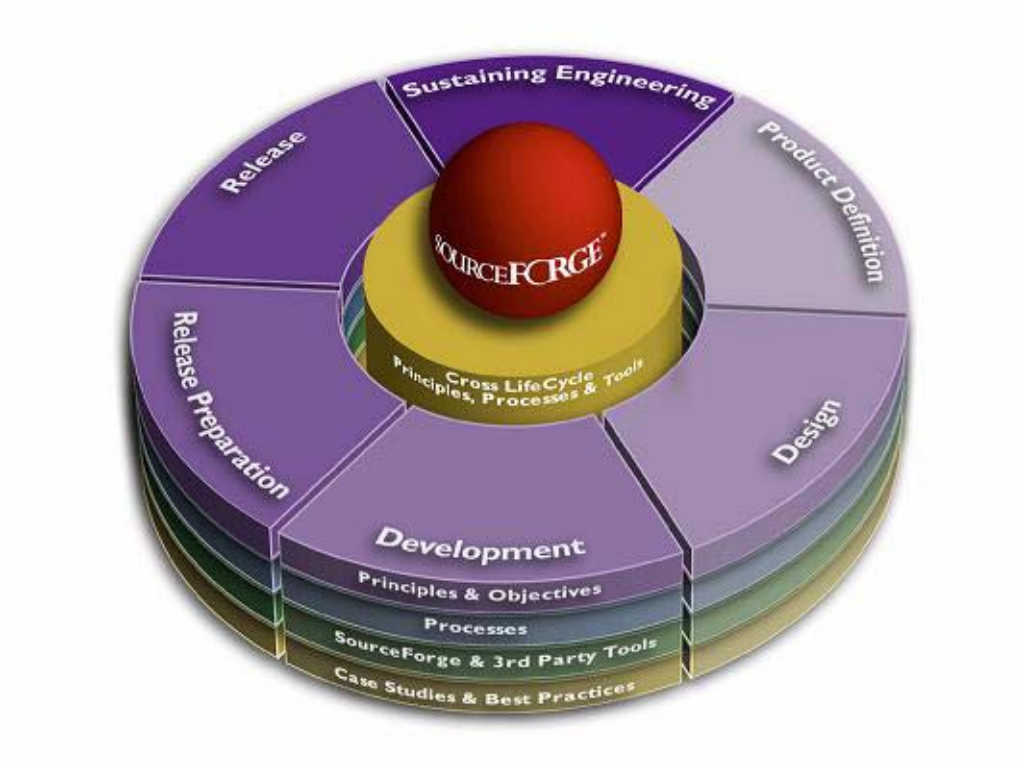


Figure 2. SourceForge Collaborative Development Process

In each chapter, the principles and overall objectives of the phase are defined. Specific detail is provided on the inputs and outputs of each phase, with detailed descriptions of all required and recommended processes, activities, documents, and other materials. Flowcharts depict recommended workflow processes, and Guided Best Practices provide methods for using SourceForge to enable specific CDP activities. Customizable templates are also included for many of the required documents.

For users interested in the SEI Capability Maturity Model (CMM), instructions are included for using SourceForge and the CDP to support the goals associated with achieving a Level 2 or 3 rating against the CMM for Software (SW-CMM.)

Conventions

Typography

The following table shows the specific typeface elements used in this guide.

Convention	Usage
Italic	URLs (*http://www.vasoftware.com*) References to additional documentation (*SourceForge Enterprise Edition 4.2SourceForge Enterprise Edition 4.2 User Guide*)
Verdana	SW-CMM Notes
Verdana in table format	Procedures

Numbering

All activities and deliverables described in the Phase Deliverables section of each chapter are identified numerically. The same number is used to represent the activity or deliverable in both the flowchart and the accompanying Phase Deliverables description. Activities and deliverables are identified by both the phase number and the deliverable number.

For example, Document B in Figure 3 on page 11 is identified as:

1.2. Document B

The number preceding the decimal indicates that this document is a deliverable in Phase 1, or the Product Definition phase.

The number following the decimal indicates that this document is the second deliverable described in the Phase Deliverables section of the Product Definition chapter.

Templates

When used to identify templates, the number includes a third digit used to represent the document revision number.

For example, the template for Document B in the above example may be identified as:

1.2.4. Document B

The number 4 would represent the fourth revision of the template.

The number 7 is used as a first digit to represent templates associated with cross-lifecycle principles.

For example, the template for the Training Worksheet is identified as:

7.8.1 Training Worksheet

Workflow Processes

All workflow processes included in this manual follow industry standard flowcharting methodology as described below.

Flowchart Display

All workflow processes are displayed in horizontal "swimlane" flowchart format. Each swimlane represents the tasks generally associated with a specified work group. In order to simplify the flowcharts, and to allow for differences in task allocations among organizations, all flowcharts in this manual are limited to a maximum of five (5) swimlanes. Please see the Suggested Owners Notations section for a description of each work group and their suggested functions.

The following figure provides an example workflow process in swimlane flowchart format.

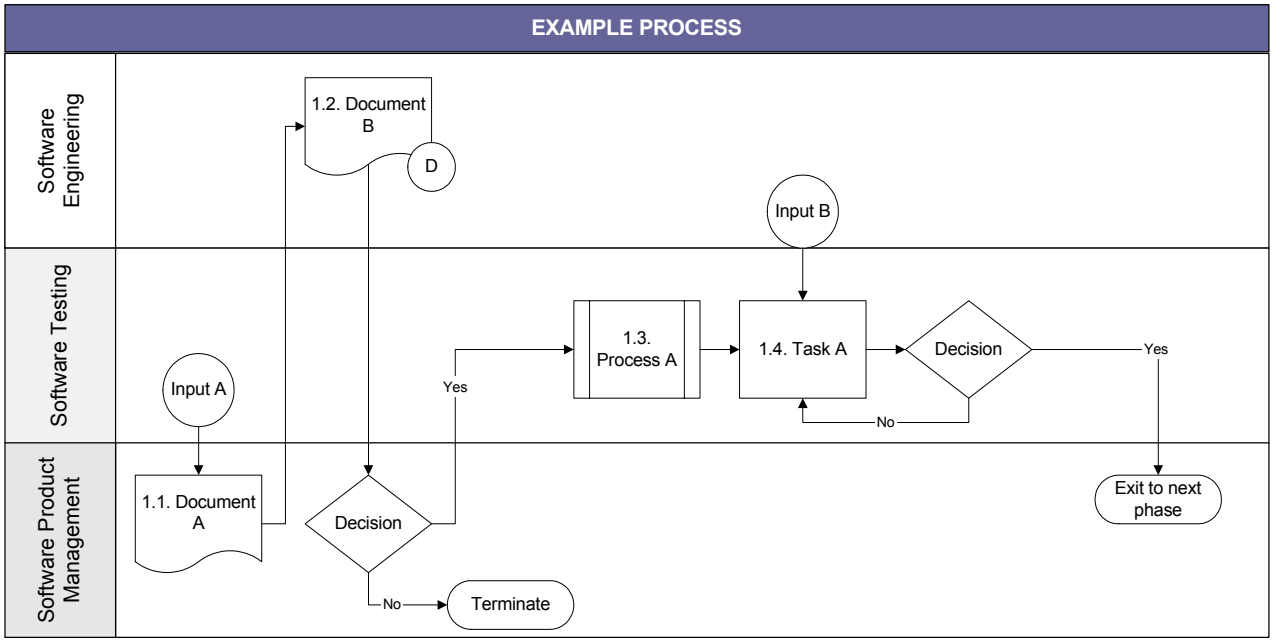
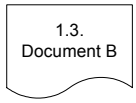
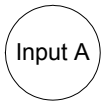
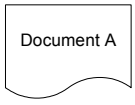
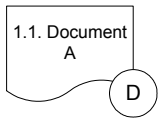
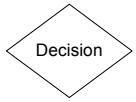
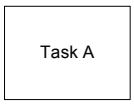
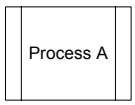
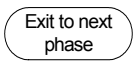


Figure 3. Example Workflow Process

Flowchart Notations

The following flowchart notations are used in the workflow processes included throughout the CDP materials.

Description	Notation
Flowchart items that have supplementary text are numbered according to the phase number and deliverable number. The example shown indicates Phase 1, Deliverable 3.	
Inputs required to begin or complete a deliverable. Note: Input symbols indicate a item completed outside of the phase; therefore the task allocations indicated by the swimlanes do not apply.	
Documents	
Documents subject to the Iterative Document Review and Approval Process Note: See Figure 21 on page 276 for additional detail on this process.	
Decisions Note: Only binary 'Yes/No' decisions are represented in the CDP.	
Processes or tasks	
Processes or tasks accompanied by a SourceForge Best Practice	
Terminators, e.g. completing or terminating the project, or exiting the phase	

Activities and Deliverables

All activities and deliverables are identified by number in each chapter's Workflow Process Diagram. Activities and deliverables for which templates are provided are identified by the following symbol: 

Recommended uses for SourceForge are identified by the following symbols:

SourceForge Feature	Notation
Document Manager	
File Release System	
Discussion Forums and Mailing Lists	
Integration with Software Configuration Management Tools (CVS)	
News	
Reporting	
Tracker	
Monitoring	
Tasks	

Deliverables for which peer review or a Software Quality Assurance audit are recommended are identified by the following symbols.

Recommended Activity	Notation
Peer Review	
Software Quality Assurance Audit	

Descriptions of each activity or deliverable, suggested contents and completion criteria, and dependencies on other activities or deliverables are provided with each item. For items where peer review is recommended, the suggested method of peer review is included. Considerations for tailoring deliverables to meet specific organizational needs are indicated where appropriate, and suggested uses for SourceForge are included at the end of each section.

SourceForge Version

The SourceForge functionality described in this version of the CDP manual is that available in SourceForge Enterprise Edition 4.2.

Phase Deliverables Diagrams

Each phase of the CDP includes a diagram of the activities and deliverables completed during that phase. This diagram appears at the beginning of each chapter and serves as an introduction to the material that will be covered in detail throughout the chapter.

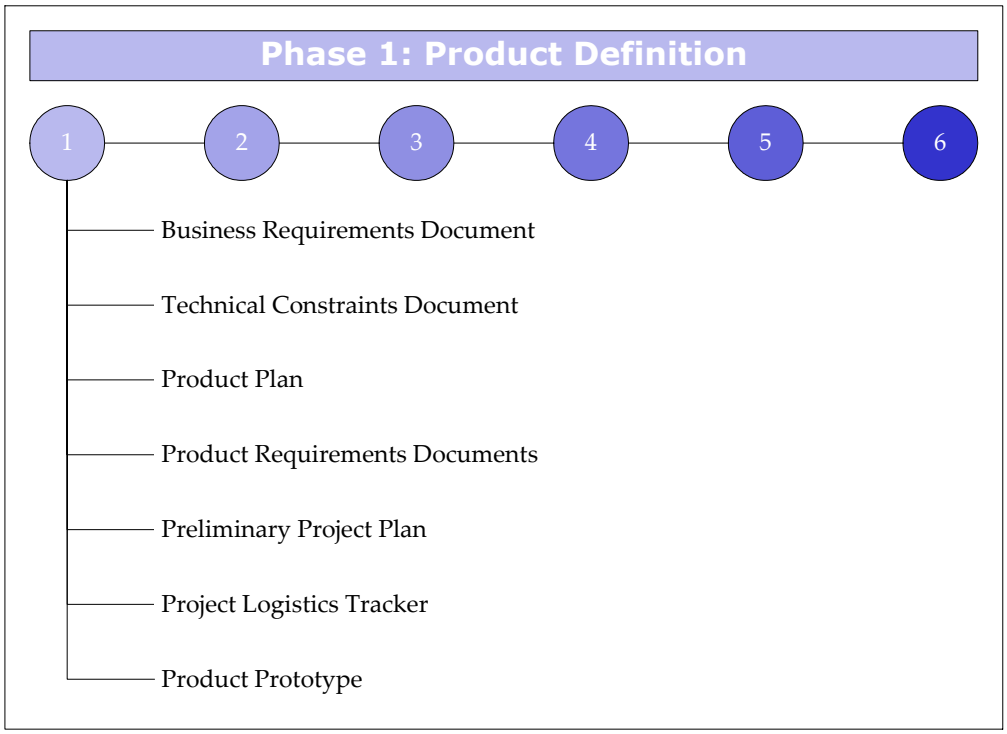


Figure 4. Example Phase Deliverables Diagram

The example diagram above indicates that the activities and deliverables completed during Phase 1, Product Definition include the Business Requirements Document, Technical Constraints Document, and the remaining items shown. Each item is included in the Phase Workflow Process flowchart and described in detail in the Phase Deliverables section of the chapter.

Suggested Owners Notations

Allocating tasks to an appropriate responsible individual or team is a key element in any process methodology. The CDP, when used internally at VA Software, defines very specific responsibilities associated with individuals and functional groups within the organization. However, responsibilities associated with individuals and functional groups may vary widely among organizations, limiting the value of a strictly defined, workgroup-based methodology.

Therefore, in order to provide a level of guidance regarding suggested responsibilities for the CDP activities, while still leaving ample room for modification at an organizational level, all flowcharts and Suggested Owners notations are limited to five (5) general categories. The Suggested Owners categories are defined as follows:

- **Software Engineering:** The Software Engineering category refers to all staff primarily engaged in technical activities related to software development. This can include developers, system architects, release engineers, technical managers, and other engineering staff.
- **Software Testing:** The Software Testing category refers to all staff primarily engaged in software quality and testing-related activities. These activities can include performing software testing, writing testing-related documentation such as test plans, test cases, and test scripts, managing field testing activities, analyzing test results, managing Software Testing staff, and other software testing activities.
- **Software Quality Assurance:** The Software Quality Assurance category differs from the Software Testing category, although they are often used interchangeably. Software Testing staff are primarily engaged in testing activities designed to ensure high software quality, while the Software Quality Assurance category is usually applied to staff engaged in a wider variety of quality management activities. Leading process improvement initiatives and conducting periodic audits on the quality of various deliverables are examples of activities often managed by Software Quality Assurance teams. Other responsibilities often associated with the Software Quality Assurance category include ensuring processes are defined and followed for each product development phase, ensuring that all phase entry and exit criteria are met, and validating the quality of the entire product, including documentation, packaging, training, and sustaining engineering processes.

- **Software Product Management:** The Software Product Management category, also known as Project or Program Management, refers to all staff engaged in overall management of the product and the development project required to produce it. Some organizations refer to members of this category as Business Analysts. Specific activities can include defining product requirements, project planning, conducting market and competitive research, managing cross-functional dependencies, managing the product release, and other management functions.

  For purposes of the flowcharts and Suggested Owners notations in this document, when an activity or deliverable falls outside the ownership of the Software Engineering, Software Testing, or Software Quality Assurance groups and is not specifically assigned to an Other category, it is generally assigned to Software Product Management. Some examples of critical functions assigned to Software Product Management include Product Marketing, Technical Support, and Training. This does not imply that Software Product Management is responsible for activities and deliverables associated with these functions; only that Software Product Management, as managers of the overall development project, should be responsible for ensuring their completion.
- **Other:** In some cases, an additional category will be used when a specific activity is called for that clearly falls outside the responsibility of the functional groups defined above. An example of such an activity is Senior Management approval of significant product-related documentation such as the Business Requirements Document. You may also define additional categories that represent other functional teams within your organization when such teams perform tasks integral to the software development and sustaining engineering process.

It is important to keep in mind that these are broad groupings of responsibilities that may or may not be applicable in your organization. They should be viewed as recommendations only, and should be modified as needed to best suit the structure of your organization.

In addition to the functional groups defined above, organizations pursuing CMM or other process improvement goals often create a separate group dedicated to managing the process improvement activities of the organization.

Software Engineering Process Group[SM]: A Software Engineering Process Group is a team made up of representatives from Software Engineering, Software Testing, Software Quality Assurance, Software Product Management, and similar functions. The goal of such a group is to actively manage the process improvement activities of the organization. A representative from the Software Engineering Process Group should participate as a core team member in each software development project. This representative acts as a consultant to ensure that the processes established by the organization are understood and followed, that the value of these processes is continually measured, and that the process-related activities of the project are coordinated with those of the rest of the organization.

CDP and the Sarbanes-Oxley Act of 2002

On 27 May 2003, the U.S. Securities and Exchange Commission (SEC) adopted new rules governing how companies comply with financial reporting requirements.

- Managers must certify the financial reports issued by their company
- Managers must describe and evaluate the effectiveness of the internal controls they use
- A third-party auditor must certify these findings
- Violations can result in jail time for CEOs and CFOs

According to a leading information technology (IT) industry research firm, Sarbanes-Oxley and other business regulations affect IT operations directly, not just IT's support of compliance. IT professionals who manage business critical internal systems and systems supporting compliance will need to be prepared to pass the scrutiny of auditors. Succeeding requires formulating new corporate policies, implementing processes, and building IT systems to support those policies and processes.

For CIOs, the challenge will be to bring auditability to IT operations as many organizations today lag in these areas. The processes described in the CDP can greatly assist CIOs and other IT professionals in achieving auditable policies and processes that comply with the Sarbanes-Oxley financial reporting requirements.

Drivers affecting IT

- Improving confidence in the process with greater security of systems, stronger authentication and deeper audit trails.
- Establishing standards.
- Making documents and communications management part of the plan.
- Implementing a comprehensive internal audit control framework, and complete audit control simulations, by year-end 2003.

SourceForge Enterprise Edition Provides a Foundation for Compliance

IT organizations today are challenged with growing globally distributed development teams and compounded by the recent trend in outsourcing to offshore development partners, and Sarbanes-Oxley Act has dramatically increased the complexity of IT. The focus for executive management is now on successful and effective execution. The Sarbanes-Oxley Act of 2002 has dramatically affected standards for reporting. IT is primarily reponsible for a majority of the data that makes up the financial reports. Financial reports are generated by IT and its related processes; therefore, the effectiveness of IT process is imperative and must be verifiable.

SourceForge Enterprise Edition provides a Global Development Platform™ that can help IT organizations implement consistent and reliable process control for managing the application development lifecycle. The CDP provides recommended best practices for implementing these reliable and repeatable processes, and for ensuring that all application development data is fully archived and auditable.

SourceForge and the CDP can be implemented to provide the following:

- A secure common development platform for internal or offshore projects
- Integration of disparate tools and project data
- Management visibility and control into all development work
- Management for all stages of software development
- Automation of manual tasks
- Enforcement of process compliance
- Traceability and auditability of IT software applications

SourceForge does this by providing a foundation to manage and capture all application development project-related information. SourceForge captures changes in activities, progress and status of notifications, change logs, reporting, downloads, discussions, documents, issue tracking, version control, tasks, code commits, and project management activities. This information is archived and indexed providing complete traceability and audit trails of all entries within SourceForge. Additionally, collaborative communication is captured providing historical context and a clear record of who, what, where, how, when, and why changes in status of activities, documents or code has occurred.

SourceForge and Agile Development

VA Software began adopting agile development processes based on eXtreme Programming (XP) at the beginning of the Saturn development in mid-2002. Saturn is the fully re-architected J2EE-based SourceForge. Prior to Saturn, the VA product development team followed a more traditional "waterfall" methodology, with planned stages from requirements definition through sustaining engineering. While this model works well in many organizations and for many types of development projects, the VA team felt that given the scope of fully re-architecting and developing SourceForge (a multi-year project), they needed a methodology that was more agile and accomodated rapidly changing requirements. They selected the XP methodology, and modified it where needed to meet the specific needs of the team and the project.

The CDP is structured around a traditional “waterfall” methodology, many parts of it can be adapted to an agile methodology. This document describes the agile development processes followed by the VA product development team, including engineering, product management, user interface design, QA, and documentation.

Prerequisites

This document assumes that the reader has a basic knowledge of XP practices, such as the 12 principles of XP and agile development concepts such as iterations, user stories, velocity, etc. It is beyond the scope of this document to provide more than cursory definitions. Readers who are not familiar with XP are encouraged to research its principles.

Processes

1. **User Stories**: The first step in the VA XP development process is the drafting of user stories by product management (the XP customer.) User stories are requirements written from the standpoint of archetypal users. They are written in the form of stories, or tasks that the user wants to accomplish, instead of specific requirements. For example, instead of a specific requirement that SourceForge have a user-based task list on the main Task Manager page, a user story might state that Cameron needs to update his task list and wants to be able to enter multiple status changes from a single interface. This leaves the development team free to implement the functionality in a way that solves the user problem in the most effective manner. End user-requested product enhancements are also submitted in the form of user stories.

   Uses of SourceForge: Product management drafts one or more user stories documents and posts them to the Document Manager for reference during working review meetings. After all discussions are complete and input from team members has been incorporated, a tracker artifact is created for each user story, and associations are created back to the original user stories document.

2. **Release Planning**: After the candidate list of user stories is completed, the product development team meets to go over each user story and estimate the amount of time it will take to implement it. These are high-level estimates for purposes of release planning; more detailed estimates will be done later. The goal is to establish a more realistic estimate of what functionality can be implemented in the time allocated for the release. Based on these estimates, the product development team establishes a release plan.

   Uses of SourceForge: The high-level estimates document is stored in the Document Manager.

3. **Iteration Planning**: A key concept of agile development is iterative development, the goal of which is to provide a steady and predictable development process while accommodating rapidly changing requirements. The VA product development team generally follows two-week iteration cycles. At the start of each iteration, the team meets again to go over each candidate user story in greater detail, clarify any outstanding questions or issues, provide more precise implementation effort estimates, and calculate velocity (the sum of the effort allocated to all accepted user stories.) These effort estimates are provided in points per pair of developers. (Pair programming is discussed in more detail later.) At the end of the iteration planning meeting, the number of points is added up and the list of user stories to be implemented in that iteration is determined. This list is determined based on the velocity from the previous iteration.

   Uses of SourceForge: All candidate user stories were entered into the release tracker earlier in the planning stage. The release tracker is configured to include XP-specific custom fields such as "Iteration", "Points", "Acceptance Date/s", "HTTP Unit Test Name". After points are assigned and a user story is scheduled for a specific iteration, values are entered for these fields.

4. **Pair Programming**: Another key concept of agile development is pair programming. Pair programming increases the quality and consistency of the code being written, and ensures that all members of the team are able to work on any module of the application. The VA product development team rotates pairs frequently, and each pair works on a different user story each day. In many cases, one pair may start work on a user story and another pair will finish it. A common problem is the formation of pairs on a day-to-day basis. VA Software has developed a module using the SourceForge SOAP APIs that randomly creates new pairs. This "Partnerator" module is available for download from the SourceForge Tapestry site at *http://tapestry.sourceforge.vasoftware.com/sf/*

   Uses of SourceForge: Because the pairs are rotated frequently, user story tracker artifacts are not assigned to individual users or pairs. When a pair begins work on a user story, they change the status of the artifact to "In Development". The number of artifacts that are "In Development" should equal the number of pairs at any given time. When completed, they change the status of the artifact to "Awaiting Acceptance".

"

5. **Automated Build and Test Process**: Throughout the development cycle, all build and test activities are automated and run continuously. This ensures that a stable, master build is always available, and reduces the amount of QA time needed at the end of the cycle. For each user story, QA and engineering write an automated acceptance test. VA Software uses HttpUnit and JwebUnit for these tests. QA provides the test cases, and engineering writes the script and checks it into CVS. Test cases are also attached to each user story artifact. Whenever code is checked in, a first-stage build is triggered automatically and the unit tests are run. If the build or unit test fails, a bug is automatically created in SourceForge. If it passes, the code is promoted to the next second-stage build, which is built every 6 hours on all supported platforms (operating system, database, and application server.) It is then automatically uploaded to the SourceForge File Release System. Additional technical detail about VA's automated build and test processes is included as an appendix to this document.

   Uses of SourceForge: SourceForge is used extensively during the development, build, and test process. All code is checked into the SourceForge CVS repository. When a code check-in completes a user story, the developer creates an association to the user story tracker artifact for traceability. Based on the results of the automated build and test process, one of the following happens:

   - The build fails and a bug is created in the SourceForge tracker. The log file from the build is included as an attachment. The tracker is configured with custom fields for the characteristics of each build, such as operating system, database, etc. This enables reporting on the number of times specific configurations pass vs. fail. The entire development team monitors this tracker. All log files from each build (pass or fail) are also published to the Document Manager automatically.
   - The build passes and the code is included in the next second-stage build for automatic upload to the SourceForge File Release System. This provides constant availability of the latest, stable build for groups such as QA, documentation, and product marketing.

6. **Acceptance Testing**: In addition to the automatic unit testing, VA also holds daily acceptance meetings throughout the iteration to ensure that the functionality implemented matches that specified in the user story. These meetings are also used to verify that the user interface was implemented correctly and that the user documentation was completed correctly. During each meeting, the team uses the latest master build to review each user story with the "Awaiting Acceptance" status. The team checks that the functionality works as described in the user story, tests interactions with other functions, confirms that the user interface is correct, and looks for any identifiable bugs. Any issues are entered in the artifact's Comments field. (Documentation is generally reviewed during the subsequent iteration to allow time for completion.)

Uses of SourceForge: If a user story is accepted, the artifact status is changed to "Accepted" and the acceptance date is entered in the "Acceptance Date/s" field. If it is not accepted, the artifact status is changed back to "In Development" for futher work by engineering. It is then reviewed again in a later acceptance meeting.

7. **Manual QA Testing**: Because of the extensive, automated build and test processes, the amount of manual testing that the QA team needs to do is greatly minimized. At the end of the development cycle, prior the final release, QA tests the entire application from a holistic standpoint, focusing on exploratory testing and complex combinations of actions that aren't specific to user stories. QA also tests installation and all permutations of reference platforms.

   Uses of SourceForge: The SourceForge Tracker is used extensively to track bugs found during the QA period. Reporting is also used to report on the numbers of open vs. closed bugs. When a final build is certified by QA as ready for general release, the Quality Statement, along with other release-related documentation, is published to the Document Manager.

Benefits

Working with SourceForge and an agile development methodology has many benefits, both to your development organization and to your customers.

To the development team:

- SourceForge is very process-agnostic, so it is easily modified to support whatever variant of agile development processes your organization chooses to follow.
- The SourceForge APIs allow you to automate and integrate your build and test systems, as the VA product development team has done.
- The continuous build and test model minimizes manual QA effort and results in increased product quality.

To your customers:

- Managing user stories in the Tracker allows internal customers such as sales and field staff to track the status of their customers' requests.
- The continuous build and test process, along with acceptance testing of each user story, ensures that a high-quality, stable build is always available. At the end of each iteration, you have a build that you could publish to a public SourceForge project for access by partners, internal, or onsite customers. (An onsite customer is an XP concept of a customer or proxy that is always available for input and decision making. Having an onsite customer helps minimize the need for extensive, up-front design documentation.)
- The browser-based SourceForge interface supports high visibility and public accessibility.

CHAPTER 1

SEI Software Capability Maturity Model

This chapter includes the following information:

- Introduction to the Software Engineering Institute Capability Maturity Model for Software
- The Capability Maturity Model for Software and the SourceForge Collaborative Development Process
- Preparing for a Capability Maturity Model-Based Appraisal using SourceForge
- Overview of Key Process Areas for the Capability Maturity Model for Software Levels 2 and 3
- Mapping Level 2 and Level 3 Key Process Areas and Goals to the SourceForge Collaborative Development Process and SourceForge Enterprise Edition
- The CDP and the Capability Maturity Model Integration (CMMI)

Introduction

The Software Engineering Institute (SEI) Capability Maturity Model for Software (SW-CMM) is a framework that describes the key elements of an effective software process. Encompassing an evolutionary improvement path from ad-hoc, immature processes to mature, disciplined processes, the SW-CMM defines five process maturity levels ranging from Level 1 to Level 5. Each maturity level is comprised of Key Process Areas (KPAs), that define a series of activities that when performed consistently and effectively, help organizations meet established goals for process capability at a defined maturity level.

The key premise of the SW-CMM is that process maturity equates to the improved ability of an organization to meet cost, schedule, functionality, and product quality goals. The SW-CMM, and other CMM models such as the Systems Engineering CMM, Software Acquisition CMM, and the CMM Integration℠ (CMMI®) model, have been widely adopted in both the government and commercial sectors, and are frequently used to assess the maturity of an organization with whom a subcontracting or other relationship is being considered. The CMM models are also widely used as the basis for developing internal process improvement action plans for organizations that simply desire a more predictable outcome to their development and other projects.

SW-CMM and the SourceForge Collaborative Development Process

While the SW-CMM framework provides a series of goals associated with each maturity level, it does not provide a methodology for achieving them. Methodologies for achieving SW-CMM maturity levels are often developed in coordination with an authorized SEI Lead Assessor or other software process improvement consultant, and in general, an average of 18 to 36 months is needed to progress from a maturity Level 1 to a maturity Level 2 rating. An additional 18 to 36 months is required to progress from a maturity Level 2 to a maturity Level 3 rating. In the interim, an initial diagnostic assessment, one or more interim assessments to measure progress, and considerable consultation time are generally required before a formal CMM-Based Appraisal for Internal Process Improvement (CBA-IPI) can be successfully completed.

The SourceForge Collaborative Development Process (CDP) provides a methodology that, when combined with the use of SourceForge, fully supports all of the goals associated with achieving SW-CMM maturity Levels 2 and 3. Following the processes and guidelines detailed in the CDP can considerably reduce the amount of time, internal resources, and dependency on consultants generally associated with achieving a maturity Level 2 or 3 rating.

Each stage of the CDP defines one or more processes that equate to goals associated with one or more SW-CMM Level 2 and Level 3 KPAs. This chapter provides a comprehensive overview on using the CDP and SourceForge to achieve SW-CMM Level 2 or 3. *SourceForge and the Capability Maturity Model for Software, A Supplement to the SourceForge Collaborative Development Process* provides detailed mappings for using SourceForge and the CDP to fulfill each key practice (commitment, ability, activity, measurement, and verification.) Throughout the CDP materials, specific references to this chapter or the Supplement are provided to allow users to quickly and easily associate the CDP processes with the SW-CMM goals and KPAs they enable. Checklists are also provided at the end of each phase to confirm that all key SW-CMM requirements have been met.

It is important to note that the SW-CMM references are not a required element of the CDP, nor is the CDP based exclusively on the SW-CMM. The references are provided for the benefit of users preparing for a formal CMM-based appraisal, and for those simply interested in using its framework as a basis for or supplement to an internal process improvement initiative.

Preparing for a CMM-Based Appraisal using SourceForge

To formally measure an organization's progress toward SW-CMM maturity levels, a CMM-Based Appraisal for Internal Process Improvement (CBA IPI) is conducted. In order for the results to be recognized by the Software Engineering Institute, a CBA IPI must be conducted by an authorized SEI Lead Assessor, who will work with a team of internal participants to collaboratively appraise and characterize the organization's software processes.

The bulk of the materials reviewed during the CBA IPI fall into two categories: procedural evidence and documentary evidence. Procedural evidence consists of the set of procedures and processes an organization has developed that, when followed, provide the foundation for compliance with SW-CMM KPAs. Documentary evidence consists of data from one or more completed projects, providing evidence that the procedures and processes have been successfully implemented and that the organization is currently operating according to the maturity standards defined by the target CMM level.

Using SourceForge as recommended throughout the CDP can provide significant value when preparing for and conducting a CBA IPI or other CMM-based appraisal. Consistently using SourceForge in the following ways will help organize and consolidate the required data, simplify the process of gathering the materials required for the appraisal, and decrease the time and effort required on the part of both the assessor and the internal team.

Appraisal-enabling Uses for SourceForge

- Create a unique SourceForge project to manage your process improvement project.
  - In order to be successful, a process improvement initiative must be managed as a project, not as an afterthought or side activity.
- Establish a Software Process Database in the Document Manager to store all process documents, procedures, templates, and other process-related documentation.
  - This will provide an easily accessible central process repository for your internal audience as well as the assessment team, and will facilitate the collection and review of procedural evidence.
  - The Software Process Database also provides a home project for the Software Engineering Process Group.
- Organize the Document Manager for each of your development projects in a consistent, logical hierarchy. Store all in-progress and completed documents, templates, and other process-related documentation in each project's Document Manager.
  - Organizing the Document Manager hierarchies for each project in a consistent manner will enable both your internal audience and the assessment team to easily locate needed documentation, and will facilitate the collection and review of documentary evidence.

- Manage all product requirements using the SourceForge Tracker.
  - This will provide the traceability of product requirements required by the Requirements Management, Software Project Planning, and other SW-CMM KPAs.
- Create relevant associations among project artifacts to establish dependencies and provide additional traceability.
  - Clear visibility into project dependencies is critical to fulfilling SW-CMM KPAs. SourceForge provides the ability to display dependencies among artifacts, and will facilitate the collection and review of documentary evidence related to the Software Project Tracking and Oversight, Software Product Engineering, and other SW-CMM KPAs.
- Manage all project plans using the SourceForge Task Manager.
  - The Task Manager will provide documentary evidence associated with tracking actual against planned progress required by the Software Project Tracking and Oversight and other SW-CMM KPAs.
  - The Task Manager will facilitate intergroup coordination by providing high visibility into the status of all projects and will provide the reporting to management required by many SW-CMM KPAs.
- Establish an Organizational Training project to manage all training-related activities and documentation.
  - This will provide a searchable repository of all training records and provide visibility into training processes and activities.
- Discuss ideas and proposals for continuous process improvement using SourceForge Discussion Forums and Mailing Lists.

Overview of the KPAs for SW-CMM Level 2 and Level 3

As the diagram below illustrates, the CMM defines five maturity levels ranging from Level 1 to Level 5 with a corresponding descriptor for each level.

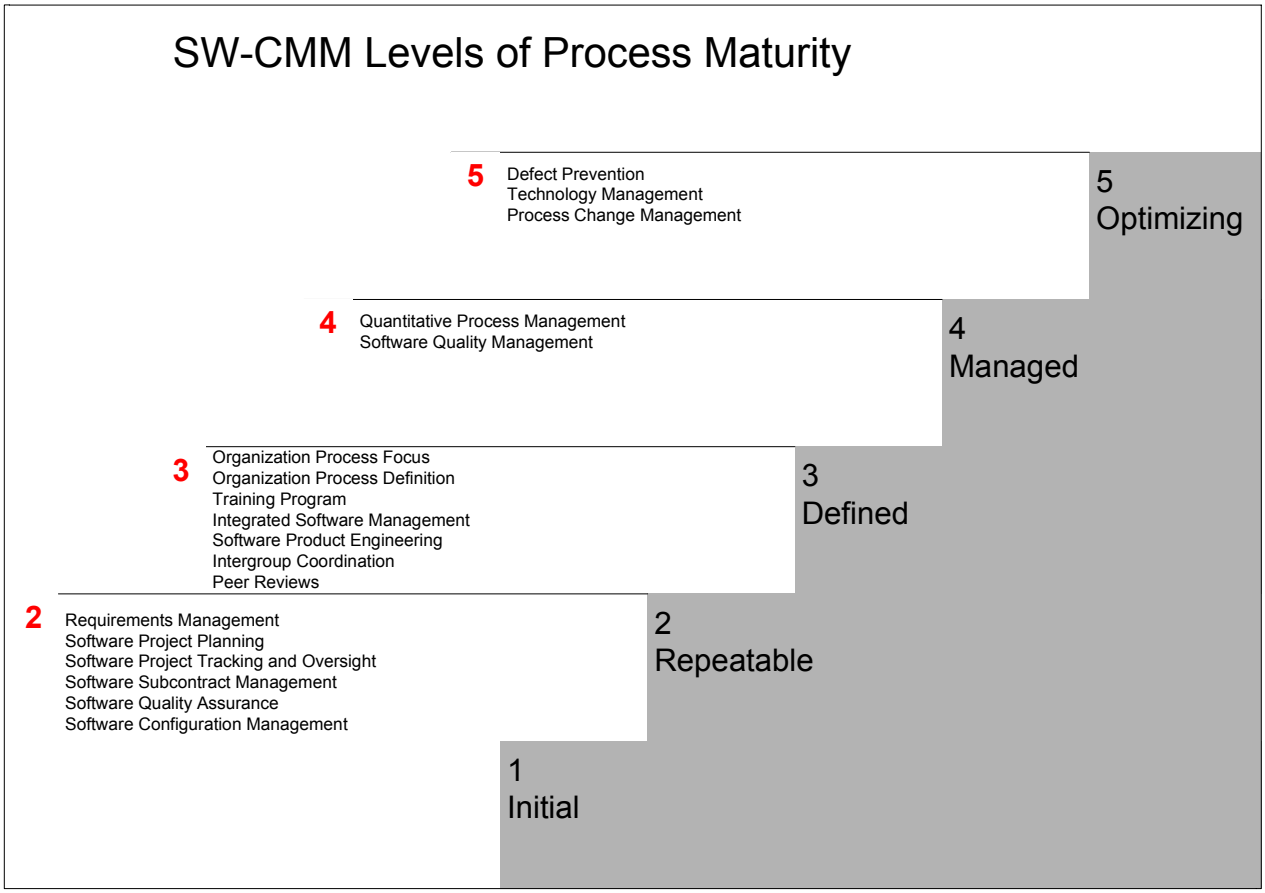


Figure 5. SW-CMM Levels of Process Maturity

The CDP, when followed using SourceForge where recommended, fully supports all of the goals associated with achieving a SW-CMM maturity Level 2 or 3. The remainder of this chapter will describe the specific requirements of SW-CMM Levels 2 and 3, and will illustrate how to use the CDP and SourceForge to achieve them.

Mapping Level 2 and Level 3 KPAs and Goals to the CDP and SourceForge

The following tables provide a description of all SW-CMM Level 2 and Level 3 KPAs , the goals associated with each, and the recommended practices for using the CDP and SourceForge to achieve them. The mappings in this section provide summary information on the goals of each KPA. For detailed mappings covering each commitment, ability, activity, measurement, verification associated with the Level 2 and 3 KPAs, please see *SourceForge and the Capability Maturity Model for Software, A Supplement to the SourceForge Collaborative Development Process*.

For additional detail on the SW-CMM and other Software Engineering Institute publications, see the Software Engineering Institute website at:
http://www.sei.cmu.edu

Level 2 KPAs and Goals

Table 1. Level 2 Key Process Area: Requirements Management

Goal	Mapping to the CDP and SourceForge
1. System requirements allocated to software are controlled to establish a baseline for software engineering and management use.	Manage feature requests and other product requirements using the SourceForge Tracker to establish a baseline and ensure full traceability of requirements. Include export of product requirements Tracker artifacts in Product Requirements Documents (Product Requirements Documents.) Manage document review process using the SourceForge Document Manager.
2. Software plans, products and activities are kept consistent with the system requirements allocated to the software	Create all technical and other design documents based on the Product Requirements Documents. Store all documents in the Document Manager. Manage all tasks and deliverables associated with a development project using the Task Manager for full visibility and control of progress, slippages, deviations. Store Change Control and other process and policy documents in the Document Manager. Map code commits to requirements Tracker artifacts using associations.

Table 2. Level 2 Key Process Area: Software Project Planning

Goals	Mapping to the CDP and SourceForge
1. Software estimates are documented for use in planning and tracking the software project	Complete Size, Effort, and Cost Estimation Worksheets. Use the resulting estimates to plan the project. Track estimated versus actual results throughout the project using the Tracker.
2. Software project activities and commitments are planned and documented	Create all Design phase planning documents using the Document Manager Iterative Document Review and Approval Process. Store all technical design and other planning documents in the Document Manager. Manage all tasks and deliverables associated with a development Project Plan using the Task Manager for full visibility and control of progress, slippages, deviations.
3. Affected groups and individuals agree to their commitments related to the software project	Record approvals of planning documents using the Document Manager Iterative Document Review and Approval Process. Create a Launch Team and use Discussion Forums and Mailing Lists to automatically archive all communications. Store all meeting minutes in the Document Manager.

Table 3. Level 2 Key Process Area: Software Project Tracking and Oversight

Goals	Mapping to the CDP and SourceForge
1. Actual results and performances are tracked against the software plans	Store baseline requirements and design documents in the Document Manager. Maintain a record of all changes made to dates associated with specific tasks using the Task Manager. Track all issues using an appropriate bugs or other issues Tracker. Understand the status of all outstanding feature requests, bugs, and other tracked items using Tracker reporting. Create a Project Logistics Tracker to track size, effort and cost estimates, risks, and critical computer resources.
2. Corrective actions are taken and managed to closure when actual results and performance deviate significantly from the software plans	Track all issues using an appropriate bugs or other issues Tracker. Ensure a high degree of management visibility and control into progress, slippages, and other deviations from the project plan using the Task Manager. Initiate corrective action based on activity in the Project Logistics Tracker.
3. Changes to software commitments are agreed to by the affected groups and individuals	Track all feature requests, bugs, and change requests using the Tracker. Manage changes to requirements in accordance with the Change Control Policy. Control ability to commit code after change control has been instituted using your SCM tool accessed through SourceForge.

Table 4. Level 2 Key Process Area: Software Subcontract Management

Goals	Mapping to the CDP and SourceForge
1. The prime contractor selects qualified software subcontractors	Manage Request for Proposals, Subcontractor Selection Worksheets,and other documents related to subcontractor selection using the Document Manager. Ensure agreement on subcontractor selection criteria using the Iterative Document Review and Approval Process.
2. The prime contractor and the software subcontractor agree to their commitments to each other	Store final Statement of Work in the Document Manager for future reference by both parties. Ensure that the subcontractor agrees to use SourceForge for communication, referencing process templates and guidelines, and storage of work products such as documents and source code.
3. The prime contractor and the software subcontractor maintain ongoing communications	Maintain a record of communications by using Discussion Forums and Mailing Lists. Store all documentation produced by, or referenced by, the subcontractor in the Document Manager. Manage all actions assigned to the subcontractor by using the Tracker and Task Manager. Expand all of the practices detailed in the other KPAs to a distributed development organization. Include subcontractors in regular status meetings.
4. The prime contractor tracks the software subcontractor's actual results and performance against its commitments	Manage all actions assigned to the subcontractor, and ensure a high degree of management visibility and control into progress, slippages, and other deviations from the project plan using the Task Manager. Track subcontractor’s progress against plan using the Task Manager and other reporting tools. Track all issues using an appropriate bugs or other issues Tracker. Understand the status of all outstanding feature requests, bugs, and other tracked items using Tracker reporting. Conduct period meetings to review the subcontractor’s progress against agreed-upon milestones.

Table 5. Level 2 Key Process Area: Software Quality Assurance

Goals	Mapping to the CDP and SourceForge
1. Software Quality Assurance activities are planned	Create a Software Quality Assurance plan for the project as part of the Product Plan. Identify the group or individuals who will conduct SQA activities in the Product Plan. Update the SQA audit checklists to reflect the items selected for audit.
2. Adherence of software products and activities to the applicable standards, procedures and requirements is verified objectively	Use the SQA audit checklists to ensure that the products and activities being audited adhere to all applicable standards, procedures, and requirements. Ensure that the SQA team has an independent reporting channel to senior management.
3. Affected groups and individuals are informed of Software Quality Assurance activities and results	Report any issues to the artifact owner and to management according to the procedure defined in the Product Plan. Report results to management according to the procedure defined in the Product Plan. Post the completed SQA audit checklists to the Document Manager and update the status of SQA tasks in the Task Manager.
4. Noncompliance issues that cannot be resolved within the software project are addressed by senior management	Manage the tasks associated with the SQA effort and ensure a high degree of management visibility and control into progress, slippages, and other deviations from the project plan using the Task Manager. Report results to senior management according to the procedure defined in the Product Plan.

Table 6. Level 2 Key Process Area: Software Configuration Management

Goals	Mapping to the CDP and SourceForge
1. Software configuration management activities are planned	Store Software Configuration Management and other process and policy documents in the Document Manager. Create associations between code commits and bugs or requirements tracker artifacts.
2. Selected software work products are identified, controlled and available	Access your chosen SCM system (CVS, Rational ClearCase, Merant PVCS) through the SCM integration interface. Store Software Configuration Management and other process and policy documents in the Document Manager. Use Tracker / Code Associations to require associations between code commits and bugs or requirements tracker artifacts.
3. Changes to identified software work products are controlled	Associate code commits with Tracker artifacts such as bugs and feature requests using Tracker / Code Associations Control ability to commit code after change control has been instituted using your SCM tool accessed through SourceForge. Monitor code commits and other SCM-related activities using Reporting and Statistics. Control changes to requirements and other documents using the Document Change Control Policy.
4. Affected groups and individuals are informed of the status and content of software baselines	Communicate the status of software configuration management tasks using the Task Manager.

Level 3 KPAs and Goals

Table 7. Level 3 Key Process Area: Organization Process Focus

Goals	Mapping to the CDP and SourceForge
1. Software process development and improvement activities are coordinated across the organization.	Form a Software Engineering Process Group (SEPG) with representatives from all key functional areas within the organization. Select members from each Launch Team to provide current insight into the process-related activities of each project team. Set up a Software Process Database project in SourceForge to store all process templates and other materials, and to act as the home project for the SEPG. Communicate with key stakeholders using News, Discussion Forums, and Mailing Lists. Provide visibility into the status of the organization's process-related activities using the Task Manager. Follow the Iterative Document Review and Approval Process when creating or revising process-related documents.
2. The strengths and weaknesses of the software processes used are identified relative to a process standard	The SEPG should meet regularly to review the strengths and weaknesses of the processes being used against an established standard such as SW-CMM. Reference the templates, historical data, and other materials stored in the SEPG project in SourceForge for comparison against the standard. Conduct post-release analysis activities to determine the strengths and weaknesses of processes followed during the previous project.
3. Organization-level process development and improvement activities are planned.	The SEPG should plan process development and improvement activities based on the results of their ongoing analysis against a process standard and against achieved results. Reference the templates, historical data, and other materials stored in the SEPG project in SourceForge. Communicate with key stakeholders using News, Discussion Forums, and Mailing Lists. Provide visibility into the status of the organization's process-related activities using the Task Manager. Follow the Iterative Document Review and Approval Process when creating or revising process-related documents.

Table 8. Level 3 Key Process Area: Organization Process Definition

Goals	Mapping to the CDP and SourceForge
1. A standard software process for the organization is developed and maintained.	The SEPG should establish a standard set of software processes for the organization based on the results of their ongoing analysis against a process standard and against achieved results. Store all templates, historical data, and other materials in the Software Process Database project in SourceForge and configure Role-based Access Control to make them accessible throughout the organization. Communicate with the organization using News, Discussion Forums, and Mailing Lists.
2. Information related to the use of the organization's standard software process by the software projects is collected, reviewed, and made available.	Follow the Iterative Document Review and Approval Process when creating or revising process-related documents. Ensure that representatives from across the organization are involved in the review process. Store all templates, historical data, and other materials in the Software Process Database project in SourceForge and configure Role-based Access Control to make them accessible throughout the organization.

Table 9. Level 3 Key Process Area: Training Program

Goals	Mapping to the CDP and SourceForge
1. Training activities are planned.	Define the training requirements for each project in the Product Plan. Identify or establish a Training Group to coordinate organizational training activities. Follow the Iterative Document Review and Approval Process when creating the Product Plan to ensure that all groups requiring training are represented. Complete a Training Worksheet for each training activity to provide supplementary information on training requirements to the Training Group.
2. Training for developing the skills and knowledge needed to perform software management and technical roles is provided.	Execute on the training activities identified in the Product Plan and the Training Worksheets. Conduct follow-up activities to ensure attendee comprehension and retention. Require attendees to complete Training Evaluation Forms following each training activity. Set up a Discussion Forum for discussion of training needs and issues. Store standard training courses, Training Evaluation Forms, and other training documentation in the Organizational Training Project for easy reference.
3. Individuals in the software engineering group and software-related groups receive the training necessary to perform their roles.	Execute on the training activities identified in the Product Plan and the Training Worksheets. Conduct follow-up activities to ensure attendee comprehension and retention. Require attendees to complete Training Evaluation Forms following each training activity. Set up a Discussion Forum for discussion of training needs and issues. Analyze the successes and challenges of any training activities during each project's post-release analysis. Modify training-related processes as needed.

Table 10. Level 3 Key Process Area: Integrated Software Management

Goals	Mapping to the CDP and SourceForge
1. The project's defined software process is a tailored version of the organization's standard software process.	Store all standard process templates, historical data, and other materials in the Software Process Database project in SourceForge and make them publicly accessible throughout the organization. Tailor the standard process set for each individual project in the Product Plan.
2. The project is planned and managed according to the project's defined software process.	Execute all software development, testing, quality assurance, peer review, configuration management, and other activities according to their respective plans developed in the Product Definition and Design phases. Manage all activities and deliverables using the Task Manager to provide visibility into the status all project activities.

Table 11. Level 3 Key Process Area: Software Product Engineering

Goals	Mapping to the CDP and SourceForge
1. The software engineering tasks are defined, integrated, and consistently performed to produce the software.	Store SCM policy documents, coding guidelines, and related policy documents in the Document Manager. Require that code commits be associated with bugs and feature request tracker artifacts using Tracker / Code Associations. Control the ability to commit code after change control has been instituted using your SCM tool accessed through SourceForge. Conduct frequent peer reviews of code to ensure high quality and consistency. Use a filter that scans code to ensure its compliance to standards before allowing check in. Conduct software testing activities as defined in the Software Test Plan to ensure the high quality of all code.
2. Software work products are kept consistent with each other.	Conduct frequent peer reviews of code to ensure high quality and consistency. Use a filter that scans code to ensure its compliance to standards before allowing check in. Manage the project plan in the Task Manager to provide high visibility into the status of the entire project.

Table 12. Level 3 Key Process Area: Intergroup Coordination

Goals	Mapping to the CDP and SourceForge
1. The customer's requirements are agreed to by all affected groups.	Require all impacted functional groups to review and approve the Product Plan and Product Requirements Documents. Follow the Iterative Document Review and Approval process for all planning and other requirements-related documentation to ensure that all stakeholders are in agreement regarding product requirements. Conduct cross-functional peer review meetings if needed. Manage all product requirements, feature requests and bugs using the Tracker.
2. The commitments between the engineering groups are agreed to by the affected groups.	Follow the Iterative Document Review and Approval process for the Product Plan, Product Requirements Documents, and draft and final Project Plans to ensure that all stakeholders are in agreement regarding project commitments. Manage all activities and deliverables using the Task Manager for real-time status on all tasks assigned to each group.
3. The engineering groups identify, track, and resolve integroup issues.	Identify problem areas using the status features of the Task Manager. Intergroup issues that cannot be more easily resolved should be addressed through regular engineering meetings or by the cross-functional Launch Team.

Table 13. Level 3 Key Process Area: Peer Reviews

Goals	Mapping to the CDP and SourceForge
1. Peer review activities are planned.	Include peer review as part of the organization's standard processes. Identify which artifacts will be subject to which type of peer review in the Product Plan. Conduct the Iterative Document Review and Approval Process on all project documents. Conduct cross-functional peer review meetings for critical or complex project documents, especially those with a high degree of cross-functional impact.. Conduct frequent peer review of product code during Technical Unit Testing and other phases. Use the Code Review Form to analyze and rate other developers' code.
2. Defects in the software work products are identified and remove.	Conduct frequent peer review of product code during Technical Unit Testing and other phases. Use the Code Review Form to analyze and rate other developers' code. Peer review and the Code Review Form provide a vehicle for identifying and removing product defects. Correct any defects identified in documents during the Iterative Document Review and Approval Process.

The CDP and the Capability Maturity Model Integration (CMMI)

CMMI Overview

Since 1991, the Software Engineering Institute has developed Capability Maturity Models (CMMs) for a myriad of disciplines. These include the CMM for Software (SW-CMM,) the Systems Engineering CMM (SE-CMM,) the Software Acquisition CMM (SA-CMM,) the Integrated Product and Process Development CMM (IPD-CMM,) and the People CMM for workforce management (P-CMM.)

Although these models have proven extremely useful to many organizations, and continue to do so, the use of multiple models has been problematic for organizations with multiple disciplines represented by separate models. Many organizations would like to focus their improvement efforts across the disciplines within their organizations. However, the differences among these discipline-specific models, including their architecture, content, and approach, have limited these organizations' ability to focus their improvements successfully. Further, applying multiple models that are not integrated within and across an organization becomes more costly in terms of training, appraisals, and improvement activities.

To address these issues, the SEI has developed the Capability Maturity Model Integration (CMMI,) a set of integrated models that successfully addresses multiple disciplines and provides integrated training and appraisal support. This model builds on and extends the best practices of the Capability Maturity Model for Software (SW-CMM,) the Systems Engineering Capability Model (SE-CMM,) and the Integrated Product Development Capability Maturity Model (IPD-CMM.)

CDP and CMMI

The SEI is currently advocating the adoption of the CMMI process improvement model for product and service development and maintenance. Updates and training for the SW-CMM were discontinued by the SEI in December 2003. However, data from SEI-authorized assessments against the SW-CMM will continue to be accepted and used by the SEI for an indefinite period. It is not necessary for organizations currently pursuing SW-CMM goals to transition to CMMI unless they feel they would benefit from the additional process areas incorporated from the other disciplines.

For organizations who wish to transition to CMMI, the SEI provides a significant amount of material designed to facilitate this transition. However, the CMMI is a relatively new model and is still considered an early adopters' technology. The majority of organizations pursuing CMM goals are continuing to follow their existing SW-CMM based process improvement plans until the CMMI becomes more mature and more significant data on the benefits of migration are available. This manual provides guidelines on using SourceForge and the CDP

to meet SW-CMM Level 2 and 3 goals. As the CMMI becomes more mature and SourceForge users demonstrate a desire to transition to the integrated model, the CDP will be updated with detailed CMMI to SourceForge and CDP mappings. Detailed material on transitioning your SourceForge-supported CMM processes to SourceForge-supported CMMI processes will also be provided.

If you are currently pursuing or interested in CMMI as a process improvement model, please contact your VA Software representative. The VA Software Professional Services organization, in partnership with an SEI-authorized Standard CMMI Appraisal Method for Process Improvement (SCAMPI^{SM}) Lead Appraiser, can work with you to develop a customized SourceForge-based process improvement plan.

CHAPTER 2
Cross-Lifecycle Principles

This chapter includes the following information:

- Preparatory Activities
- Cross-Lifecycle Principles
- Software Engineering Process Group roles and responsibilities
- Setting up and managing your SourceForge Project
- Setting up and managing your Software Process Database
- Peer Review Processes
- Software Quality Assurance Audits
- Managing your Product Roadmap
- Training Programs
- Templates and Organizational Policies

Preparatory Activities

Before you begin any software development project, there is a minimum set of activities that should be conducted to ensure that your organization is prepared to manage the project effectively.

If you or your team is new to SourceForge, the first essential item is to ensure that all staff who will be using it receive a sufficient level of user training. All staff should become comfortable using the SourceForge tools applicable to their daily activities. The *SourceForge Enterprise Edition 4.2 User Guide* provides an excellent reference for both new and experienced SourceForge users. The VA Software Professional Services organization also offers comprehensive, fully customizable training programs to help new users quickly achieve proficiency.

New CDP Projects

If this is your first project in which you will follow the CDP guidelines, it is important to review the entire set of CDP materials, including this manual, the SourceForge Best Practices, and the accompanying set of document templates and checklists before beginning your project. While the majority of the processes, activities, and deliverables with which you will be working are described in detail during the appropriate CDP phase, having a good understanding of the entire lifecycle before you begin will greatly assist you when planning your project, training your staff, securing any necessary management sponsorship, and other preparatory work.

Pay particular attention to the Cross-Lifecycle Principles section beginning on page 50, as it describes concepts and activities that are applicable throughout all stages of the CDP, such as the creation of a Software Engineering Process Group, the establishment of a software process database, methods of managing action items, and formalizing your peer review and software quality assurance functions. If you are using the SW-CMM as a process improvement framework, review *SourceForge and the Capability Maturity Model for Software, A Supplement to the SourceForge Collaborative Development Process* to familiarize yourself with the ways in which SourceForge and the CDP are used to support the goals associated with the SW-CMM.

Comprehensive and early review of the CDP materials will also help you identify any areas that you may wish to customize to better meet the specific needs of your organization. The VA Software Professional Services organization is available to help you identify, develop, deploy, and train your staff on any process-related customizations.

SW-CMM NOTE

The SW-CMM requires that all staff receive training in performing their assigned activities. A training program based on the CDP and completed by staff responsible for process-related activities will satisfy many of the training requirements specified in Levels 2 and 3 of the SW-CMM. The VA Software Professional Services organization offers such a training program, also customizable to meet the specific needs of your organization.

See *SourceForge and the Capability Maturity Model for Software, A Supplement to the SourceForge Collaborative Development Process* for specific SW-CMM requirements satisfied by a process activities training program.

Subsequent CDP Projects

If you have followed the CDP guidelines in a previous project, and are preparing to begin a new one, now is the time to review your organization's process database for any updates that may have been made since completing your last project. It is also a good time to check in with your Software Engineering Process Group or whoever is responsible for managing your organization's process improvement activities. The Software Engineering Process Group should be able to answer any process-related questions, record and address any feedback you may have regarding the successes and challenges of your previous project, provide training for your team on any new processes or other CDP elements, and ensure your activities are coordinated with those of the rest of the organization. The Software Engineering Process Group should also be able to provide recommended process changes based on their analysis of prior project experiences.

Cross-Lifecycle Principles

The majority of processes, activities, and deliverables discussed in the CDP are described in detail in the CDP phase in which they take place. The following processes, however, are used extensively throughout the CDP beginning in the earliest conceptual stages. It is therefore important to spend some time preparing the necessary participants, processes, and materials as early as possible. You may wish to establish the processes described in this section before beginning your first project, or you may overlap their development with your project's Product Definition phase. All of the processes in this section are required before exiting the Product Definition phase, and are used repeatedly throughout the phases of the CDP.

Software Engineering Process Group

A Software Engineering Process Group is a team made up of representatives from Software Engineering, Software Testing, Software Quality Assurance, Software Product Management, and similar functions. The goal of such a group is to actively manage the process improvement activities of an organization. While organizations pursuing goals associated with SW-CMM, CMMI, ISO 9000, and other formal process improvement frameworks are required to maintain such a group, many organizations pursuing more customized or informal process improvement goals often create similar teams dedicated to managing their process improvement activities.

Before you begin your first CDP project, a Software Engineering Process Group or similar team should be established. The specific responsibilities of this team will vary based on the process improvement goals established by your organization, as will the size of the team. In some organizations, a single person may be sufficient to oversee your process management activities. Others may require a larger group of individuals to ensure adequate representation of all key functional areas and to share responsibility for process-related activities. It is important to identify the individual or group who will be responsible for process improvement early in the planning stages of your project, as this individual or group will be expected to provide guidance to project members on process-related activities, beginning in the earliest stages of each project.

Some of the core functions of the Software Engineering Process Group may include:

- Identifying any standards to which the organization will adhere, such as SW-CMM, CMMI, ISO 9000, or those defined internally.
- Obtaining necessary management sponsorship to support the organization's process improvement initiatives.
- Establishing the process improvement plan for the organization.
- Establishing a method for communicating the status of key process-related activities within the organization.
- Establishing and maintaining a Software Process Database or similar method for providing access to the organization's process materials.
- Participating as a core a team member in each software development project.
- Acting as a consultant to ensure that the processes established by the organization are understood and followed.
- Periodically measuring the value of the organization's processes and making modifications where needed.
- Coordinating each project's process-improvement activities with those of the rest of the organization.

SourceForge Project

One of the first activities to complete prior to beginning your first CDP development project is to establish a SourceForge project to manage your development project's schedule, documents, action items, and other artifacts. Many of the SourceForge tools and artifacts such as Trackers, the Task Manager, and the SCM repository will not be used until later in the CDP. However, establishing a SourceForge project early in the project planning stage is useful for storing preliminary planning documents such as the Business Requirements Document, managing action items prior to completing the Project Plan, and facilitating communication between team members. Establishing the project in the early stages also provides continuity throughout the project, from its earliest conceptual stages through its conclusion.

If you determine during the Product Definition phase that the project will not be continued, the SourceForge project will still provide a useful repository of planning data that can be used when evaluating your next development project.

Before beginning the Product Definition phase of your development project, your SourceForge project should contain the following:

SourceForge Component	Contents
Project Description	Provide a description of your project and its objectives for the Project Home Page.
Project Members	Add members participating in the preliminary planning stages to your project.
Roles	Assign appropriate roles to members using Role-based Access Control.
Task Manager	Create a task folder for managing action items using the Task Manager. If desired, you may choose to create a Microsoft® Project file or input tasks directly into the Task Manager.
Document Manager	Establish a Document Manager folder hierarchy for use in managing document approval and storage. Please see "Managing the Document Lifecycle using the SourceForge Document Manager" on page 273 for best practices on setting up a folder hierarchy.

SourceForge Component	Contents
Software Process Database	Retrieve the appropriate CDP process templates from your Process Database and store them in your project's Document Manager. At this stage, only those process templates needed for the early planning stages of your project are required. (Business Requirements Document, Technical Constraints Document, and Product Plan.) Please see "Managing your Software Process Database using the SourceForge Document Manager" on page 247 for best practices on managing your process database.
Trackers	If feature requests, product defects, or other product-related artifacts will be used as input for building the project's business case, identify the Trackers in which these artifacts will be managed. These Trackers may have been established in a previous project. Please see "Managing Product Requirements Using the SourceForge Tracker" on page 280 for best practices on setting up and using Trackers.
Discussion Forums and Mailing Lists	Create one or more Discussion Forums and Mailing Lists for discussion of Product Definition topics.

Each chapter of this manual provides the recommended contents of your SourceForge project when entering and exiting each CDP phase. If you need help using any of the SourceForge tools or completing any of the activities specified, please reference the *SourceForge Enterprise Edition 4.2 User Guide* or contact your VA Software representative.

Software Process Database

One of the key responsibilities of the Software Engineering Process Group is to establish and maintain a Software Process Database. The purpose of this Database is to collect and maintain all of the organization's process-related materials in a centralized location. Maintaining a centralized database enables project members to quickly and easily locate the correct process templates, guidelines, and other materials needed to perform their job functions.

The contents of the Software Process Database may include:

- A library containing your organization's standard processes, guidelines, templates, checklists, and other standard materials. These standard materials can be those provided by the CDP, the CDP materials customized to meet your organization's specific needs, or other materials developed by your organization.
- Completed templates, checklists, and other materials from current and previous projects.
- Estimates of software size, effort, cost, and other metrics.
- Actual data on software size, effort, cost, and other metrics for use in estimating future projects.
- Completed peer reviews, software quality assurance reports, and results from post-release analysis activities.

The SourceForge Document Manager provides an excellent resource for establishing and maintaining your Software Process Database. A SourceForge Software Process Database utility is provided with the CDP to help you establish and customize this Database. Please see "Managing your Software Process Database using the SourceForge Document Manager" on page 247 for additional guidelines and recommended best practices.

Peer Review

Peer review is a key element throughout every phase of the CDP and is applied to nearly all documents, software code, and other artifacts. Establishing strong peer review processes has many benefits, including collecting input from stakeholders in a variety of functional areas, identifying defects in work products, and facilitating intergroup coordination. The quality of peer-reviewed work is generally higher and contributes to a more successful project than that which is not peer-reviewed.

Peer review can take several forms, depending on criteria such as the importance of the document, the number of desired reviewers, the time available for review, and the software and other tools available to support review processes. The SourceForge Document Manager enables a comprehensive document review process for managing a "review by distribution" method of peer review. Other methods include cross-functional meetings (often useful for critical documents with a broad organizational impact such as the Product Plan,) code review (used for reviewing software code for defects,) or "buddy reviews" (often useful when a limited amount of review time is available.)

Please see "Managing your Peer Review Processes using SourceForge" on page 249 for descriptions of these methods plus recommendations for using SourceForge to support Peer Review processes.

Peer Review Icon

Throughout the CDP manual, the Peer Review icon shown here will indicate which artifacts should undergo peer review. The recommended method of peer review will also be included.

Software Quality Assurance Audits

The Software Quality Assurance function as defined by the CDP refers to staff engaged in a variety of quality oversight activities. Leading process improvement initiatives and conducting periodic audits on the timely completion, quality, and processes followed to produce various deliverables are examples of activities often managed by Software Quality Assurance teams. Throughout the remainder of the CDP materials, Software Quality Assurance refers specifically to this periodic quality audit function. It does not refer to staff engaged in software testing activities, although the terminology used in different organizations may vary.

In addition to Peer Review, Software Quality Assurance audits are one of the most pervasive activities conducted throughout the phases of the CDP. In some organizations, for example large aerospace and defense contractors, a group dedicated to conducting quality audits and similar activities on a full-time basis exists. In others where a full-time Software Quality Assurance organization is not necessary or cost-effective, the audit function may be performed on an as-needed basis by staff normally assigned to other functions.

In either scenario, Software Quality Assurance audits are conducted beginning in the Product Definition phase, so it is important to ensure that the appropriate staff are identified and prepared to perform this function.

Please see “Managing Software Quality Assurance Audits using SourceForge” on page 257 for additional guidelines and recommended best practices for performing Software Quality Assurance audits.

Software Quality Assurance Icon

Throughout the CDP manual, the Software Quality Assurance icon shown here will indicate the artifacts for which quality audits are recommended. It should be noted, however, that Software Quality Assurance audits are conducted on a number of activities and processes that are not associated with a specific deliverable, such as effective ongoing management of subcontractors. Refer to the Product Plan template and the Software Quality Assurance audit checklists provided for each CDP phase for a comprehensive list of activities, deliverables, and processes for which audits are recommended.

Managing Your Product Roadmap

Capturing product requirements clearly, thoroughly, and accurately is a fundamental goal of any software development organization. The CDP recommends a continuous requirements management process, running in parallel with the product-specific requirements specification processes detailed in the Product Definition through Sustaining Engineering phases.

A continuous requirements management process enables staff in Business Strategy, Market Analysis, Product Management, or similar roles to continually research and analyze ways to improve the product offerings of the organization. They can then define and maintain a high-level product strategy for the organization that is treated independently from individual product-specific requirements. This high-level product strategy is refered to throughout the CDP as the Product Roadmap.

From the Product Roadmap, product-specific requirements documents are created during the Product Definition phase of each project. These requirements documents are refined to a greater and greater level of detail until they are ultimately translated into the detailed technical designs used by product development staff to code and test the product.

Throughout this process of specifying requirements and developing products, the Product Roadmap may continue to be updated with changing requirements. There can be a one-to-many relationship between the Product Roadmap and individual products under development. Through a change control process, critical requirements identified in the Product Roadmap can be incorporated into a selected product at various points during its development.

Please see “Managing Product Requirements Using the SourceForge Tracker” on page 280 for additional guidelines and recommended best practices for managing your Product Roadmap and product-specific requirements using SourceForge.

Training Program

Training for staff members is an important activity conducted throughout the product lifecycle. Providing quality training programs contributes to the success of your development projects, results in a more productive work force, and promotes long-term staff retention.

Every development project will have training needs specific to that project, such as skills training in areas as coding, testing methodologies, and the use of new hardware and software platforms. In addition, organizations will likely have training needs that are either independent of any specific development project, or shared across more than one project simultaneously.

The CDP provides a model for managing both project-specific and broader organizational training needs using SourceForge. In the CDP model, a Training Group is responsible for coordinating the training needs of each development project and identifying opportunities to standardize certain programs at an organizational level. The activities and deliverables asssociated with each training activity are managed to completion using the Task Manager, and records and evaluations of training activities completed by all staff members are maintained in an Organizational Training Project in SourceForge.

Please see “Managing your Organization’s Training Programs using SourceForge” on page 260 for additional guidelines and recommended best practices for managing your training activties using SourceForge.

Templates and Organizational Policies

The CDP describes many recommended processes for planning, developing, testing, releasing, and supporting your software products. Many templates, checklists, sample policies, and other similar materials are provided for your use. These are intended to provide guidelines, but it is expected that some modification will likely be necessary before you can effectively use them in your organization.

Before you begin your first project following the CDP guidelines, and again before beginning any subsequent projects, it is important to ensure that the necessary templates, policies and other materials are tailored to the specific needs of your organization, reviewed with and approved by the appropriate stakeholders, and posted to your Software Process Database. In particular, those policies covering cross-lifecycle activities such as the development and ongoing use of your project plan should be completed before beginning your first project. Any policies covering activities conducted in the early stages of your project, such as management of subcontractors, plus templates for all Product Defintion phase deliverables should also be completed.

Other templates and organizational policies may be customized and completed later as they become necessary. It is recommended that a plan for completing these items be created and managed using the Task Manager in the Software Process Database. This will ensure high visibility across the organization into the status of these (and other) process-related activities.

SW-CMM NOTE

The SW-CMM requires a number of written organizational policies for managing various activities. Examples of the organizational policies required for compliance with Level 2 and 3 Key Process Areas are included throughout the CDP materials. In general, they are located in the chapter or best practices section that describes processes for conducting each activity.

Preparatory and Cross-Lifecycle Activities Exit Criteria

Before beginning the Product Definition phase, the preparatory activities described in this section should be completed.

You may wish to conduct a Preparatory Activities Exit Review meeting prior to initiating the Product Definition Phase of your project. Exit Review meetings should be held with key stakeholders from all impacted organizations. Based on whether or not you are beginning your first project following the CDP guidelines, select the appropriate checklist to ensure all exit criteria are met.

✓	**Preparatory Activities Exit Checklist for New CDP Projects**
❑	Complete a SourceForge training program to ensure all staff are comfortable using SourceForge in their daily activities.
❑	Review all CDP materials.
❑	Work with the VA Software Professional Services organization on any product or process-related customizations.
❑	Complete a CDP training program to ensure all staff are familiar with the process-related activities they will be conducting throughout the CDP.
❑	Establish a Software Engineering Process Group to manage the organization's process improvement activities.
❑	Create a Software Process Database to collect and maintain all of the organization's process-related materials.
❑	Create a SourceForge project to manage your development project. The SourceForge project should contain all elements specified on page 52.
❑	Establish and document your organization's Peer Review process. You can use the templates provided in "Managing your Peer Review Processes using SourceForge" on page 249 as guidelines.
❑	Establish and document your organization's Software Quality Assurance audit process. You can use the templates provided in "Managing Software Quality Assurance Audits using SourceForge" on page 257 as guidelines.
❑	Establish and document your organization's Product Requirements Management process. You can use the best practices provided in "Managing Product Requirements Using the SourceForge Tracker" on page 280 as guidelines.
❑	Establish and document your organization's Training Program. You can use the best practices provided in "Managing your Organization's Training Programs using SourceForge" on page 260 as guidelines.
❑	Establish and document any remaining organizational policies or procedures.
❑	Complete a Preparatory Activities Exit Review using this checklist. Post the checklist to the SourceForge Document Manager.

✓	Preparatory Activities Exit Checklist for Subsequent CDP Projects
❑	Check the VA Software CDP website at *http://www.vasoftware.com/cdp/* for any updates you may wish to incorporate into your Software Process Database.
❑	Review your Software Process Database for any updates.
❑	Check in with your Software Engineering Process Group to address any process-related issues and coordinate your activities with those of the rest of the organization.
❑	Update standard training programs to reflect any changes to your CDP processes.
❑	Review the training needs of your staff and complete any necessary training. Include comprehensive training for new staff.
❑	Complete a refresher CDP training program to ensure all staff are familiar with the process-related activities they will be conducting throughout the CDP.
❑	Work with the VA Software Professional Services organization on any product or process-related customizations.
❑	Complete a Preparatory Activities Exit Review using this checklist. Post the checklist to the SourceForge Document Manager.

Preparatory and Cross-Lifecycle Activities Templates

The following table provides a quick reference to the templates used for Preparatory and Cross-Lifecycle activities.

All templates are provided on the CD included with this manual and at the following URL: *https://www.vasoftware.com/cdp/*

Table 14. Preparatory and Cross-Lifecycle Activities Templates

Ref#	Template	File Location
0.1.1	Preparatory Activities Exit Checklist	\prep\[0.1.1]prepexit.dot \prep\[0.1.1]prepexit.pdf

Please reference “Numbering” on page 10 for additional information on document numbering conventions.

CHAPTER 3

Product Definition Phase

This chapter includes the following information:

- Product Definition Phase Overview
- Product Definition Phase Workflow Process Diagram
- Product Definition Phase Inputs
- Product Definition Phase Activities and Deliverables
- Product Definition Phase Exit Criteria
- Using SourceForge to Support the Product Definition Phase
- Product Definition Phase Templates

Product Definition Phase Overview

The primary goal of the Product Definition phase is to define and document all product requirements in sufficient detail to enable engineering and other technical staff to create detailed product design documents. At the conclusion of this phase, a comprehensive overview of the business, technical, and scheduling requirements of the product should be complete, forming the basis for the more detailed design work to be conducted in the Design phase.

To enter the Product Definition phase, a compelling business case for beginning the project is required. If available, historical data from previous projects is also desirable as it forms a basis for continually improving the processes associated with the development lifecycle.

Activities conducted in the Product Definition phase include:

- Conducting technical and market research
- Identifying, analyzing, and finalizing product requirements
- Estimating the size, effort, cost, and organizational impacts associated with the project
- Identifying and selecting subcontractors
- Creating high-level plans and schedules
- Developing comprehensive requirements documents
- Establishing mechanisms for tracking the project

The materials produced in this phase provide the basis for all future technical and non-technical activities surrounding product development; therefore it is critical that sufficient time and effort be allowed for producing complete and high quality deliverables.

The following diagram indicates the deliverables necessary to exit the Product Definition phase. The suggested contents of each activity and deliverable and the recommended process for producing them are defined in the following sections.

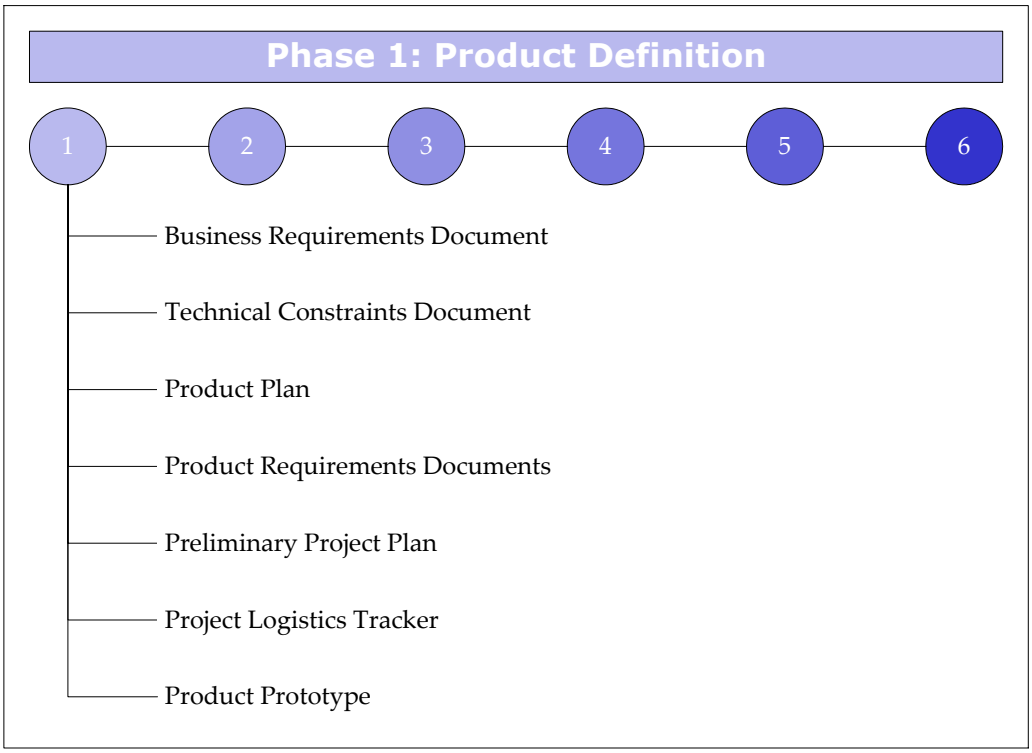


Figure 6. Product Definition Phase Deliverables

The Product Definition Phase Workflow Process

The following diagram illustrates the recommended best practice workflow process for the Product Definition Phase.

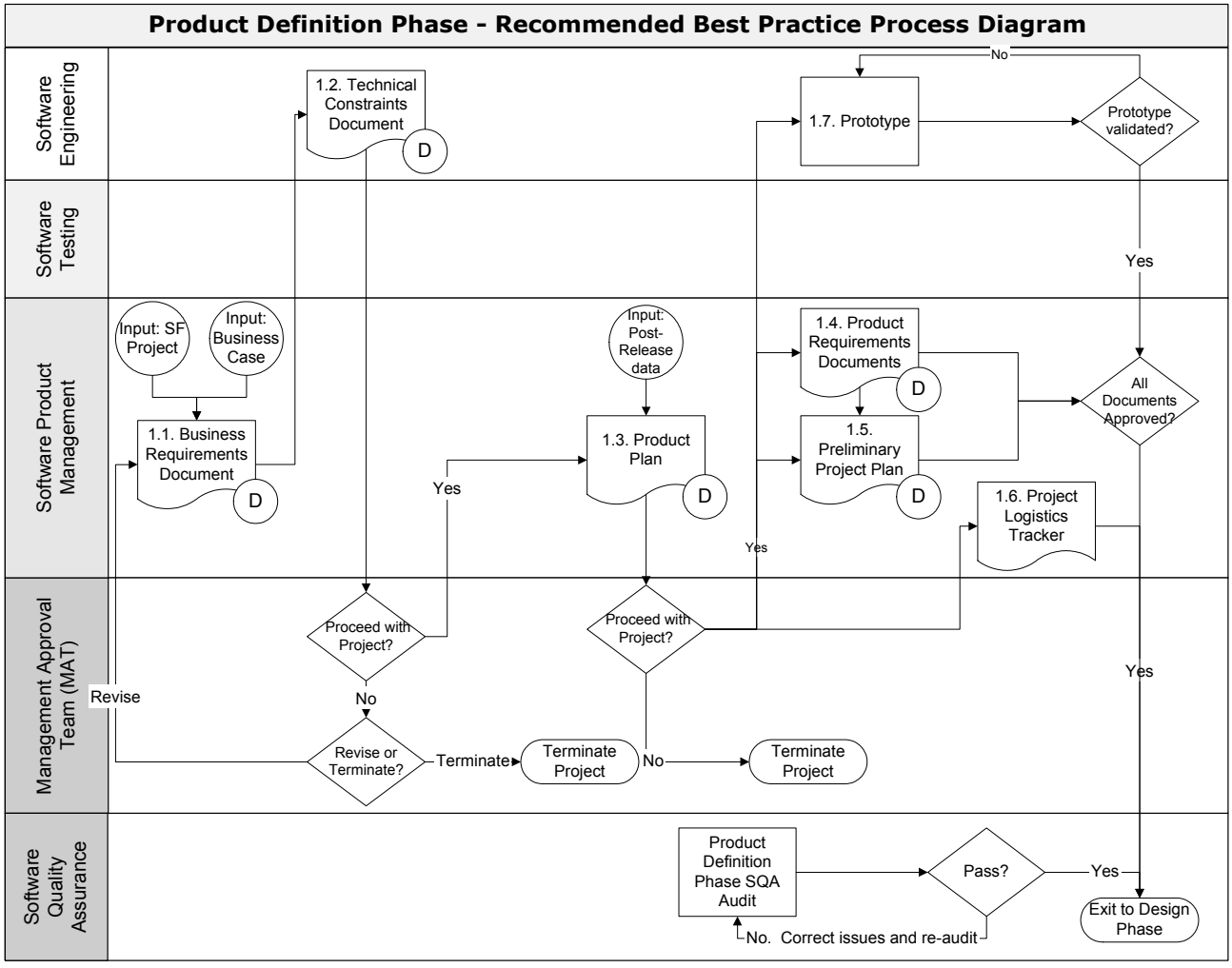


Figure 7. Product Definition Phase Workflow Process

Please see "Workflow Processes" on page 11 for a full description of all flowchart notation.

Product Definition Phase Inputs

The following inputs are required to enter the Product Definition phase:

Preparatory Activities

As described in the Cross-Functional Impacts chapter, a number of preparatory activities are recommended prior to initiating the Product Definition Phase. These include preparation of standard processes, training of staff, and the establishment of a SourceForge project in which to manage the project.

Post-Release Analysis Data from Previous Projects

Measuring and analyzing the successes and challenges of a previous project is a key factor in continuous process improvement. If available, data from previous projects should be collected and reviewed prior to initiating a new project. Please see Chapter 8, Sustaining Engineering Phase for additional information on collecting post-release data for measurement and analysis.

Business Case

A business case is a compelling reason for initiating the development of the product. The business case generally comes from the Product Roadmap and serves as the justification to begin the Product Definition phase. The business case will be expanded in the Business Requirements Document.

SourceForge Project

The SourceForge project created prior to beginning the Product Definition phase should now be populated with the materials you have created.

SourceForge Component	Contents
At the beginning of the phase	
Project Information	Project-level information such as project description, members, and roles.
Task Manager	A task folder used for managing action items using the Task Manager.
Document Manager	A Document Manager folder hierarchy established and used for managing document approval and storage. The appropriate CDP process templates retrieved from your Software Process Database and stored in your project's Document Manager. At this stage, only those process templates needed for the early planning stages of your project are required. (Business Requirements Document, Technical Constraints Document, and Product Plan.)
Software Process Database	A Software Process Database used for managing all standard procedures, process templates, checklists, and other materials. Please see *"Managing your Software Process Database using the SourceForge Document Manager" on page 247* for best practices on managing your process database.
Trackers	If feature requests, product defects, or other product-related artifacts will be used as input for building the project's business case, one or more Trackers should be identified in which these artifacts will be managed. Please see *"Managing Product Requirements Using the SourceForge Tracker" on page 280* for best practices on setting up and using Trackers.
Discussion Forums and Mailing Lists	One or more Discussion Forums and Mailing Lists for discussion of Product Definition and other topics.
SCM Repository	Access to an SCM repository may be established at this stage if needed for the development of software for a product prototype.

SourceForge Component	Contents
By the end of the phase	
Task Manager	A project created using the Task Manager for management of your project plan. Please see *"Managing your Project Plan Using the SourceForge Task Manager" on page 297* for best practices on managing a project plan.
Document Manager	Final, approved versions of all required Product Definition phase documents stored in the Document Manager. All applicable procedures, process templates, and other materials retrieved from your Software Process Database and stored in the project's Document Manager.
Tracker	A Project Logistics Tracker established to track critical estimates, resources, and risks.

Product Definition Phase Activities and Deliverables

All activities and deliverables are identified by number in the Product Definition Phase Workflow Process Diagram.

Descriptions of each activity or deliverable, suggested contents and acceptance criteria, and dependencies on other activities or deliverables are provided with each item. Activities or deliverables for which peer review or a Software Quality Assurance audit is recommended are identified. Considerations for tailoring deliverables to meet specific organizational needs are indicated where appropriate, and recommended uses for SourceForge are included at the end of each section.

Icons

Activities and deliverables for which templates are provided are identified by the symbol shown here. Please see “Activities and Deliverables” on page 13 for a complete list of all symbols used to identify recommended activities and uses for SourceForge.

1.1. Business Requirements Document

Suggested Owner: Software Product Management

The Business Requirements Document serves as the explanation and justification for continuing the definition and development of the product. Its creation should serve as a means for management to examine a new product idea, or a significant change to an existing one, prior to investing significant resources. Often, the creation of a Business Requirements Document is driven by the organization's long-term product strategy as defined in the Product Roadmap.

The audience for the Business Requirements Document is generally the management team that must sponsor, approve, and/or fund the project. This team is referred to throughout the CDP materials as the Management Approval Team.

The Business Requirements Document should include:

Business Objectives

The business objectives desired as a result of developing the product. These objectives may include:

- Enabling entry into a new market
- Expanding sales by providing new product features or functionality
- Satisfying an internal business requirement

Customer Requirements

Include any specific customer requirements driving the development of the new release. These may include enhancements required by current customers and/or prospects. Based on the target audience for the product, customers may also be internal to your own company.

Market Opportunity

If the product will be offered for sale, or is expected to generate direct revenue in any way, provide information as to the target market for the product. For example, "Adoption of this product is expected to be highest among new customers running Platform X." Include detail on any potential competitors and the expected position of your product relative to any competitive products. Provide as much detail as possible as this will serve as the starting point for development of a more comprehensive Marketing Plan in the Launch Preparation phase.

Product Strategy

Describe the overall strategy for the product's lifecycle, from release through product end of life.

Projected Activity

Include any specific activity goals, such as revenue, quantity sold, or adoption rate by current user base. If the product will be used internally, define the projected relationship with the internal customer.

Architectural Considerations

List any architectural and high-level product development considerations. These will be expanded in the Technical Constraints Document.

Internal Resources

Identify the organization, division, or department expected to be responsible for development of the product. Specific individuals need not be identified at this stage; responsiblity for specific functional areas will be identified in the Product Plan.

Risk Analysis and Contingency Planning

List any major risks known at this point. Detailed risk analysis and contingency planning will be completed in the Product Plan.

Acceptance criteria for the Business Requirements Document should include:

- Completion of peer review. The Iterative Document Review and Approval Process and a Cross-Functional Peer Review Meeting are recommended. Peer review criteria are included as an appendix to the document template.

Recommended Uses for SourceForge:

Use the **SourceForge Document Manager** to store the Business Requirements Document at all stages of its development (draft through final) and to manage the Iterative Document Review and Approval Process. See Figure 21 on page 276 for additional detail on this process.

Use the **SourceForge Task Manager** to manage the schedule for all Product Definition phase deliverables. See “SourceForge Project” on page 52 for additional information on managing action items prior to creating the Project Plan.

1.2. Technical Constraints Document

Suggested Owner: Software Engineering

The Technical Constraints Document serves as an initial technical effort estimate for the development of the product. The Technical Constraints Document may be developed separately, or as part of the Business Requirements Document ; the business requirements plus its technical constraints together form a comprehensive overview of the product. The audience for the Technical Constraints Document is generally the Management Approval Team.

The Technical Constraints Document should include:

Feasibility Analysis

Based on the requirements specified in the Business Requirements Document, describe what is required from a technical or development perspective in order to fully realize the product under consideration. Include such items as:

- Estimated technical design time
- Estimated development time
- Estimated time needed for software testing and quality assurance activities
- Any additional research or other preliminary work required
- Costs associated with staff and other resources (cost will be a rough estimate at this point)
- Technical and other skills required
- Hardware, software, and other materials required
- Assumptions and dependencies

Constraints

Constraints on any of the items described in the feasibility analysis should be explicitly stated. Examples of constraints include:

- Available resources (headcount, personnel with necessary skills, conflicts with other projects)
- Financial constraints
- Upcoming deadlines (e.g. end of a fiscal quarter, predetermined date for product release, etc.)
- Critical computer resources

Where possible, propose mitigation strategies for removing the constraint. For example, if properly trained personnel are not available, include an effort estimate for training, outsourcing some or all of the project, or hiring new resources.

Description of Enabling Technology

Describe any technology that will be required in order to successfully complete the project. Some examples are:

- Computers and peripherals for software development
- Computers and peripherals for software testing
- Target computer environment software
- Other support software

Estimated Delivery Schedule

If there are options available, based on resources, constraints, product features or other factors, indicate them here.

Subcontractors and other External Resources

Identify any areas where subcontractors or other external resources may be needed. The process for subcontractor selection will be included in the Product Plan.

Third Party Products

If any third party products will be required, either for integration or for use by staff during the development of the product, include the requirements here. Specific products need not be identified at this stage; the process for identifying and obtaining needed third party products will be included in the Product Plan.

Acceptance criteria for the Technical Constraints Document should include:

- Completion of peer review. The Iterative Document Review and Approval Process and a Cross-Functional Peer Review Meeting are recommended. Peer review criteria are included as an appendix to the document template.

Dependencies for the Technical Constraints Document include:

- Business Requirements Document

Recommended Uses for SourceForge:

Use the **SourceForge Document Manager** to store the Technical Constraints Document at all stages of its development (draft through final) and to manage the Iterative Document Review and Approval Process. See Figure 21 on page 276 for additional detail on this process.

Use the **SourceForge Task Manager** to manage the schedule for all Product Definition phase deliverables. See “SourceForge Project” on page 52 for additional information on managing action items prior to creating the Project Plan.

1.3. Product Plan

Suggested Owner: Software Product Management

The Product Plan is intended to provide a comprehensive, high-level overview of both the product and its associated development project. The Product Plan should take the conceptual material presented and approved in the Business Requirements and Technical Constraints Documents, address any constraints, and provide a comprehensive view of the product and the development effort required to produce it. Review of the Product Plan also serves as a vehicle for collecting feedback and input from product customers and stakeholders.

The Product Plan should be viewed as the "Master" document in which all administrative, process, and technical elements of the project are elaborated. It covers activities and deliverables for all stages of the CDP, and will be referenced frequently by various functional groups when creating detailed plans for their activities. Some key elements of the Product Plan are product requirements, roles and responsibilities, process-related activities and deliverables, plans for subcontractor selection, peer review, Software Quality Assurance audits, training requirements, and risk analysis and contingency planning.

Preliminary size, effort, and cost estimates are also important components of the Product Plan. It is understood that such estimates may need to be revised in later stages when additional information is available; however an initial estimate of the time and resources needed to develop and test each proposed feature will help accurately define the scope of the project in the early planning stages.

A recommendation is to create and finalize the Product Plan in several iterations. The first draft, prepared by Software Product Management, should contain the entire set of proposed features. During the first peer review cycle, Software Engineering should perform preliminary size, effort, and cost analyses for each proposed product feature and provide these estimates as input along with any other review comments. Worksheets are provided to assist users in estimating the size of the product, turning the size estimates into effort estimates, and finally turning effort estimates into cost estimates. After completing the preliminary size, effort, and cost estimates, the Product Plan should then be revised to reflect a product scope that is in line with the budget, resources, and timeline available for the product.

The Product Plan is cross-functional in nature, and generally serves as the first introduction to the product available to the organization as a whole. It should not be expected that all readers of the Product Plan will have read the Business Requirements and Technical Constraints Documents that were intended for a Management Approval Team audience. The Product Plan generally serves as the formal decision point for the Management Approval Team; if the product proposed is not desirable or satisfactory at this point, the project may be terminated or deferred.

The Product Plan should include:

Background Information

Since the Product Plan will be reviewed by a cross-functional audience, many of whom may not have read the the Business Requirements and Technical Constraints Documents, it is important to provide a summary of the product's goals and justifications.

Roles and Responsibilities

Identify the groups and key individuals responsible for execution of the project.

Requirements Overview

For a new product, include an overview of all product features. For a revision to an existing product, include an overview of all new product features, plus an overview of all enhancements to be made to existing product features. Include the source of the requirement to facilitate traceability. Requirements should be prioritized and adjusted if necessary based on early size, effort, and cost estimates by Software Engineering.

Size

Estimate the size of the product to be developed. It is often useful to estimate the size on an individual feature level, to aid in prioritizing and refining the list of features that will be included in the product. Size estimation worksheets are provided.

Effort

Based on the estimated size of the product, estimate the effort associated with developing it. If size estimates were done on a feature level, effort estimates should also be done on a feature level. Worksheets are provided to translate size estimates into effort estimates.

Cost

Based on the estimated effort, estimate the cost of the project, both in resources and in financial cost. Include both fixed and variable costs, sustaining engineering throughout the expected life of the product (including updates and revisions), and Return on Investment (ROI) if the product will be sold. Worksheets are provided to translate effort estimates into cost estimates.

Hardware and Software Requirements

Describe the specification, configuration requirements, cost estimate, suggested procurement, receipt, and installation considerations for all required hardware and system software. Identify any hardware or software resources that are critical to the success of the project as critical computer resources. These critical computer resources will be tracked and managed throughout the project. Information regarding specific enabling technology, critical computer resources, and other hardware and software requirements is included in the Technical Constraints Document.

Selection of Subcontractors and other External Resources

Areas where subcontractors or other external resources may be needed are identified in the Technical Constraints Document. The actual subcontractors or other external resources to be used should be identified in the Product Plan. If this is not possible or desirable, the process by which the appropriate subcontractors will be selected should be included. A subcontractor selection worksheet is provided.

Third Party Products

Areas where third party products will be required, either for integration or for use by staff during the development of the product, are identified in the Technical Constraints Document. The actual third party products to be used should be identified in the Product Plan. If this is not possible or desirable, the process by which the appropriate third party products will be selected should be included.

Software Quality Assurance Plan

Software Quality Assurance as defined by the CDP refers to quality oversight activities such as periodic audits. Documents and other work products that will be subject to such audits should be identified in the Product Plan.

Peer Review Plan

Documents and other work products that will be subject to peer review should be identified in the Product Plan.

Training Requirements

Training requirements for staff involved in developing the product and supporting activities should be identified in the Product Plan. Training activities surrounding using and supporting the finished product will be specified in the Product Training Plan.

Process Tailoring

The CDP establishes a set of standard processes to be followed to design, develop, test, and release a product. This set of standard processes may be modified, according to established tailoring guidelines, based on the needs of each particular project. This tailored set of processes should be identified in the Product Plan.

Software Configuration Management

Documents, code, and other artifacts to be placed under formal change control should identified in the Product Plan. The point at which each artifact will be placed under change control should also be provided.

Quality

If there are any specific quality goals associated with the release, such as percentage of outstanding bugs and feature requests resolved, number of bugs permissable upon release, or any other metrics, include them here. Also include any requirements for adherance to applicable customer, corporate, or industry standards. The Software Test Plan completed in the Design phase will establish the quality criteria associated with each testing activity.

Performance

Define the minimum performance characteristics the finished product must achieve. If applicable, define performance characteristics for each hardware configuration that is to be supported. Performance may be defined in relation to a previous product version, for example "Performance must meet or exceed that of the previous version."

Reference Platforms

Define the combinations and permutations of hardware and software on which the product will be built and deployed.

Field Testing and other Customer Involvement

If customers will be involved in providing early feedback, either through a formal Beta program, informal demonstrations, or other form of early access, it should be described here. The strategy for conducting the early access programs and incorporating customer feedback can be defined in more detail in the Field Test Plan.

Delivery

Describe the delivery mechanism for the product. Include whether it will be distributed internally and/or externally to the organization, whether physical media will need to be produced, whether an electronic distribution mechanism will be used, whether manufacturing needs to be planned, and other such considerations.

Risk Analysis and Contingency Planning

List any specific risks and their mitigation strategies.

Milestones and Deliverables

List the significant milestones and deliverables associated with the product. As detailed development estimates and other planning has not yet taken place, schedule-driven deliverables should be limited at this point to those having a fixed date.

Acceptance criteria for the Product Plan should include:

- Completion of peer review. The Iterative Document Review and Approval Process and a Cross-Functional Peer Review Meeting are recommended.
- Completion of Size, Effort, and Cost Estimation Worksheets.
- Completion of Subcontractor Selection Worksheet, if applicable.
- All requested deviations from standard processes approved.
- All cross-functional impacts accounted for and acceptable.
- Budget for subcontractors, necessary hardware or software, and other costs approved.
- All requirements identified and clearly stated.
- All requirements feasible, consistent with each other, and testable.
- Additional peer review criteria are included as an appendix to the document template.

Dependencies for the Product Plan include:

- Business Requirements Document
- Technical Constraints Document

Software Quality Assurance Audits

Software Quality Assurance audits are recommended for a number of the components of the Product Plan. Audits of processes used to produce various components of the Product Plan and activities conducted as a result of the material in the Product Plan are also recommended.

Reference the Software Quality Assurance section of the Product Plan and the Software Quality Assurance audit checklists for each CDP phase for a complete list of all artifacts, processes, and activities to be audited.

Recommended Uses for SourceForge:

Use the **SourceForge Document Manager** to store the Product Plan at all stages of its development (draft through final) and to manage the Iterative Document Review and Approval Process. See Figure 21 on page 276 for additional detail on this process.

Use the **SourceForge Task Manager** to manage the schedule for all Product Definition phase deliverables. See “SourceForge Project” on page 52 for additional information on managing action items prior to creating the Project Plan.

1.4. Product Requirements Documents

Suggested Owner: Software Product Management

Product Requirements Documents are intended to provide a comprehensive, detailed overview of both the product and the development project required to produce it. You may wish to create one or more Product Requirements Documents, based on such factors as the availability of Product Management resources, the scope of the work contemplated, and the division of responsibilities associated with different product features. For purposes of this document, a group of multiple Product Requirements Documents will be referenced.

The Product Requirements Documents take the overview descriptions of the product features presented in the Product Plan, and expand and refine them to a level of detail sufficient for Software Engineering to create detailed technical specifications. The Product Requirements Documents also form the basis for other functional departments, such as Marketing and Technical Support, to create their own detailed plans for activities surrounding the product. The Product Requirements Documents go through a rigorous, often lengthy, Iterative Document Review and Approval Process, and generally serve as the formal reference point for all future inquiries regarding the product. They should include:

Introductory Information

Include overview information such as scope, background, constraints, and assumptions and dependencies related to the development of the product feature or features described in the Product Requirements Document.

Specific Requirements

Describe each requirement in detail. Include the problem the requirement will solve and its specific functions. If prioritization was done based on engineering effort estimates in the Product Plan, list the requirements by order of priority. The level of detail must be sufficient to enable the creation of technical specifications and other design documents such as User Interface specifications and prototypes, without additional clarification.

Include the source of the requirement to facilitate traceability and acceptance criteria to be used when creating test cases.

Example Use Cases

Include examples of expected use cases for each area of functionality. Additional use cases will be defined during the development of the Technical Design Documents in the Design phase.

Feature Requests, Bugs and Other Tracker Artifacts

If feature requests, bugs, and/or other types of custom tracker artifacts are being addressed, include a file export with the relevant artifacts from the SourceForge Tracker in the Product

Requirements Document. Please see *Chapter 9, Managing Product Requirements Using the SourceForge Tracker* on page 280 for additional detail on this process.

External Dependencies

List any external dependencies, if applicable. Include dependencies on suppliers of third party components that will be integrated into the product or upon which there is some other dependency.

Cross-functional Impacts

The Product Definition phase is generally too early to fully describe the specific activities of each functional group. Each group will also be producing their own detailed plans in later phases. However, any key impacts that can be identified at this point should be included. Some examples of groups or functions that may be impacted by the development of a new product are:

- Marketing
- Technical Support
- Administration
- Upgrade and Migration
- User Documentation
- Training
- Software Testing
- Software Quality Assurance
- Sales Engineering
- Professional Services
- Finance
- Legal
- IT

Deferred Items

Maintain a list of items deferred to a later release to provide a reference point when planning a future release.

Open Issues

Maintain a list of all open issues related to the topics discussed in the Product Requirements Documents.

Acceptance criteria for the Product Requirements Documents should include:

- Completion of peer review. The Iterative Document Review and Approval Process and a Cross-Functional Peer Review Meeting are recommended.
- All cross-functional impacts accounted for and acceptable.
- All requirements identified and clearly stated.
- All requirements feasible, consistent with each other, and testable.
- Additional peer review criteria are included as an appendix to the document template.

Dependencies for the Product Requirements Documents include:

- Product Plan

Software Quality Assurance Audits

Software Quality Assurance audits are recommended for a number of the components of the Product Requirements Documents. Audits of processes used to produce various components of the Product Requirements Documents and activities conducted as a result of the material in these are also recommended.

Reference the Software Quality Assurance section of the Product Plan and the Software Quality Assurance audit checklists for each CDP phase for a complete list of all artifacts, processes, and activities to be audited.

Recommended Uses for SourceForge:

Use the **SourceForge Document Manager** to store the Product Requirements Documents at all stages of their development (draft through final) and to manage the Iterative Document Review and Approval Process. See Figure 21 on page 276 for additional detail on this process.

Use the **SourceForge Tracker** to manage the Feature Request process as an integral part of defining the requirements for the product. See Figure 26 on page 296, for additional detail on this process.

Use the **SourceForge Task Manager** to manage the schedule for all Product Definition phase deliverables. See “SourceForge Project” on page 52 for additional information on managing action items prior to creating the Project Plan.

1.5. Preliminary Project Plan

Suggested Owners: Software Product Management, Software Testing, and Software Engineering

The Preliminary Project Plan is intended to provide a milestone-level schedule of the project, providing a basis for the more detailed and comprehensive planning to be conducted in the Design phase.

The results of future activities and deliverables, such as the creation of the Technical Design Documents in the Design phase, may necessitate adjustments to the milestone dates indicated in the Preliminary Project Plan; however, the number of iterations required to finalize more detailed plans in later stages will be greatly reduced if the milestone dates indicated in the Preliminary Project Plan remain as fixed as possible. Establishing standard nomenclature for product features and related terminology will also reduce the amount of effort required to integrate the Preliminary Project Plan with any project plans developed separately. Building a working product prototype, ensuring that the requirements of the product are well-researched and clearly defined, and analyzing the post-release data collected from previous projects, will facilitate accuracy in estimating the time and resources that will be needed to complete the project.

Milestones that can be included in the Preliminary Project Plan are:

- Completion dates for all key deliverables required to exit the Product Definition phase
- Entry dates for all future CDP dates
- Exit Review Meeting dates for all future CDP phases
- Date for finalizing Project Plan
- Coding start and end dates
- Unit testing start and end dates
- System testing start and end dates
- Alpha period start and end dates
- Beta period start and end dates
- Date for Release Readiness Checkpoint
- Date for product general availability, for packaged products
- Date for release to internal or external customers, for products intended for internal use only or for integration as a sub-component of a larger product
- Milestone dates established in the Business Requirements Document, Technical Constraints Document, and Product Plan, such as dates for required training, subcontractor selection, and other dates
- Dates for creation and peer review of all documents due prior to completion of the Final Project Plan

- Dates for any Software Quality Assurance audits due prior to completion of the Final Project Plan

You can create your Preliminary Project Plan using the Task Manager, or you may wish to create the initial document in Microsoft Project and then import it into the Task Manager for ongoing management. Recommended best practices for working with Microsoft Project and the Task Manager to develop and manage your project plan are provided in "Best Practices for Working with Microsoft Project and the Task Manager" on page 304.

You may also wish at this time to establish separate task folders for each functional group that will responsible for a significant section of the Project Plan. Please see "Best Practices for Managing Multiple Projects in the Task Manager" on page 306 for additional guidelines on using task folders.

Acceptance criteria for the Preliminary Project Plan should include:

- Dates established for all key project milestones according to the established procedure
- Completion of peer review. The Iterative Document Review and Approval Process and a Cross-Functional Peer Review Meeting are recommended.
- The Preliminary Project Plan created in or imported into the SourceForge Task Manager.

Dependencies for the Preliminary Project Plan include:

- Product Plan
- Post-release data from previous projects

Software Quality Assurance Audits

Software Quality Assurance audits are recommended for a number of the components of the Preliminary Project Plan. Audits of processes used to produce various components of the Product Plan and activities conducted as a result of the material in the Preliminary Project Plan are also recommended.

Reference the Software Quality Assurance section of the Product Plan and the Software Quality Assurance audit checklists for each CDP phase for a complete list of all artifacts, processes, and activities to be audited.

Recommended Uses for SourceForge:

Use the **SourceForge Document Manager** to store the Preliminary Project Plan at all stages of its development (draft through final) and to manage the Iterative Document Review and Approval Process. See Figure 21 on page 276 for additional detail on this process.

Use the **SourceForge Task Manager** to create the Preliminary Project Plan or to import a Preliminary Project Plan created in Microsoft Project. Please see “Managing your Project Plan Using the SourceForge Task Manager” on page 297 for additional information on this process.

1.6. Establish Project Logistics Trackers

Suggested Owner: Software Product Management

In the Product Plan a number of factors were identified as having the potential to impact the success of the development project. These factors include estimates for size, effort, cost, critical computer resources, and risks. Other factors may also have been identified. The schedule, quality, and ultimate success of the development project are dependent upon these factors remaining consistent with their original estimates, or being managed appropriately if the original estimates must be adjusted. In order to provide a high level of visibility and actively manage these factors, trackers should be set up and populated with each item to be tracked.

Each item being tracked should be entered as a unique tracker artifact and should be assigned to the individual responsible for its management. That person is then responsible for updating the artifact if and when the item's status changes to a point where it needs to be flagged as a potential issue. The artifact will then be routed to the appropriate person for review and action according to the workflow rules you establish. In this way, deviations from critical estimates of size, effort, cost, and resources will be identified as soon as they become imminent. The occurance or increased likelihood of a risk will be flagged and the appropriate contingency plan can be implemented. Visibility is also provided at any time into the status of these critical factors, and the monitoring function can be used to inform users when the status of any item has changed.

You may wish to create one or more trackers to manage the following categories of items:

Size

The size of the software product as estimated in the Product Plan.

Effort

The effort associated with the software product as estimated in the Product Plan.

Cost

The cost associated with the software product as estimated in the Product Plan.

If size, effort, and cost estimates were calculated on a per-feature basis, you may wish to enter a multiple artifacts for each feature.

Critical Computer Resources

All critical computer resources identified in the Product Plan.

Risks and Contingency Plans

All risks identified in the Product Plan. For risks with associated contengency plans, include the contingency plan in the tracker artifact. If that risk reaches a high likelihood of occurance, the contingency plan can be triggered.

Please see “Managing Project Logistics using the SourceForge Tracker and the Monitoring Function” on page 264 for additional detail on setting up trackers, using configurable fields, and managing changes to the project.

Acceptance criteria for the Project Logistics Tracker should include:

- Tracker artifacts created for all factors being tracked. All artifacts assigned to the responsible parties.

Dependencies for the Project Logistics Tracker include:

- Product Plan

Software Quality Assurance Audits

Software Quality Assurance audits are recommended for a number of the components of the Project Logistics Tracker. Audits of processes used to produce various components of the Project Logistics Tracker and activities conducted as a result of the material in the Project Logistics Tracker are also recommended.

Reference the Software Quality Assurance section of the Product Plan and the Software Quality Assurance audit checklists for each CDP phase for a complete list of all artifacts, processes, and activities to be audited.

Recommended Uses for SourceForge:

Use the **SourceForge Tracker** to create and manage all project logistics artifacts.

Use the **SourceForge Task Manager** to create the Preliminary Project Plan or to import a Preliminary Project Plan created in Microsoft® Project. Please see “Managing your Project Plan Using the SourceForge Task Manager” on page 297 for additional information on this process.

Use **SourceForge Monitoring** to be informed via email when the status of any tracked artifact has changed.

1.7. Product Prototype

Suggested Owner: Software Engineering

A Product Prototype is intended to validate the product architecture, including its required scalability and performance, and to further refine the effort estimates included in the Technical Constraints Document. A full working prototype may not be necessary or desirable in all cases; you may choose to produce architecture documents and/or high-level technical designs in addition to or instead of a working prototype. For a minor product release, this step may not be necessary.

Creating a Product Prototype, architecture and/or high-level Technical Design Documents should be completed in parallel with the Product Requirements Documents, as discoveries made during the creation of the prototype may impact the feature set or other considerations detailed in the Product Requirements Documents.

Acceptance criteria for the Product Prototype should include:

- Completion of peer review. Code Review is recommended for a product prototype. The Iterative Document Review and Approval Process is recommended for architecture or high-level technical design documents.

Dependencies for the Product Prototype include:

- Technical Constraints Document

Recommended Uses for SourceForge:

Use the **SourceForge Document Manager** to store any documentation associated with the Product Protytpe at all stages of its development (draft through final) and to manage the Iterative Document Approval Process. See Figure 21 on page 276 for additional detail on this process.

Use CVS as your **Software Configuration Management (SCM)** tool to manage all code developed during the Product Definition and other phases. Use the **SourceForge SCM interface** to browse the contents of your SCM repository.

Associate code developed for the prototype with product requirements **Tracker** artifacts where applicable.

Use the **SourceForge Task Manager** to manage the schedule for all Product Definition phase deliverables.

Product Definition Phase Exit Criteria

During the Product Definition phase, members of all stakeholder groups will have worked closely to ensure that the scope of the release has been satisfactorily defined.

A Product Definition Phase Exit Review meeting should be held with key stakeholders from all impacted organizations. The following checklist should be used to ensure all exit criteria are met.

✓	Product Definition Phase Exit Checklist
❑	SourceForge project established containing all elements specified on page 68.
❑	Business Requirements Document completed, peer-reviewed, and approved by required approvers. Final version stored in the Document Manager and locked.
❑	Technical Constraints Document completed, peer-reviewed, and approved by required approvers. Final version stored in the Document Manager and locked.
❑	Product Plan completed, peer-reviewed, and approved by required approvers. Final version stored in the Document Manager.
❑	Product Requirements Documents completed, peer-reviewed, and approved by required approvers. Final versions stored in the Document Manager and locked.
❑	Preliminary Project Plan completed, peer-reviewed, approved by required approvers, and imported into the Project Managment Console.
❑	All project training activities scheduled for completion in the Product Definition phase completed.
❑	All Training Evaluation Forms completed and posted to the Organizational Training project on SourceForge.
❑	Project Logistics Tracker established and all size, effort, and cost estimates, critical computer resources, and risks entered as artifacts.
❑	Product Prototype completed, peer-reviewed, and approved by required approvers.
❑	Product Definition phase Software Quality Assurance audit checklist completed. Checklist posted to the Document Manager.
❑	Periodic reviews of subcontractor progress completed, if applicable.
❑	Product Definition Phase Exit Review successfully completed using this checklist. Checklist posted to the Document Manager.

Using SourceForge to Support the Product Definition Phase

SourceForge supports the goals of the Product Definition phase by allowing you to:

- Manage feature request and other requirements input using the **Tracker.**
- Manage the document creation and approval process using the **Document Manager.**
- Set up and manage your project plan using the **Task Manager.**
- Manage changes to critical estimates, resources, and risks using the **Tracker** and **Monitoring**.
- See *SourceForge and the Capability Maturity Model for Software, A Supplement to the SourceForge Collaborative Development Process* for additional detail on using SourceForge and the CDP to satisfy SW-CMM Level 2 and 3 requirements.

Product Definition Phase Templates

The following table provides a quick reference to all templates used for the Product Definition phase deliverables and activities.

All templates are provided on the CD included with this manual and at the following URL: *https://www.vasoftware.com/cdp/*

Table 15. Product Definition Phase Templates

Ref#	Template	File Location
1.1.1	Business Requirements Document	\proddef\[1.1.1]brd.dot \proddef\[1.1.1]brd.pdf
1.2.1	Technical Constraints Document	\proddef\[1.2.1]tcd.dot \proddef\[1.2.1]tcd.pdf
1.3.1	Product Plan	\proddef\[1.3.1]prodplan.dot \proddef\[1.3.1]prodplan.pdf
1.4.1	Product Requirements Document	\proddef\[1.4.1]prd.dot \proddef\[1.4.1]prd.pdf
1.5.1	Project Plan	\proddef\[1.5.1]projplan.mpp \proddef\[1.5.1]projplan.pdf
1.8.1	Subcontractor Selection Worksheet	\proddef\[1.8.1]subcontractor.dot \proddef\[1.8.1]subcontractor.pdf
1.9.1	Size Estimation Worksheet	\proddef\[1.9.1]size.dot \proddef\[1.9.1]size.pdf
1.10.1	Effort Estimation Worksheet	\proddef\[1.10.1]effort.dot \proddef\[1.10.1]effort.pdf
1.11.1	Cost Estimation Worksheet	\proddef\[1.11.1]cost.dot \proddef\[1.11.1]cost.pdf
1.12.1	SQA Checklist Product Definition Phase	\proddef\[1.12.1]sqaproddef.dot \proddef\[1.12.1]sqaproddef.pdf
1.13.1	Product Definition Phase Exit Checklist	\proddef\[1.13.1]proddefexit.dot \proddef\[1.13.1]proddefexit.pdf

Please reference “Numbering” on page 10 for additional information on document numbering conventions.

CHAPTER 4

Design Phase

This chapter includes the following information:

- Design Phase Overview
- Design Phase Workflow Process Diagram
- Design Phase Inputs
- Design Phase Activities and Deliverables
- Design Phase Exit Criteria
- Using SourceForge to Support the Design Phase
- Design Phase Templates

Design Phase Overview

The primary goal of the Design phase is to create the detailed and comprehensive design documents required prior to beginning development of the product. The Project Plan is also expanded and finalized during this phase, and a Launch Team established that will meet regularly throughout the remaining phases of the project to monitor its ongoing progress.

To enter the Design phase, completed documentation from the Product Definition phase is required, including Product Requirements Documents, Product Prototype (if created), and a Preliminary Project Plan.

Activities conducted in the Design phase include:

- Creating detailed technical and other design documents
- Creating plans for related activities such as field testing, quality assurance, and documentation
- Finalizing a comprehensive Project Plan
- Establishing Launch Team

The materials produced in this phase build upon the documents finalized in the Product Definition phase, and provide the basis for future development, software testing, and project management activities.

The following diagram indicates the deliverables necessary to exit the Design phase. The suggested contents of each activity and deliverable and the recommended process for producing them are detailed in the following sections.

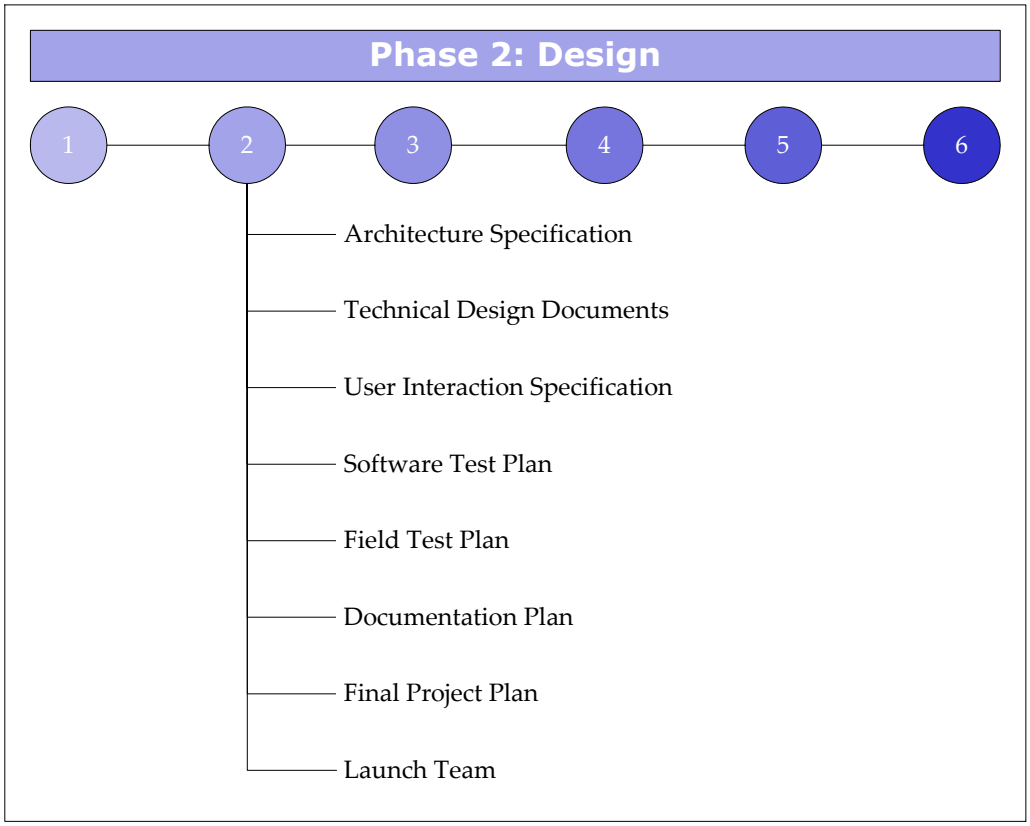


Figure 8. Design Phase Deliverables

The Design Phase Workflow Process

The following diagram illustrates the recommended best practice workflow process for the Design phase.

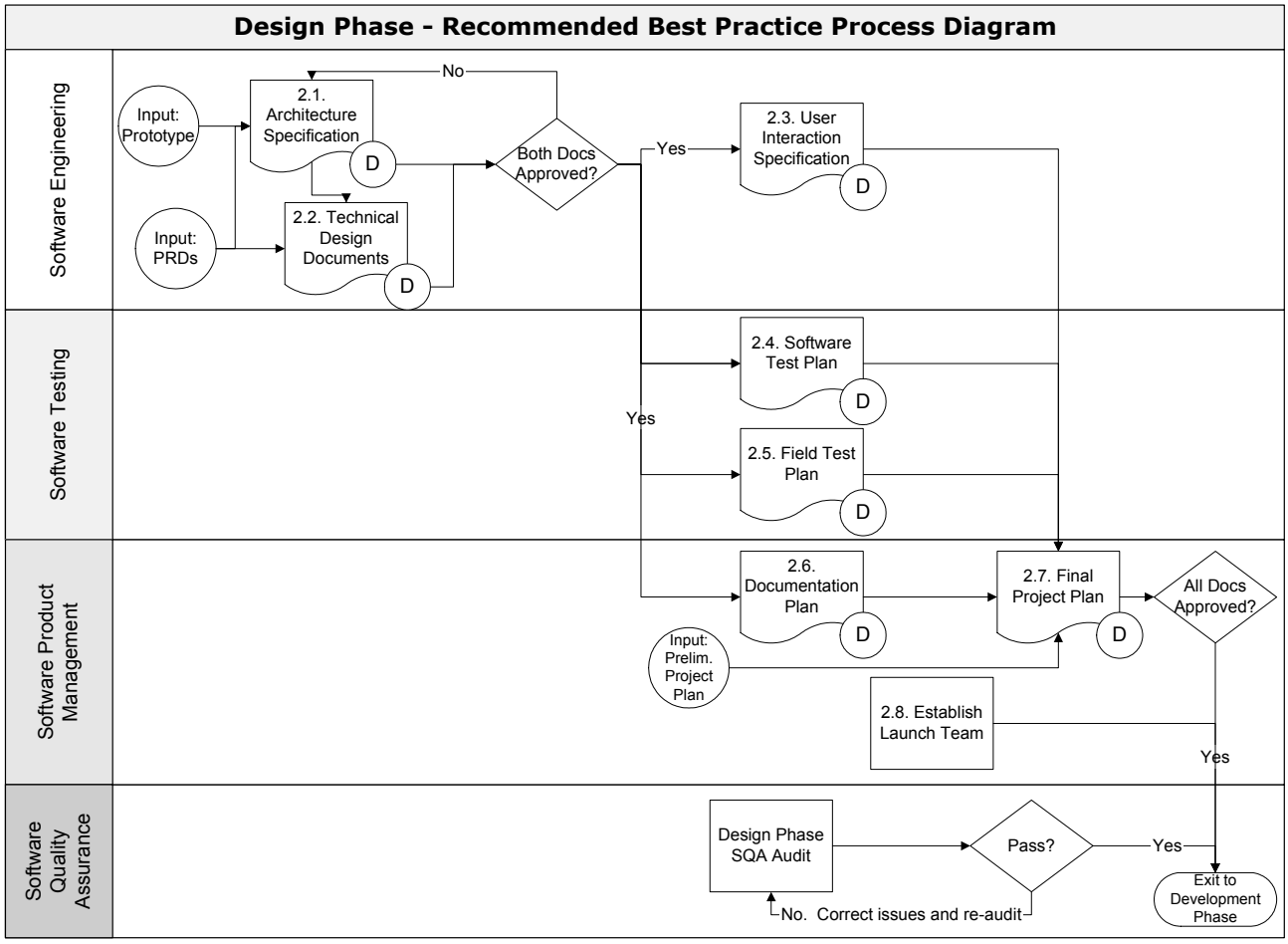


Figure 9. Design Phase Workflow Process

Please see Please see “Workflow Processes” on page 11 for a full description of all flowchart notation.

Design Phase Inputs

The following inputs are required to enter the Design phase:

Product Requirements Documents

The Product Requirements Documents created in the Product Definition phase form the basis for creating the Technical Design Documents and the User Interaction Specification.

Product Prototype

The Product Prototype, the preliminary architecture and/or high-level Technical Design Documents created in the Product Definition phase, serve as a basis for the more detailed Architecture Specification and Technical Design Documents.

Preliminary Project Plan

The milestone-level Preliminary Project Plan will be expanded and finalized during the Design phase.

Project Logistics Tracker

The critical estimates, resources, and risks identified in the Product Definition phase will be tracked throughout the Design phase and the remaining phases of the CDP.

SourceForge Project

The SourceForge project created prior to beginning the Product Definition phase should now be populated with the materials you have created.

SourceForge Component	Contents
At the beginning of the phase	
Project Information	Project-level information such as project description, members, and roles.
Document Manager	A Document Manager folder hierarchy established for use in managing document approval and storage, now containing final versions of all documents created in the Product Definition phase and all applicable procedures, process templates, and other materials retrieved from your Software Process Database. Other related documents, such as meeting minutes, notes, and reference materials, may also be stored. The Document Manager will have automatically stored copies of all draft versions and the review history of all documents.
Software Process Database	A Software Process Database used for managing all standard procedures, process templates, checklists, and other materials.
Trackers	Trackers containing all feature requests, bugs, and other artifacts that will be addressed in the product. A Project Logistics Tracker to track critical estimates, resources, and risks.
Discussion Forums and Mailing Lists	One or more Discussion Forums and Mailing Lists used for discussion of various project-related topics.
SCM Repository	An SCM repository containing any code developed for the Product Prototype.
Task Manager	A project in the Task Manager used for management of your project plan.
By the end of the phase	
Task Manager	Final Project Plan created in or imported into the Task Manager.
Document Manager	Final, approved versions of all required Design phase documents stored in the Document Manager. A new Document Manager folder created to store Launch Team meeting minutes and other documents. Design phase exit checklist stored in the Document Manager.

Design Phase Activities and Deliverables

All activities and deliverables are identified by number in the Design Phase Workflow Process Diagram.

Descriptions of each activity or deliverable, suggested contents and acceptance criteria, and dependencies on other activities or deliverables are provided with each item. Activities or deliverables for which peer review or a Software Quality Assurance audit is recommended are identified. Considerations for tailoring deliverables to meet specific organizational needs are indicated where appropriate, and recommended uses for SourceForge are included at the end of each section.

Icons

Activities and deliverables for which templates are provided are identified by the symbol shown here. Please see “Activities and Deliverables” on page 13 for a complete list of all symbols used to identify recommended activities and uses for SourceForge.

2.1. Architecture Specification

Suggested Owner: Software Engineering

The Architecture Specification is intended to provide a detailed description of the product architecture. The Architecture Specification covers the overall product from a high level perspective without describing the specific details of database schemas, APIs, and other implementation considerations of individual components. It should describe how different components of the product are layered and how they interact with each other. The Architecture Specification should be developed prior to, or in conjunction with, the Technical Design Documents.

The specific contents of the Architecture Specification will vary greatly based on the individual product being described. It may not be necessary for new versions of a product where an earlier Architecture Specification is still applicable.

Acceptance criteria for the Architecture Specification should include:

- Completion of peer review. The Iterative Document Review and Approval Process and a Cross-Functional Peer Review Meeting are recommended.
- Completion of all required elements sufficient to enable Software Engineering to write the Technical Design Documents.
- Additional peer review criteria are included as an appendix to the document template.

Dependencies for the Architecture Specification include:

- Product Prototype, if applicable
- Product Requirements Documetns

Recommended Uses for SourceForge:

Use the **SourceForge Document Manager** to store the Architecture Specification at all stages of its development (draft through final) and to manage the Iterative Document Review and Approval Process. See Figure 21 on page 276 for additional detail on this process.

Use **SourceForge Discussion Forums and Mailing Lists** to collaboratively discuss and resolve any technical design challenges.

Use the **SourceForge Task Manager** to manage the schedule for all Design phase deliverables.

2.2. Technical Design Documents

Suggested Owner: Software Engineering

The Technical Design Documents, sometimes known as Technical Specifications, describe in detail the technical requirements and methods that will be used for developing the product or product feature. The Technical Design Documents focus primarily on specific component areas of the product, providing all the necessary details for beginning the implementation, e.g. concrete object model, database schema, application programming interfaces (APIs), etc. Technical Design Documents must conform to the overall product architecture, and therefore should be written following, or in conjunction with, the Architecture Specification.

The specific contents of the Technical Design Documents will vary based on many factors, including the nature and scope of the product or product feature, the coding language being used [Java, C++, Perl, etc.], dependencies upon third party tools or applications, such as databases, application servers, or operating systems, and other factors.

The Technical Design Documents should include the following minimum information:

- Sufficient detail to enable software engineers to write and test the necessary code
- Test cases specified for Technical Unit Testing in the Development phase
- Effort estimates to assist in project planning

Additional, specific information to be included in the Technical Design Documents can include:

- Technical challenges
- User interface implementation
- Database implementation
- Application programming interface considerations
- Sequence diagrams
- Class or object diagrams
- Exception handling
- Implementation risks
- Selected tools for code development and verification
- Dependencies on third parties
- Technical assumptions
- Coding guidelines and standards to which development will adhere
- future enhancement considerations
- Considered but dropped design ideas

Acceptance criteria for the Technical Design Documents should include:

- Completion of peer review. The Iterative Document Review and Approval Process and a Cross-Functional Peer Review Meeting are recommended. Additional peer review criteria are included as an appendix to the document template.

Dependencies for the Technical Design Documents include:

- Product Requirements Documents
- Architecture Specification

Software Quality Assurance Audits

Software Quality Assurance audits are recommended for a number of the components of the Technical Design Documents. Audits of processes used to produce various components of the Technical Design Documents and activities conducted as a result of the material in the Technical Design Documents are also recommended.

Reference the Software Quality Assurance section of the Product Plan and the Software Quality Assurance audit checklists for each CDP phase for a complete list of all artifacts, processes, and activities to be audited.

Recommended Uses for SourceForge:

Use the **SourceForge Document Manager** to store the Technical Design Documents at all stages of their development (draft through final) and to manage the Iterative Document Approval Process. See Figure 21 on page 276 for additional detail on this process.

Use **SourceForge Discussion Forums and Mailing Lists** to collaboratively discuss and resolve any technical design challenges.

Use the **SourceForge Tracker** to reference the details of requested product features and to communicate with requestors for any additional technical detail required to complete the Technical Design Documents.

Use the **SourceForge Task Manager** to manage the schedule for all Design phase deliverables.

2.3. User Interaction Specification

Suggested Owner: Software Engineering or Software Product Management

The User Interaction Specification is intended to describe the manner in which a user will interact with the product. The User Interaction Specification only pertains to applications that will have a user and/or administrator interface. In the case of embedded software, for example, a User Interaction Specification may not be required.

For software products, user interaction generally includes such elements as task flow, navigation, other user-performed actions, display of tabular and form data, page layout, and textual and graphical elements. The goal of the User Interaction Specification is to provide sufficient detail to enable Software Engineering to develop the code to render the specified user interface. Software Testing may also use the User Interaction Specification as input when developing Test Scripts in the Development phase.

Wireframes, or diagrams indicating the placement of all elements on each screen, may also be created as part of, or in addition to, the User Interaction Specification. Entity Relationship Diagrams (ERDs) are also useful to user interaction design, allowing designers to see the structure of the data they will be presenting or gathering.

In some cases, it may be sufficient to develop a master User Interaction Specification, or Style Guide, that describes the user interaction parameters clearly enough that Software Engineering is able to develop the code without supplementary page-specific wireframe diagrams. If wireframes are desired, a master User Interaction Specification may also serve as a reference guide, from which wireframes for each product or product feature are developed.

The User Interaction Specification can include:

- Configuration considerations
  - For example, 8-bit color, 800 x 600 with no horizontal scrolling
- Coding guidelines
  - For example, HTML coding guidelines for Web applications
- Style sheets
  - Style sheets should describe the font face, size, colors, and other considerations for each action to be performed within the application
- Color palette

- Window design and layout
  - This should include look and feel considerations, descriptions of tables, forms, and buttons, and descriptions of all actions, e.g. clicking, sorting, etc., and their results
- Task flow
- Component interaction
- Guidelines for use of logos and other graphic elements
- Error messaging
- Data rendering considerations
- Text guidelines
  - Guidelines regarding specific wording to use or avoid, where and how to use text, date formats, etc.
- Navigation guidelines
- Internationalization and localization guidelines
- Where and how to access support or additional information
- Wireframes
- ERDs

Acceptance criteria for the User Interface Specification should include:

- Completion of peer review. The Iterative Document Review and Approval Process is recommended. Additional peer review criteria are included as an appendix to the document template.

Dependencies for the User Interaction Specification include:

- Product Requirements Documents

Software Quality Assurance Audits

Software Quality Assurance audits are recommended for a number of the components of the User Interaction Specification. Audits of processes used to produce various components of the User Interaction Specification and activities conducted as a result of the material in the User Interaction Specification are also recommended.

Reference the Software Quality Assurance section of the Product Plan and the Software Quality Assurance audit checklists for each CDP phase for a complete list of all artifacts, processes, and activities to be audited.

Recommended Uses for SourceForge:

Use the **SourceForge Document Manager** to store the User Interaction Specification at all stages of its development (draft through final) and to manage the Iterative Document Review and Approval Process. See Figure 21 on page 276 for additional detail on this process.

Use the **SourceForge Task Manager** to manage the schedule for all Design phase deliverables.

2.4. Software Test Plan

Suggested Owner: Software Testing

The Software Test Plan is intended to provide a comprehensive overview of all Software Testing activities that will take place prior to the release of the product. The contents of the Software Test Plan may vary based on the scope of the project and other considerations. The testing activities described in the Software Test Plan form the basis for the more detailed Test Scripts to be developed and executed during the Development and Release Preparation phases.

The Software Test Plan can include:

Release Scope

A summarized list of the new features, enhanced functionality, and pre-existing product defects that will be addressed in the release. In addition to product features, quality goals such as product performance, scalability, and maintainability should also be included.

Test Phases

Define the phases of testing that will be conducted prior to release, e.g. Functional Unit Testing, Technical Unit Testing, Alpha System Testing, Beta Testing, regression testing, performance testing, scalability testing, integration testing, or Final System Testing. All types of testing are described in more detail in Chapter 5, Development Phase and Chapter 6, Release Preparation Phase.

Test Processes

Define the processes by which each testing phase will be conducted, e.g. manual testing, automated testing.

Test Environment

Define all hardware, software, and other test environment elements required for completion of all testing. This test environment will become the product reference environment, or list of supported hardware and software configurations as defined in the Product Plan Reference Platforms section.

Test Cases

List all test cases that will be written, including contents, owners, and dates for creation and execution. Test cases are overviews of the functional areas to be tested during product quality assurance activities, and are generally accompanied by more detailed test scripts.

Test Scripts

List all test scripts that will be created, including contents, test type [automated, manual, other], goals, owners, and dates for creation and execution. Test scripts should be created to cover functional testing, performance testing, and environmental testing, plus integration testing if necessary. Test scripts are described in more detail in Chapter 5, Development Phase.

Risks

List the risks, constraints, and mitigation plans associated with each. All additional risks identified in the Software Test Plan should be input into the Project Logistics Tracker for monitoring.

Test Cycle Milestones

Provide a milestone-level schedule of all significant testing activities. Some milestones that may be included are completion and execution of test scripts and test cases, availability of test environments, start and end dates for Functional Unit Testing, Technical Unit Testing, Alpha System Testing, Beta Testing, Final System Testing, and other testing activities.

Reporting

Provide a list of all reports that will be produced both during and at the conclusion of the test cycle. Reports that may be produced include pre-existing product defects (bugs), periodic reports of bugs found and fixed during each testing cycle, build analyses describing the overall acceptance level of individual builds, automated test results, and system and other test results.

Product Defect Prioritization

Include a policy statement regarding actions to be taken for each product defect priority level (as entered with each artifact in the SourceForge Tracker Priority field). For example, you may wish to release a product only after all highest-priority defects have been corrected, or limit the number of any priority defects that may be allowed to remain in the product.

Product Release Criteria

Define the quality level that must be in place before the product can be released, plus any other applicable release criteria. Quality standards should be defined for product functionality, performance, and scalability on all required reference environments.

Product Maintenance Plan

Define the strategy for issuing product maintenance releases. Include the triggering event or events for issuing a release, e.g. pre-established time period, accumulation of high-priority bugs, or other criteria. Define the method or methods for issuing maintenance releases, e.g. patches, full product replacement, electronic or CD delivery. Product maintenance activities will be conducted during the Sustaining Engineering phase.

Acceptance criteria for the Software Test Plan should include:

- Completion of peer review. The Iterative Document Review and Approval Process is recommended. Additional peer review criteria are included as an appendix to the document template.
- All risks posted to the Project Logistics Tracker.

Dependencies for the Software Test Plan include:

- Product Plan
- Product Requirements Documents

Recommended Uses for SourceForge

Use the **SourceForge Document Manager** to store the Software Test Plan at all stages of its development (draft through final) and to manage the Iterative Document Review and Approval Process. See Figure 21 on page 276 for additional detail on this process.

Use the **SourceForge Task Manager** to manage the schedule for all Design phase deliverables.

Use the **SourceForge Tracker** to manage any risks identified in the Software Test Plan.

2.5. Field Test Plan

Suggested Owner: Software Product Management or Software Quality Assurance

The Field Test Plan is intended to provide an overview of all pre-release field testing activities that will be conducted prior to the product release. Some common pre-release field testing phases include Alpha and Beta Testing. Field testing may be limited to users internal to the organization, or may be opened to an audience of customers or other third party users external to the organization.

The key differentiator between field testing and other testing activities is the participation of users outside of the Software Testing team.

Some of the goals of field testing include:

- Soliciting feedback from customers who will be using the product
- Gauging the level of satisfaction with the product feature set
- Collecting feature enhancement requests for consideration in the current or future releases
- Supplementing the efforts of the Software Testing team to identify product defects
- Providing early access to current or prospective customers
- Providing an opportunity for Marketing to gather customer testimonials required for product announcement
- Providing a period of stable product quality to allow functional groups such as Documentation, Marketing, Sales, and Training to conduct pre-release product-related activities
- Providing a period of stable product quality to allow third party or downstream developers within your organization to begin development work to integrate their components or products with your product

The Field Test Plan should include:

- Overview of testing phases and the objectives of each
- Length of each testing period
  - — Sufficient time is required for participants to receive and install the software, conduct testing activities, and report findings. If sufficient time is not available for these activities, you may wish to conducted a hosted demonstration, or other type of time-bounded field testing activity, minimizing the commitment required from participants, and maximizing the amount of feedback generated.
- Entry criteria for each testing period
  - — Product quality metrics such as number of bugs outstanding, completion of specified Software Testing activities, or successful implementation of specified product features

- Product delivery method
  - CD, product download, hosted demo, other methods
- Participant criteria
  - Define the criteria for establishing eligibility to participate, plus the method that will be used to identify and select participants.
- Management process
  - Define the process by which the field test will be conducted, including who will be responsible for identifying and selecting participants, distributing the product and supporting information to participants, providing support to participants, reviewing feedback, resolving any issues that may arise, and other management activities.
- Action plan for addressing reported product defects and feature requests
- Other follow-up activities

Acceptance criteria for the Field Test Plan should include:

- Completion of peer review. The Iterative Document Review and Approval Process is recommended. Additional peer review criteria are included as an appendix to the document template.

Dependencies for the Field Test Plan include:

- Product Plan
- Product Requirements Documents
- Preliminary Project Plan

Recommended Uses for SourceForge:

Use the **SourceForge Document Manager** to store the Field Test Plan at all stages of its development (draft through final) and to manage the Iterative Document Review and Approval Process. See Figure 21 on page 276 for additional detail on this process.

Use the **SourceForge Task Manager** to manage the schedule for all Design phase deliverables.

2.6. Documentation Plan

Suggested Owner: Software Product Management, Software Engineering, or Software Testing

The Documentation Plan is intended to provide a comprehensive outline of all technical documentation to be produced in support of the product, and should include an estimated schedule for its creation. The Documentation Plan should be created in parallel with the Product Requirements Documents and Preliminary Project Plan, to ensure that all product features are representated, and to maintain consistency with the overall project schedule.

Types of technical documentation that can be included are:

- New or updated user documentation: Hard copy or soft copy documentation aimed at an end user audience, e.g. User Guide, Site Preparation and Installation Guide, FAQs, or other documentation.
- Documentation for context-sensitive help to be built into the product
- Online documentation
- Internal documentation for Technical Support
- Application Programming Interface (API) documentation
- End user training materials

The Documentation Plan should include the following minimum information:

- An overview of all new documentation to be created
- An overview of all existing documentation that must be revised
- Document section reviewers and draft review schedule
- Dependencies on other deliverables, e.g. frozen code base, frozen user interface, etc.
- Tools and standards to be employed in the creation of the documentation
- Requirements for document localization

Acceptance criteria for the Documentation Plan should include:

- Completion of peer review. The Iterative Document Review and Approval Process is recommended. Additional peer review criteria are included as an appendix to the document template.
- Incorporation of relevant data from Product Requirements Documents and Preliminary Project Plan.

Dependencies for the Documentation Plan include:

- Product Requirements Documents
- Preliminary Project Plan

Recommended Uses for SourceForge:

Use the **SourceForge Document Manager** to store the Documentation Plan at all stages of its development (draft through final) and to manage the Iterative Document Approval Process. See Figure 21 on page 276 for additional detail on this process.

Use the **SourceForge Task Manager** to manage the schedule for all Design phase deliverables.

2.7. Final Project Plan

Suggested Owners: Software Product Management, Software Engineering, and Software Testing

The Final Project Plan should significantly expand on the milestone-level Preliminary Project Plan created in the Product Definition phase. Upon completion, the Final Project Plan should contain task-level detail for all activities necessary to ensure successful, on time completion of the project.

The Final Project Plan should be cross-functional and include tasks and deliverables required from all contributing functional groups. These functional groups may include, but are not limited to, Software Engineering, Software Testing, Software Product Management, Software Quality Assurance, Documentation, Marketing, Technical Support, Sales and other Field Readiness organizations. Tasks and deliverables assigned to subcontractors may also be broken out separately if desired. Owners for each task should also be identified.

Managing a lengthy project plan with many hundreds or even thousands of tasks can often become unwieldy. In such cases, it may be more desirable to create separate project plans for each major functional group, e.g a Software Engineering project plan, a Software Testing project plan, a Marketing project plan, and any others. The Task Manager allows you to establish separate task folders for each functional group that will responsible for a significant section of the Project Plan.

If you have established your Preliminary Project Plan using multiple task folders, you can import and update each functional group's project plan separately. You may need to confirm that milestones and critical path items are still in alignment. Please see "Best Practices for Managing Multiple Projects in the Task Manager" on page 306 for additional guidelines on using task folders.

It is also helpful to organize your Final Project Plan, or its component functional area project plans, according to the phases of the CDP. In this way, the activities and deliverables due for completion in each phase can be easily identified, and Phase Exit Reviews can be conducted more quickly.

Items that can be included in the Final Project Plan are:

- Completion dates for all key deliverables required to exit each phase of the CDP
- Dates for creation and peer review of all documents and activities throughout the remainder of the CDP.
- Entry dates for all future the CDP phases
- Exit Review Meeting dates for all future CDP phases
- Milestone dates established in all documents completed in the Product Definition and Design phases.
- Coding start and end date/s
  - Start and end dates for all development tasks
  - Milestone completion dates for all significant functional product areas
  - Dates for any unit and/or system testing to be done within the Software Engineering group
- Dates for regular engineering reviews
- Cutoff date for correction of product defects
- Code freeze dates
- Database schema freeze dates
- User Interface freeze dates
- Alpha and/or Beta period start and end dates (Field testing)
  - Milestones from the Field Test Plan
  - Start and end dates for fixing product defects or feature requests identified during field testing
- Software testing start and end dates
  - Completion dates for all test scripts and other preparatory documents
  - Completion dates for setup and configuration of necessary testing environments (hardware, software, etc.)
  - Start and end dates for all software testing activities
- Documentation dates
  - Start and end dates for all documentation to be produced
  - Review periods for all documentation
  - Other milestones from the Documentation Plan
- Dates for all Software Quality Assurance audits
- Dates for all activities and deliverables assigned to subcontractors, if being tracked separately

- Dates for significant Marketing-related events with product-related depedencies, if applicable
  - Production of collateral materials
  - Packaging and distribution
  - Events such as tradeshows and other product demonstrations
  - Press releases

Note: Marketing-related dates may be added or modified in the Release Preparation phase following completion of the Marketing and Channel Plan.

- Training dates

Note: Training-related dates may be added or modified in the Release Preparation phase following completion of the Product Training Plan

- Service, Support and other Field Readiness dates

Note: Service, Support and Field Readiness-related dates may be added or modified in the Release Preparation phase following completion of the Service and Support Plan.

- Dates for product release and post-release activities

Acceptance criteria for the Final Project Plan should include:

- Completion of peer review. The Iterative Document Review and Approval Process is recommended.
- Inclusion of all milestone, critical path, and general tasks.
- Final Project Plan or component functional area project plans created or exported into the Task Manager.

Dependencies for the Final Project Plan include:

- Product Plan
- Preliminary Project Plan
- Technical Design Documents
- User Interaction Specification
- Software Test Plan
- Field Test Plan
- Documentation Plan

Software Quality Assurance Audits

Software Quality Assurance audits are recommended for a number of the components of the Final Project Plan. Audits of processes used to produce various components of the Final

Project Plan and activities conducted as a result of the material in the Final Project Plan are also recommended.

Reference the Software Quality Assurance section of the Product Plan and the Software Quality Assurance audit checklists for each CDP phase for a complete list of all artifacts, processes, and activities to be audited.

Recommended Uses for SourceForge:

Use the **SourceForge Task Manager** to create the Final Project Plan or to export a Final Project Plan created in Microsoft Project.

2.8. Launch Team

Suggested Owner: Software Product Management

The Launch Team is a cross-functional team comprised of representatives from all groups involved in the development, release, support, and sustaining engineering of the product. A representative from the Software Engineering Process Group should also be a member of the Launch Team.

Once established, the Launch Team will meet regularly throughout the remaining phases of the project to monitor its ongoing progress and address any issues. The Launch Team is responsible for monitoring the Project Logistics Tracker on a regular basis to track any changes to critical estimates, resources, and risks. The Launch Team may be thought of as a Core Product Team or Steering Committee, whose primary goal is to ensure the successful, on-time, on-budget release of the product.

The Launch Team should serve as the first escalation point for any issues surrounding the project, the approval point for any deviations from the agreed upon schedule or processes, and the vehicle for communicating project status to the rest of the organization.

Acceptance criteria for establishing the Launch Team should include:

- Inclusion of stakeholders representing all impacted functional areas, including the Software Engineering Process Group
- Meeting time, location, and frequency established
- Method for reviewing project status established
  - Regular review of the Task Manager to identify at-risk items
  - Regular review of the Project Logistics Tracker
  - During the Development and Launch Preparation phases, regular review of quality metrics to continually assess product readiness
- Method of communicating status and action items to the broader organization
- Creation of SourceForge Document Manager folder for managing meeting notes and other documents
- Creation of a SourceForge discussion forum for Launch Team members

Dependencies for establishing the Launch Team include:

- Product Plan, from which roles and responsibilities are derived

Software Quality Assurance Audits

Software Quality Assurance audits are recommended for a number of the processes and activities conducted by the Launch Team.

Reference the Software Quality Assurance section of the Product Plan and the Software Quality Assurance audit checklists for each CDP phase for a complete list of all artifacts, processes, and activities to be audited.

Recommended Uses for SourceForge:

Use the **SourceForge Task Manager** to review project status on a regular basis.

Use the **SourceForge Document Manager** to store meeting minutes and all other relevant documents.

Use **SourceForge News** to publicize progress, issues, revisions to plans, and other information of general interest to the organization.

Use **SourceForge Discussion Forums and Mailing Lists** to discuss any topic related to the development, release, support, or sustaining engineering of the product.

Use the **SourceForge Tracker** to review the Project Logistics Tracker on a regular basis.

Design Phase Exit Criteria

During the Design phase, members of all stakeholder groups will have worked closely to ensure that detailed design documents have been created for all components of the product and its supporting activities.

A Design Phase Exit Review meeting should be held with key stakeholders from all impacted organizations. The following checklist should be used to ensure all exit criteria are met.

✓	Design Phase Exit Checklist
❑	SourceForge project now containing all elements specified on page 98.
❑	Architecture Specification completed, peer-reviewed, and approved by required approvers. Final version stored in the Document Manager and locked.
❑	Technical Design Documents completed, peer-reviewed, and approved by required approvers. Final version stored in the Document Manager and locked.
❑	User Interaction Specification completed, peer-reviewed, and approved by required approvers. Final version stored in the Document Manager and locked..
❑	Software Test Plan completed, peer-reviewed, and approved by required approvers. Final version stored in the Document Manager and locked.
❑	Field Test Plan completed, peer-reviewed, and approved by required approvers. Final version stored in the Document Manager and locked.
❑	Documentation Plan completed, peer-reviewed, and approved by required approvers. Final version stored in the Document Manager and locked.
❑	Final Project Plan completed, approved, and uploaded to the SourceForge Task Manager.
❑	Launch Team established.
❑	Periodic reviews of the Task Manager and the Project Logistics Tracker completed.
❑	All project training activities scheduled for completion in the Design phase completed.
❑	All Training Evaluation Forms completed and posted to the Organizational Training project on SourceForge.
❑	Design phase Software Quality Assurance audit checklist completed. Checklist posted to the Document Manager.

✓	**Design Phase Exit Checklist**
❑	Periodic reviews of subcontractor progress completed, if applicable.
❑	Design Phase Exit Review successfully completed using this checklist. Checklist posted to the Document Manager.

Using SourceForge to Support the Design Phase

SourceForge supports the goals of the Design phase by allowing you to:

- Manage the document creation and approval process using the **Document Manager.**
- Refine and manage your project plan using the **Task Manager.**
- Create and assign **Tracker** and **Task Manager** artifacts to the individual task level. Maintain traceability to Product Requirements Document feature requirements using the Create Association feature.
- Collaborate on design challenges using **Discussion Forums and Mailing Lists.**
- See *SourceForge and the Capability Maturity Model for Software, A Supplement to the SourceForge Collaborative Development Process* for additional detail on using SourceForge and the CDP to satisfy SW-CMM Level 2 and 3 requirements.

Design Phase Templates

The following table provides a quick reference to all templates used for the Design phase deliverables and activities.

All templates are provided on the CD included with this manual, and at the following URL: *https://www.vasoftware.com/cdp/*

Table 16. .Design Phase Templates

Ref#	Template	File Location
2.2.1	Technical Design Document	\design\[2.2.1]tdd.dot \design\[2.2.1]tdd.pdf
2.3.1	User Interaction Specification	\design\[2.3.1]uispec.dot \design\[2.3.1]uispec.pdf
2.4.1	Software Test Plan	\design\[2.4.1]testplan.dot \design\[2.4.1]testplan.pdf
2.5.1	Field Test Plan	\design\[2.5.1]fieldtest.dot \design\[2.5.1]fieldtest.pdf
2.6.1	Documentation Plan	\design\[2.6.1]docplan.dot \design\[2.6.1]docplan.pdf
2.7.1	Final Project Plan	\design\[2.7.1]finalprojplan.mpp \design\[2.7.1]finalprojplan.pdf
2.9.1	SQA Checklist Design Phase	\design\[2.9.1]sqadesign.dot \design\[2.9.1]sqadesign.pdf
2.10.1	Design Phase Exit Checklist	\design\[2.10.1]designexit.dot \design\[2.10.1]designexit.pdf

Please reference “Numbering” on page 10 for additional information on document numbering conventions.

CHAPTER 5

Development Phase

This chapter includes the following information:

- Development Phase Overview
- Development Phase Workflow Process Diagram
- Development Phase Inputs
- Development Phase Activities and Deliverables
- Development Phase Exit Criteria
- Using SourceForge to Support the Development Phase
- Development Phase Templates

Development Phase Overview

The primary goal of the Development phase is to develop and test the product code and associated materials such as user documentation. During this phase, all coding and testing activities internal to Software Engineering are completed, in addition to two or more phases of testing activities conducted by Software Testing. If code from third parties will be incorporated into the product, it will be accepted, validated, and integrated as appropriate during this phase. Additional internal and external documentation is also produced in preparation for the product launch.

To enter the Development phase, the detailed design documents completed in the Design phase are required. The Final Project Plan, managed using the SourceForge Task Manager, is also required and is used extensively to manage the many tasks and deliverables required for completion in the Development phase.

Activities conducted in the Development phase include:

- Developing all product code
- Conducting regular engineering reviews
- Conducting preliminary, module-specific technical unit and Functional Unit Testing
- Conducting integrated engineering system and Alpha System Testing
- Preparing draft user documentation, product training plans and product launch strategy

The goal of the development and testing activities completed in the Development phase is to produce a functional product of sufficient quality to enter a Beta Testing period involving participants external to the organization.

The following diagram indicates the activities and deliverables necessary to exit the Development phase. The suggested contents of each activity and deliverable and the recommended process for producing them are detailed in the following sections.

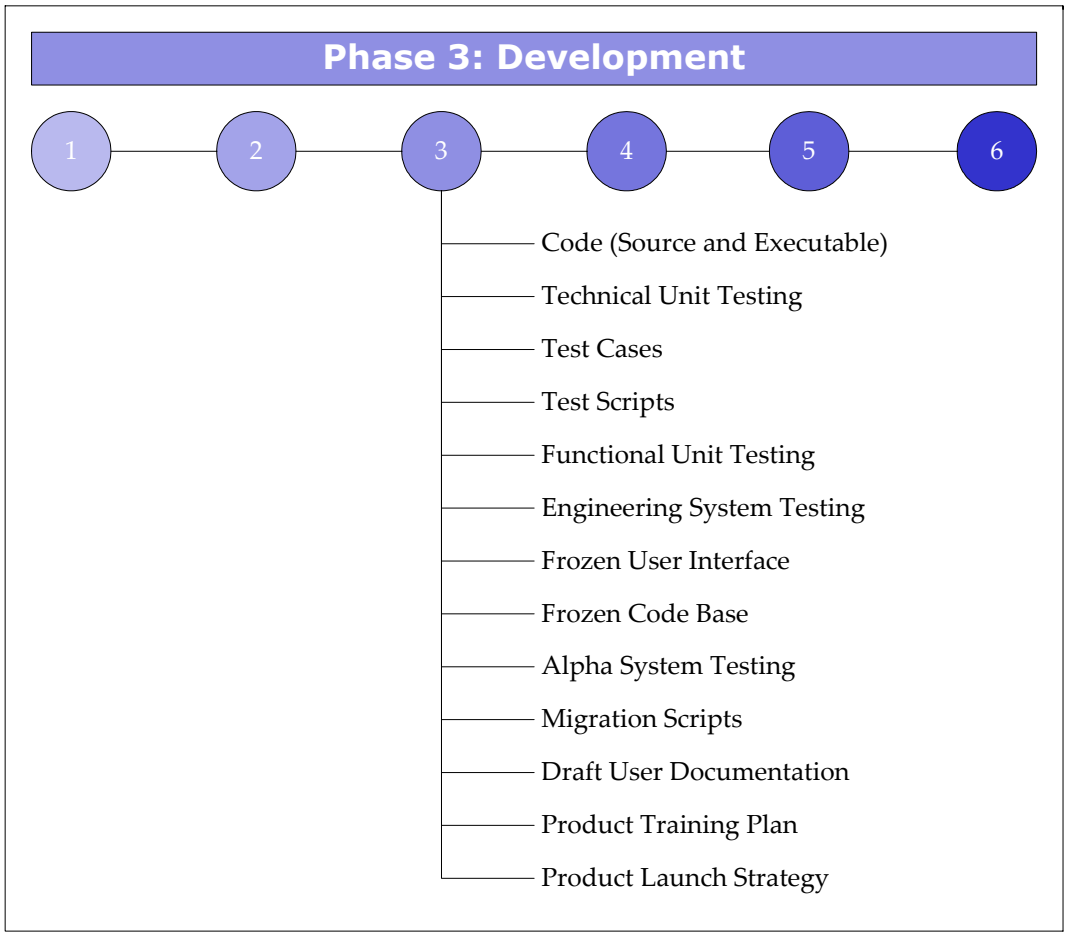


Figure 10. Development Phase Deliverables

The Development Phase Workflow Process

The following diagram illustrates the recommended best practice workflow process for the Development phase.

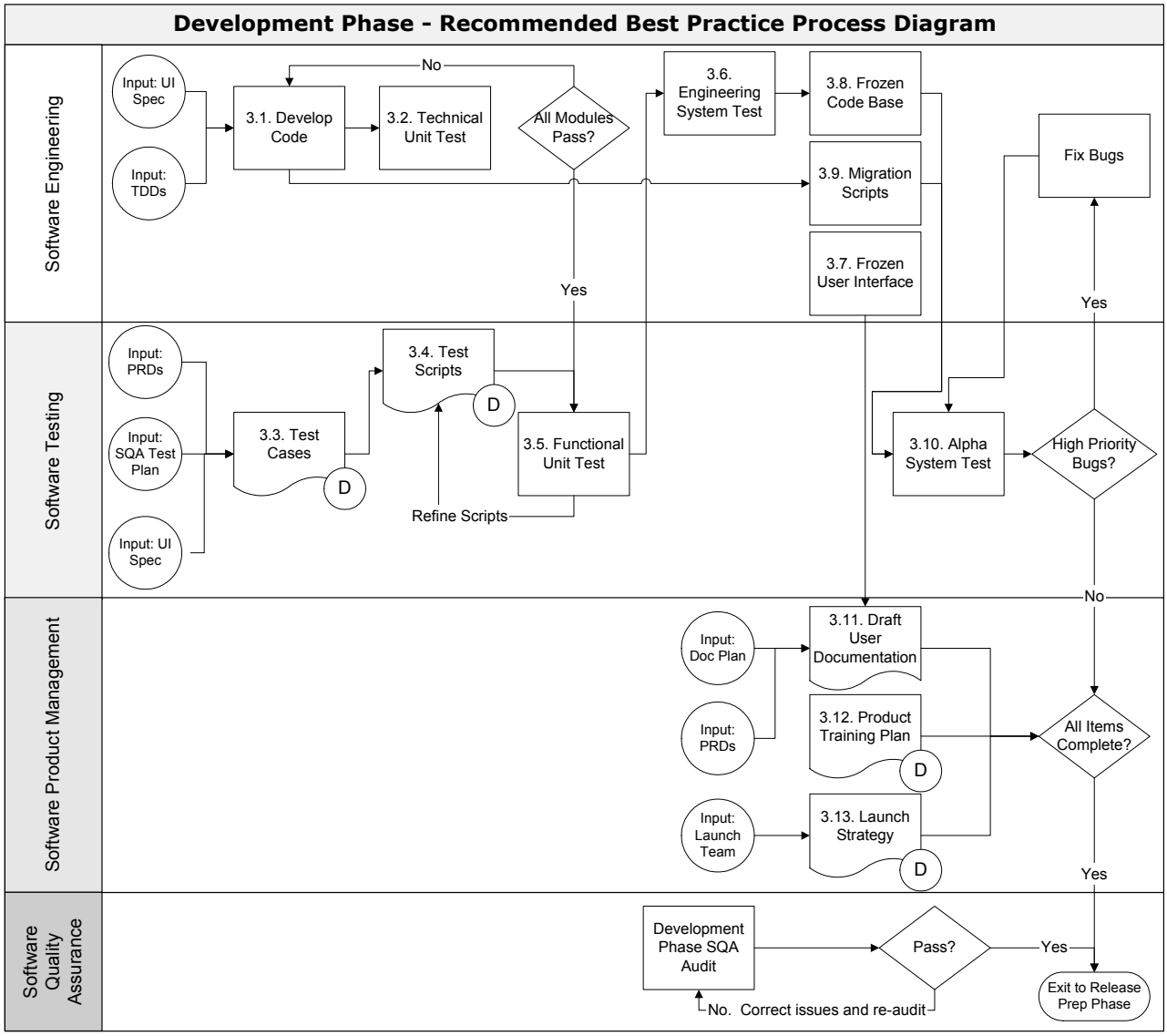


Figure 11. Development Phase Workflow Process

Please see Please see "Workflow Processes" on page 11 for a full description of all flowchart notation.

Development Phase Inputs

The following inputs are required to enter the Development phase:

Technical Design Documents

Technical Design Documents define how all product code will be developed, and can be considered the key input for entering the Development phase. The Technical Design Documents also provide input for Software Testing in preparing the Test Scripts.

Documentation Plan

The Documentation Plan will be followed duing the Development phase to produce the draft user documentation.

User Interaction Specification

The User Interaction Specification defines how all code will be developed to render the product user interface, and is used as the basis for developing and freezing the user interface during the Development phase.

Software Test Plan

The Software Test Plan provides a detailed overview of all quality assurance activities that will take place during the Development and other remaining phases. The Software Test Plan therefore guides the production of the test scripts, and execution of all testing activities conducted during the Development phase.

Launch Team

The Launch Team will continue meeting regularly during the Development phase to monitor the ongoing progress of the project, address any issues or deviations from the agreed upon schedule or process, and ensure organization-wide visibility into project status. The Launch Team will also create the Product Launch Strategy during this phase.

SourceForge Project

The SourceForge project created prior to beginning the Product Definition phase should now be populated with the materials completed in the Product Definition and Design phases..

SourceForge Component	Contents
At the beginning of the phase	
Project Information	Project-level information such as project description, members, and roles.
Document Manager	A Document Manager folder hierarchy containing final versions of all documents created in the Product Definition and Design phases, including automatically stored revision and review histories. A new folder used to manage the meeting minutes, notes, and other documentation generated by the Launch Team. Supplementary documentation, such as research or other reference material, stored in the Document Manager. The appropriate CDP process templates retrieved from your Software Process Database and stored in your project's Document Manager.
Software Process Database	A Software Process Database used for managing all standard procedures, process templates, checklists, and other materials.
Trackers	Trackers containing all feature requests, bugs, and other artifacts that will be addressed in the product. A Project Logistics Tracker used to track critical estimates, resources, and risks.
Discussion Forums and Mailing Lists	One or more Discussion Forums and Mailing Lists for discussion of project-related topics, including collaborative decision making on design challenges.
SCM Repository	An SCM repository containing any code developed for the product prototype.
Task Manager	One or more projects in the Task Manager to manage various pieces of your Final Project Plan.

SourceForge Component	Contents
By the end of the phase	
Document Manager	Final, approved versions of all required Development phase documents stored in the Document Manager. Completed Software Testing test results stored in the Document Manager. All applicable procedures, process templates, and other materials retrieved from your Software Process Database and stored in the project's Document Manager.
SCM Repository	Completed product code stored in your SCM repository.
File Release System	One or more completed product build/s released to Software Testing in the File Release System.
Tracker / Code Associations	Code commits associated with Tracker artifacts for traceability of all relevant product requirements.
Tracker	Closure of all product requirements Tracker artifacts (with the exception of any items deferred per formal change request.)
Task Manager	One or more actively used Task Manager projects, now including task history for all tasks in the Final Project Plan.

Development Phase Activities and Deliverables

All activities and deliverables are identified by number in the Development Phase Workflow Process Diagram.

Descriptions of each activity or deliverable, suggested contents and acceptance criteria, and dependencies on other activities or deliverables are provided with each item. Activities or deliverables for which peer review or a Software Quality Assurance audit is recommended are identified. Considerations for tailoring deliverables to meet specific organizational needs are indicated where appropriate, and recommended uses for SourceForge are included at the end of each section.

Icons

Activities and deliverables for which templates are provided are identified by the symbol shown here. Please see “Activities and Deliverables” on page 13 for a complete list of all symbols used to identify recommended activities and uses for SourceForge.

3.1. Code (Source and Executable)

Suggested owner: Software Engineering

The primary activity conducted during the Development phase is the creation of the product code. There are many ways to develop software code, the details of which are beyond the scope of the CDP to define. A series of general software engineering best practices applicable across many development methods is included in *Chapter 9, SourceForge Best Practices* on page 245.

Some key points regarding development of code are that it should be developed according to the specifications in the Technical Design Documents, in accordance with the Final Project Plan. It should also be managed in your Software Configuration Management (SCM) repository according to an established SCM policy. (A sample SCM policy is provided.) Periodic engineering review meetings are also recommended throughout the Development and Release Preparation phases.

Developing code is an iterative process, and is generally not considered completed until the product is released. An initial code completion milestone (Step 3.1 in the Development Phase Workflow Process) is followed by a series of testing activities, conducted by both Software Engineering and Software Testing. All issues found during these testing activities are reported either by entering artifacts into a Bug tracker or other tracker, or through more informal collaboration, and submitted to Software Engineering for correction. Final completion of the code will not occur until the Release Preparation phase when all bugs have been either corrected, deferred as per the Product Defect Prioritization Policy or deferred by agreement of the Launch Team.

For this first phase (Step 3.1 in the Development Phase Workflow Process):

Acceptance criteria for the code should include:

- Agreement by each Software Engineer that their code is complete and functional
- Code commits associated with product requirements Tracker artifacts where applicable
- Identification of peer reviewers for Technical Unit Testing code review

Software Quality Assurance Audits

Software Quality Assurance audits are recommended for a number of the components of the code. Audits of processes used to produce the code and activities conducted as a result of the quality of the code are also recommended.

Reference the Software Quality Assurance section of the Product Plan and the Software Quality Assurance audit checklists for each CDP phase for a complete list of all artifacts, processes, and activities to be audited.

Dependencies for the code include:

- Technical Design Documents

Recommended Uses for SourceForge:

Use CVS as your Software Configuration Management (SCM) tool to manage all code developed during the Development and other phases. Use the **SourceForge SCM interface** to browse the contents of your SCM repository.

Associate code commits with product requirements and bugs **Tracker** artifacts where applicable.

Use the **SourceForge Task Manager** to manage the schedule for all Development phase deliverables.

3.2. Technical Unit Testing

Suggested Owner: Software Engineering

Technical Unit Testing is the first round of formal testing activity conducted following the initial code completion milestone. Technical Unit Testing is generally conducted internally by Software Engineering, and involves running the Technical Unit Testing test cases defined in the Technical Design Documents and peer review of each code module.

During the Technical Unit Testing phase, code modules are tested and reviewed individually, in order to ensure that all issues identified are attributable to the specific code being reviewed, and not the result of integration with other modules. Technical Unit Testing is considered "white box" testing, meaning that all code is visible to the testers.

Code review consists of one or more software engineers reviewing and rating another software engineer's code on the following criteria:

- Correctness of code
- Performance
- Documenting or commenting code
- Coding style, including ease of comprehension and of maintenance and extension

Following the code review, the software engineer then makes any necessary corrections to their code. If it was deemed necessary during the first review (based on severity of issues or other criteria), a follow-up peer review is then conducted. This process is repeated until the quality of all code is determined to be satisfactory by the peer review team.

Acceptance criteria for Technical Unit Testing should include:

- Completion of peer review. Code review is recommended. Additional peer review criteria are included in the Code Review Checklist.
- Satisfactory rating of all code modules following peer review

Software Quality Assurance Audits

Software Quality Assurance audits are recommended for a number of the components of Technical Unit Testing. Audits of processes used during Technical Unit Testing and activities conducted as a result of Technical Unit Testin gare also recommended.

Reference the Software Quality Assurance section of the Product Plan and the Software Quality Assurance audit checklists for each CDP phase for a complete list of all artifacts, processes, and activities to be audited.

Dependencies for Technical Unit Testing include:

- Code
- Technical Design Documents
- Possible dependencies on completion of other modules
- Availability of staff for code review

Recommended Uses for SourceForge:

Use CVS as your Software Configuration Management (SCM) tool to manage all code developed during the Development and other phases. Use the **SourceForge SCM interface** to browse the contents of your SCM repository.

Associate code commits with product requirements and bugs **Tracker** artifacts where applicable. Any updates to code associated with a Tracker artifact during the initial code creation stage will automatically be reflected in the history of the Tracker artifact, as will any updates to the artifact itself.

Use the **SourceForge Document Manager** to store the Technical Unit Testing Test Cases (in the Technical Design Documents), the Code Review Template, and the completed Code Review evaluations.

Use the **SourceForge Task Manager** to manage the schedule for all Development phase deliverables.

3.3. Test Cases

Suggested Owner: Software Testing

The goal of Test Cases is to provide a high-level overview of the product components to be tested during Software Testing activities, prior to beginning the actual testing. Test Cases generally consist of a simple description of each area of functionality, performance, or other area that will need to be tested, with some high-level actions to be performed in each test.

Test Cases can be written prior to the beginning of testing, based on the information contained in the Product Requirements Documents and Technical Design Documents.

The Test Cases should include the following minimum information:

- Overview of each area of functionality to be tested
- High-level actions to be performed in each test

Acceptance criteria for Test Cases should include:

- Completion of peer review. The Iterative Document Review and Approval Process is recommended. Additional peer review criteria are included as an appendix to the document template.

Dependencies for Test Cases include:

- Technical Design Documents
- Product Requirements Documents
- Software Test Plan

Software Quality Assurance Audits

Software Quality Assurance audits are recommended for a number of the components of the Test Cases. Audits of processes used to produce various components of the Test Cases and the testing activities conducted as a result of the material in the Test Cases are also recommended.

Reference the Software Quality Assurance section of the Product Plan and the Software Quality Assurance audit checklists for each CDP phase for a complete list of all artifacts, processes, and activities to be audited.

Recommended Uses for SourceForge:

Use the **SourceForge Document Manager** to store the Test Cases at all stages of their development (draft through final) and to manage the Iterative Document Review and Approval Process. See Figure 21 on page 276 for additional detail on this process.

Use the **SourceForge Task Manager** to manage the schedule for all Development phase deliverables.

3.4. Test Scripts

Suggested Owner: Software Testing

The goal of Test Scripts is to document the scenarios to be tested during software testing activities in considerably more detail than the Test Cases, which provide only overview information.

Test Scripts describe in a tabular, or other step-by-step format, each individual action that the tester must take to complete the test scenario, plus the expected outcome of each test. Test Scripts also detail any preconditions that must exist prior to beginning the test, the specific entry point into the test scenario, and any other impacts or considerations.

While Test Cases can be written prior to the beginning of testing, Test Scripts are not started until the beginning of Functional Unit Testing, which is generally the first opportunity for Software Testing to access the product being developed. The level of detail required for the Test Scripts (specific field values, page layout, etc.), is generally not provided in the Product Requirements Documents, Technical Design Documents or other design documents. The Test Scripts are written and refined during Functional Unit Testing, and the final scripts are then used during System Testing and other quality assurance activities throughout the remainder of the development cycle.

Test Scripts should be created to cover testing of the following elements of the product:

Functional Testing

Test scripts that ensure the product functions as described in the Product Requirements Documents and Technical Design Documents.

Performance Testing

Test scripts that measure the performance of the product against standards established in the Software Test Plan.

Environmental Testing

Test scripts that measure the performance of the product on or with various combinations of third party software such as operating systems or databases, and with various hardware configurations. The hardware and software configurations to be tested are defined in the Software Test Plan completed in the Design phase.

Integration Testing

If the product will contain integrations with third party tools or applications, test scripts should be written to confirm that the integrations function as specified in the Product Requirements Documents and Technical Design Documents, and that any Application Programming Interfaces (APIs) function as specified by the third-party tool or application provider.

The Test Scripts should include the following information:

- Process Description
  - A general description of the steps to be followed. Specific actions for each step will be detailed in the Process Flow section.
- Pre-loaded data requirements
  - Any data that must exist within the product prior to completion of the script
- Setup or execution conditions
- Component entry points
  - The specific position in the product from which the component being tested is entered
- Processes to be executed prior to completion of the script
  - List any processes that should be performed before the current process. Each process may be marked mandatory or optional.
- Tables and related fields affected by the test
- Any user interface considerations
- Process Flow
  - For each step indicated in the Process Description, the Process Flow should detail each individual action the tester must take (click, enter data, select, browse, other actions.)
- Expected Results
  - Provide the expected results for each action included in the Process Flow. The expected results should be based on the functionality described in the Technical Design Documents. Expected results will be compared against actual results to assess the pass or fail status of each script.

Acceptance criteria for Test Scripts should include:

- Completion of peer review. The Iterative Document Review and Approval Process is recommended. Additional peer review criteria are included as an appendix to the document template.

Software Quality Assurance Audits

Software Quality Assurance audits are recommended for a number of the components of the Test Scripts. Audits of processes used to produce various components of the Test Scripts and testing activities conducted as a result of the Test Scripts are also recommended.

Reference the Software Quality Assurance section of the Product Plan and the Software Quality Assurance audit checklists for each CDP phase for a complete list of all artifacts, processes, and activities to be audited.

Dependencies for Test Scripts include:

- Test Cases
- User Interaction Specification
- Software Test Plan

Recommended Uses for SourceForge:

Use the **SourceForge Document Manager** to store the Test Scripts at all stages of their development (draft through final) and to manage the Iterative Document Review and Approval Process. See Figure 21 on page 276 for additional detail on this process.

Use the **SourceForge Task Manager** to manage the schedule for all Development phase deliverables.

3.5. Functional Unit Testing

Suggested Owner: Software Testing

After Technical Unit Testing is completed, the code is then delivered to Software Testing for Functional Unit Testing. As with Technical Unit Testing, the goal of Functional Unit Testing is to test each code module individually, in order to ensure that any issues identified are attributable to the code being reviewed, and not related to integration with other modules.

Unlike Technical Unit Testing, however, Functional Unit Testing is considered "black box" testing, meaning that the code is not visible to the testers; only the functionality of the product is being tested. Testing at a strictly functional level exercises the interdependencies between modules, and may result in the need to integrate some or all of the code modules in order to accurately test each module's functionality. While the goal of Functional Unit Testing is to test each module independently, this may not always be possible, and a partially or fully integrated product build may be required.

Functional Unit Testing is generally the point at which Software Testing and Software Engineering begin using the SourceForge Tracker as a means to record and communicate the status of product defects. Software Testing executes each test script during Functional Unit Testing, and if at any point the outcome of a specific action is not as predicted, the test is considered failed and the issue is entered into the Bugs or other tracker for resolution by Software Engineering. This process continues until all test scripts have been completed successfully. If the product will support more than one reference platform (configuration of Operating System, Database, etc., as specified in the Product Requirements Documents), all Functional Unit Testing Test Scripts are run until completed successfully on all reference platforms.

Since the product is not fully integrated at this point, there are generally no specific exit criteria surrounding outstanding product defects at given priority levels. Throughout Functional Unit Testing, however, Software Engineering continues to resolve as many product defects as possible prior to proceeding to Engineering System Testing.

Acceptance criteria for Functional Unit Testing should include:

- All Functional Unit Testing Test Scripts completed successfully on all reference platforms
- Functional Unit Testing results stored in the Document Manager
- Bugs identified during Functional Unit Testing stored in the Tracker.

Dependencies for Functional Unit Testing include:

- Code
- Technical Unit Testing
- Test Scripts

Software Quality Assurance Audits

Software Quality Assurance audits are recommended for a number of the components of Functional Unit Testing. Audits of processes used to during Functional Unit Testing and activities conducted as a result of Functional Unit Testing are also recommended.

Reference the Software Quality Assurance section of the Product Plan and the Software Quality Assurance audit checklists for each CDP phase for a complete list of all artifacts, processes, and activities to be audited.

Recommended Uses for SourceForge:

Use CVS as your Software Configuration Management (SCM) tool to manage all code developed during the Development and other phases. Use the **SourceForge SCM interface** to browse the contents of your SCM repository.

Associate code commits with product requirements and bugs **Tracker** artifacts where applicable. Any updates to code associated with a Tracker artifact during the initial code creation stage will automatically be reflected in the history of the Tracker artifact, as will any updates to the artifact itself.

Use the **SourceForge Tracker** to manage all product defects identified during Functional Unit Testing.

Use the **SourceForge Document Manager** to store the Test Scripts to be conducted during Functional Unit Testing, and the completed Functional Unit Testing results.

Use the **SourceForge Task Manager** to manage the schedule for all Development phase deliverables.

3.6. Engineering System Testing

Suggested Owner: Software Engineering

After all module-specific Functional Unit Testing is successfully completed, the individual code modules are then ready to be integrated to form the first build of the complete product. Once the product is built, it is installed on a server or other central location. (Testing prior to this point is most often performed on individual software engineers' development machines.)

During Engineering System Testing, each software engineer is responsible for testing their own code to ensure it is still functional within the context of the newly integrated product. Engineering System Testing may be conducted exclusively within the Software Engineering group, or may be opened to a broader audience of reviewers internal to the organization, such as Software Product Management or other key stakeholders.

As with Technical Unit Testing, Engineering System Testing is often less formally structured than testing conducted by the Software Testing group. The SourceForge Tracker should continue to be used to manage product defects at this stage. All issues identified during code review should be entered into the Bugs or other tracker and managed to completion.

Acceptance criteria for Engineering System Testing should include:

- Satisfactory rating of all integrated code modules by Software Engineering
- No outstanding high priority bugs as per the Product Defect Prioritization Policy
- Ready to freeze code base and user interface
- Ready for Alpha System Testing

Dependencies for Engineering System Testing include:

- Functional Unit Testing

Software Quality Assurance Audits

Software Quality Assurance audits are recommended for a number of the components of Engineering System Testing. Audits of processes used during Engineering System Testing and activities conducted as a result of Engineering System Testing are also recommended.

Reference the Software Quality Assurance section of the Product Plan and the Software Quality Assurance audit checklists for each CDP phase for a complete list of all artifacts, processes, and activities to be audited.

Recommended Uses for SourceForge:

Use CVS as your Software Configuration Management (SCM) tool to manage all code developed during the Development and other phases. Use the **SourceForge SCM interface** to browse the contents of your SCM repository.

Associate code commits with product requirements and bugs **Tracker** artifaacts where applicable. Any updates to code associated with a Tracker artifact during the initial code creation stage will automatically be reflected in the history of the Tracker artifact, as will any updates to the artifact itself.

Use the **SourceForge Tracker** to manage some or all product defects identified during Engineering System Testing.

Use the **SourceForge Task Manager** to manage the schedule for all Development phase deliverables.

3.7. Frozen User Interface

Suggested Owner: Software Engineering

Once Engineering System Testing is successfully completed, all product components, including the code, build scripts, graphics, and other files developed to render the user interface, should be considered complete. At this point, a Change Control Policy is initiated and no further changes are made to the user interface code, graphics, or other files, except to correct product defects, or in response to formal change requests. A sample Change Control Policy is provided.

Acceptance criteria for a Frozen User Interface should include:

- All code and designs detailed in the User Interaction Specification fully incorporated into product code base
- Launch Team agrees there are no outstanding features to implement
- Change Control Policy initiated

Dependencies for a Frozen User Interface include:

- User Interaction Specification
- Engineering System Testing

Software Quality Assurance Audits

Software Quality Assurance audits are recommended for a number of artifacts after they are placed under the Change Control Policy. Audits of change control processes activities conducted following formal change requests are also recommended.

Reference the Software Quality Assurance section of the Product Plan and the Software Quality Assurance audit checklists for each CDP phase for a complete list of all artifacts, processes, and activities to be audited.

Recommended Uses for SourceForge:

Use CVS as your Software Configuration Management (SCM) tool to manage all code developed during the Development and other phases. Use the **SourceForge SCM interface** to browse the contents of your SCM repository. When the code base and user interface have been frozen, tag the code as per your SCM Policy.

Use the **SourceForge File Release System** to release product builds to Software Testing or other audiences.

Use the **SourceForge Task Manager** to manage the schedule for all Development phase deliverables.

3.8. Frozen Code Base

Suggested Owner: Software Engineering

Once Engineering System Testing is successfully completed, all code should be considered complete. At this point, a Change Control Policy is initiated and no further changes are made to any portion of the code except to correct product defects, or in response to formal change requests. A sample Change Control Policy is provided.

Product builds should be released to Software Testing using the SourceForge File Release System.

Acceptance criteria for a Frozen Code Base should include:

- Ready for Alpha System Testing (The product meets minimum performance criteria and contains no outstanding high priority bugs as per the Product Defect Prioritization Policy.)
- Launch Team agrees there are no outstanding features to implement
- Product builds released to Software Testing

Dependencies for the Frozen Code Base include:

- Engineering System Testing
- Frozen User Interface

Software Quality Assurance Audits

Software Quality Assurance audits are recommended for a number of the artifacts once placed under the Change Control Policy. Audits of change control processes and activities conducted following formal change requests are also recommended.

Reference the Software Quality Assurance section of the Product Plan and the Software Quality Assurance audit checklists for each CDP phase for a complete list of all artifacts, processes, and activities to be audited.

Recommended Uses for SourceForge:

Use CVS as your Software Configuration Management (SCM) tool to manage all code developed during the Development and other phases. Use the **SourceForge SCM interface** to browse the contents of your SCM repository. When the code base has been frozen, tag the code as per your SCM Policy.

Use the **SourceForge File Release System** to release product builds to Software Testing or other audiences.

Use the **SourceForge Task Manager** to manage the schedule for all Development phase deliverables.

3.9. Migration Scripts

Suggested Owner: Software Engineering

Migration scripts, or upgrade scripts, are developed to enable users to easily upgrade from an earlier version to the new version of a product. Migration scripts should also enable users to migrate any data stored using a previous version, including any custom settings or configurations. Providing migration scripts with any new version of a product is critical to facilitating customer adoption and minimizes the costs , downtime, and risk of data loss often associated with a manual upgrade. If your Beta Testing phase will involve current customers external to your organization, it is important that migration scripts be available prior to the beginning of Beta Testing.

Migration scripts may be developed in one of several ways based on resource availablity and allocation. If the resources are available, a desirable situation is to have all migration scripts allocated exclusively to a single software engineer, or if migration scripts are complex, to a single software engineering team. A database administrator may write and maintain all migration scripts associated with any database schema changes, and/or a release engineer may write and maintain all migration scripts at the file system level. The migration scripts can be developed along with the rest of the product code, and be subject to the same level of quality assurance activities as the main body of code. Any defects identified during testing can then be assigned to the dedicated software engineers for resolution.

Often however, resources are more constrained and dedicated software engineers are not available for end-to-end ownership of all migration scripts. In such cases, each software engineer may develop the migration scripts for their own functional areas, and be responsible for resolving any defects identified during testing. It is preferable to develop the migration scripts along with the rest of the product code, in order to maximize the time available for testing.

For software testing activities, it is highly recommended that two separate testing environments be set up, one with a newly installed version of the product and the second with a version installed using the migration scripts to upgrade from a previous version. In this way, any defects identified in the migrated version can be compared with the newly installed version to verify whether the defect is in the migration scripts or the main body of code.

Acceptance criteria for Migration Scripts include:

- Migrated version of the product passes all test scripts executed during Alpha System Testing

Dependencies for Migration Scripts include:

- Technical Design Documents
- Code

Recommended Uses for SourceForge:

Use CVS as your Software Configuration Management (SCM) tool to manage all code, including code for migration scripts and other utilities, developed during the Development and other phases. Use the **SourceForge SCM interface** to browse the contents of your SCM repository.

Associate code commits with product requirements and bugs **Tracker** artifacts where applicable.

Use the **SourceForge Task Manager** to manage the schedule for all Development phase deliverables.

Use the **SourceForge File Release System** to release migration scripts to Software Testing or other audiences.

3.10. Alpha System Testing

Suggested Owner: Software Testing

Following successful completion of Engineering System Testing, the product is released to Software Testing for Alpha System Testing. During Alpha System Testing, Software Testing re-executes all test scripts that were executed and refined during Functional Unit Testing, this time against the fully integrated product.

Use of the SourceForge Tracker is generally resumed during Alpha System Testing to record and communicate the status of all product defects or bugs. Software Engineering continues to be actively engaged in correcting bugs, and stricter exit criteria are applied to this testing cycle than were applied to the previous Technical Unit Testing, Functional Unit Testing, and Engineering System Testing. The process of testing, identifying and correcting bugs, and re-testing continues until the exit criteria (generally specifying no high priority bugs as defined in the Product Defect Prioritization Policy), are met.

The goal of Alpha System Testing is to produce a product of high enough quality to release to an audience of Beta Testing participants external to the organization. Alpha System Testing is sometimes opened to an audience of participants external to the organization, however given the considerable effort required to document and support the program, Alpha System Testing is generally limited to participants internal to the organization.

A recommended practice is to use the Alpha System Testing period to engage a broad, internal audience of stakeholders responsible for supporting activities and materials, such as Marketing, Training, and Documentation, as well as stakeholders who will be using, selling, and supporting the product following its release.

Acceptance criteria for Alpha System Testing should include:

- No outstanding high priority bugs, as defined in the Product Defect Prioritization Policy
- Readiness for release to external Beta Testing customers
- Outstanding issues detailed in Beta Release Notes
- Formal agreement by key stakeholders to proceed to Beta Testing

Dependencies for Alpha System Testing include:

- Engineering System Testing
- Test Scripts
- Migration Scripts

Software Quality Assurance Audits

Software Quality Assurance audits are recommended for a number of the components of Alpha System Testing. Audits of processes used during Alpha System Testing and activities conducted as a result of Alpha System Testing are also recommended.

Reference the Software Quality Assurance section of the Product Plan and the Software Quality Assurance audit checklists for each CDP phase for a complete list of all artifacts, processes, and activities to be audited.

Recommended Uses for SourceForge:

Use CVS as your Software Configuration Management (SCM) tool to manage all code developed during the Development and other phases. Use the **SourceForge SCM interface** to browse the contents of your SCM repository.

Associate code commits with product requirements and bugs **Tracker** artifacts where applicable. Any updates to code associated with a Tracker artifact during the initial code creation stage will automatically be reflected in the history of the Tracker artifact, as will any updates to the artifact.

Use the **SourceForge Tracker** to manage all product defects identified during Alpha System Testing. Separate trackers can be used if users external to the organization are participating in Alpha System Testing.

Use the **SourceForge Task Manager** to manage the schedule for all Development phase deliverables.

Use the **SourceForge Document Manager** to store the Test Scripts to be conducted during Alpha System Testing and the completed test results.

Use the **SourceForge File Release System** to release product builds to Software Testing or other audiences.

3.11. Draft User Documentation

Suggested Owner: Software Product Management

A draft of all product user documentation should be completed during the Development phase. While the product is not yet in its final form, the completion of Engineering System Testing should result in a product build with the code and user interface sufficiently frozen (with the exception of bug fixes) to enable the technical writer to adequately document the functionality of the product. If your Beta Testing phase will involve participants external to your organization, it is important that draft user documentation be available prior to the beginning of Beta Testing for distribution to participants along with the product.

It is expected that updates will need to be made following the Iterative Document Review and Approval Process, and as the result of any product or documentation issues raised duing the Alpha and Beta Testing periods. You can create a Tracker to manage defects and requests generated as a result of these review activities.

Acceptance criteria for Draft User Documentation should include:

- Completion of peer review. The Iterative Document Review and Approval Process is recommended.
- All draft chapters and other documentation completed according to the Documentation Plan

Dependencies for Draft User Documentation include:

- Documentation Plan
- Product Requirements Documents
- Technical Design Documents

Recommended Uses for SourceForge:

Use the **SourceForge Document Manager** to store the user documentation at all stages of its development (draft through final) and to manage the Iterative Document Review and Approval Process. See Figure 21 on page 276 for additional detail on this process.

Use the **SourceForge Task Manager** to manage the schedule for all Development phase deliverables.

Use the **SourceForge Tracker** to manage documentation issues reported during the draft user documentation review cycles.

3.12. Product Training Plan

Suggested Owner: Software Product Management

The Product Training Plan is intended to provide a detailed description of the product training that will provided to internal staff on the use, sale, and support of the finished product. For additional information on the processes for managing project and organizational training, please see "Managing your Organization's Training Programs using SourceForge" on page 260.

The Product Training Plan should include:

- Scheduled times, locations, and other logistical information
- Overview description of all training activities
  - Type of training (on-site training session, Webinar or other online training sessions, conference call, or other types of training.)
  - Content and scope (full product training, update on new features, preliminary product roadmap update.)
  - Intended audience (sales staff, support staff, external customers, or other audiences.)
- Agenda
  - Topic, goals, presenters, and time allotted for each training segment
- Proposed attendees
- Requirements for attendees
  - Minimum technical skill set, reading course materials prior to the training session, completing prior training sessions, or other prerequisite knowledge or preparatory work.
- Requirements for presenters
  - Preparing presentation and demonstration materials, attending practice sessions or other meetings prior to the training session, availability for follow-up support.
- Product requirements
  - If the product will be demonstrated, indicate any product-related requirements such as availability (hosted on a server, available for presenters only, CDs needed to distribute to attendees) quality (Alpha, Beta, other), other hardware, software, or infrastructure requirements.

- Training materials
  - Describe any new or updated training materials that will need to be produced, including specific content and owners.
- Logistical requirements
  - Include any preparatory activities required of groups such as Facilities or IT. Include necessary resources such as hardware, network access, training rooms, and other resources.
- Testing
  - Include any plans for testing attendees to ensure knowledge retention.
- Follow up strategy
  - Include the strategy for continuing education. This may include preparing Frequently Asked Questions (FAQ) documents, providing a mechanism for staff to ask questions after the conclusion of the training sessions, Training Evaluation Forms, or other follow-up activities.
- Project Plan
  - Include a project plan listing dates, owners, and dependencies for all tasks and deliverables required to prepare for the training activities.

Upon completion of the Product Training Plan, the Task Manager should be used to manage all required activities and deliverables. You may choose to update your Final Project Plan and use the Task Manager project already in use, or you may create a new project or task folder to manage only training-specific items.

Acceptance criteria for the Product Training Plan should include:

- Completion of peer review. The Iterative Document Review and Approval Process is recommended.
- All training-related activities and deliverables incorporated into the Project Plan and managed using the Task Manager.

Dependencies for the Product Training Plan include:

- Product Plan
- Product Requirements Documents

Recommended Uses for SourceForge:

Use the **SourceForge Document Manager** to store the Product Training Plan at all stages of its development (draft through final) and to manage the Iterative Document Review and Approval Process. See Figure 21 on page 276 for additional detail on this process.

Use the **SourceForge Task Manager** to manage any additional dates added to the Final Project Plan as a result of the Product Training Plan, or to manage a new project plan specific to the training activities.

3.13. Product Launch Strategy

Suggested Owner: Software Product Management

The Product Launch Strategy is intended to describe all key actions and deliverables associated with the launch of the product. For products that will be sold, or otherwise marketed for internal or external use, the Product Launch Strategy should include key manufacturing, marketing, and sales activities, as well as product-related items such as delivery dates and methods. Any changes to the pricing and licensing model should also be included. Marketing, manufacturing, and sales-related activities should be described at a high level only; more significant detail will be included in the Marketing and Channel Plan completed in the Release Preparation phase. Post-release activities to measure the success of the product launch should also be included.

For products that will be used as components in larger products, or are intended for internal use without the need for complex packaging or distribution methods, the Product Launch Strategy should include only applicable components..

As the Product Launch Strategy is cross-functional in nature, it is recommended that the Launch Team be responsible for its creation.

The Product Launch Strategy can include:

Manufacturing Activities

Describe the strategy for packaging and distributing the product.

Marketing Activities

Include any press releases, tradeshows, print and online advertising, speaking engagements, Website updates, promotions, and other marketing activities related to the launch of the product. This is a strategy document only; all marketing activities will be described in more detail in the Marketing and Channel Plan developed during the Release Preparation phase.

Sales Activities

Aside from training, include any sales activities that will take place, such as upgrade campaigns, reseller or channel events, sales incentives, or other promotions.

Pricing and Licensing

Describe the strategy for pricing and licensing the product. If there are any changes to the pricing or licensing model from a previous version, include the justification and potential impacts. If applicable, describe the transition plan from the previous model. This is a strategy document only; the price list and other more detailed information may be produced in the Release Preparation phase.

Post-Release Activities

Describe any post-release activities that will be conducted to measure the success of the product launch. See “6.1. Internal Post-Release Analysis” on page 225 for additional information on suggested activities.

When the Product Launch Strategy has been completed, the Task Manager should be used to manage all required activities and deliverables. You may choose to update your Final Project Plan and use the Task Manager project already in use, or you may create a new project or task folder to manage only launch-specific items.

Acceptance criteria for the Product Launch Strategy should include:

- Completion of peer review. The Iterative Document Review and Approval Process is recommended.
- All product launch-related activities and deliverables incorporated into the Project Plan and managed using the Task Manager.

Dependencies for Product Launch Strategy include:

- Product Plan
- Product Requirements Documents
- Final Project Plan
- Launch Team

Recommended Uses for SourceForge:

Use the **SourceForge Document Manager** to store the Product Launch Strategy at all stages of its development (draft through final) and to manage the Iterative Document Review and Approval Process. See Figure 21 on page 276 for additional detail on this process.

Use the **SourceForge Task Manager** to manage any additional dates added to the Final Project Plan as a result of the Product Launch Strategy.

Development Phase Exit Criteria

During the Development phase, members of all stakeholder groups will have worked closely to ensure that the code for the product has been developed, tested, and is ready for Beta Testing.

A Development Phase Exit Review meeting should be held with key stakeholders from all impacted organizations. The following checklist should be used to ensure all exit criteria are met.

✓	Development Phase Exit Checklist
❑	SourceForge project now containing all elements specified on page 128.
❑	Code, including user interface, completed, frozen, and archived in your SCM repository.
❑	Change Control Policy initiated.
❑	Test Cases completed, completed, peer-reviewed, and approved by required approvers. Final versions stored in the Document Manager and locked.
❑	Test Scripts completed, peer-reviewed, and approved by required approvers. Final versions stored in the Document Manager and locked.
❑	Migration scripts completed and submitted for testing with the product code.
❑	Technical Unit Testing, Functional Unit Testing, Engineering System Testing, and Alpha System Testing completed with no high priority bugs outstanding.
❑	Product ready to enter Beta Testing.
❑	Draft User Documentation completed according to Documentation Plan and submitted for peer review. Draft User Documentation stored in the Document Manager.
❑	Product Training Plan completed, peer-reviewed, and approved by required approvers. Final version stored in the Document Manager.
❑	Actions and deliverables from the Product Training Plan input into the Task Manager.
❑	Product Launch Strategy completed, peer-reviewed, and approved by required approvers. Final version stored in the Document Manager.
❑	All project training activities scheduled for completion in the Development phase completed.
❑	All Training Evaluation Forms completed and posted to the Organizational Training project on SourceForge.

✓	Development Phase Exit Checklist
❑	Development phase Software Quality Assurance audit checklist completed. Checklist posted to the Document Manager.
❑	Periodic engineering review meetings conducted.
❑	Regular Launch Team meetings conducted, including reviews of the Task Manager and the Project Logistics Tracker.
❑	Periodic reviews of subcontractor progress completed, if applicable.
❑	Development Phase Exit Review successfully completed using this checklist. Checklist posted to the Document Manager.

Using SourceForge to Support the Development Phase

SourceForge supports the goals of the Development Phase by allowing you to:

- Use CVS as your Software Configuration Management (SCM) tool to manage all source code. Use the **SourceForge SCM interface** to browse the contents of your SCM repository.
- Associate code commits with product requirements Tracker artifacts using **Tracker / Code Associations.**
- Release product builds to Software Testing and other audiences using the **File Release System.**
- Manage product defects and enhancement requests using the **Tracker.**
- Enable management visibility and control of work being done internally and by subcontractors using the **Task Manager.**
- Manage the document creation and approval process using the **Document Manager.**
- See *SourceForge and the Capability Maturity Model for Software, A Supplement to the SourceForge Collaborative Development Process* for additional detail on using SourceForge and the CDP to satisfy SW-CMM Level 2 and 3 requirements.

Development Phase Templates

The following table provides a quick reference to all templates used for the Development phase deliverables and activities.

All templates are provided on the CD included with this manual, and at the following URL: *https://www.vasoftware.com/cdp/*

Table 17. Development Phase Templates

Ref#	Template	File Location
3.3.1	Test Case	\dev\[3.3.1]testcase.dot \dev\[3.3.1]testcase.pdf
3.4.1	Test Script	\dev\[3.4.1]testscript.dot \dev\[3.4.1]testscript.pdf
3.12.1	Product Training Plan	\dev\[3.12.1]training.dot \dev\[3.12.1]training.pdf
3.13.1	Product Launch Strategy	\dev\[3.13.1]launch.dot \dev\[3.13.1]launch.pdf
3.14.1	SQA Checklist Development Phase	\dev\[3.14.1]sqadev.dot \dev\[3.14.1]sqadev.pdf
3.15.1	Development Phase Exit Checklist	\dev\[3.15.1]devexit.dot \dev\[3.15.1]devexit.pdf

Please reference “Numbering” on page 10 for additional information on document numbering conventions.

CHAPTER 6

Release Preparation Phase

This chapter includes the following information:

- Release Preparation Phase Overview
- Release Preparation Phase Workflow Process Diagram
- Release Preparation Phase Inputs
- Release Preparation Phase Activities and Deliverables
- Release Preparation Phase Exit Criteria
- Using SourceForge to Support the Release Preparation Phase
- Release Preparation Phase Templates

Release Preparation Phase Overview

The primary goal of the Release Preparation phase is to complete all product testing, documentation, product marketing, and other supporting activities necessary to ensure a successful product release. The product code remains subject to the change control policy initiated in the Development phase, and the Software Engineering and Quality Assurance organizations focus exclusively on correcting product defects identified during Beta Testing and other testing activities.

To enter the Release Preparation phase, the product must have successfully passed Alpha System Testing and be ready to enter a Beta Testing period potentially involving participants external to the organization. The Product Training Plan and Product Launch Strategy are also critical, as the activities defined within will be executed during the Release Preparation and future phases.

Activities conducted in the Release Preparation phase include:

- Conducting Beta Testing
- Completing all user documentation
- Completing Service, Support, Marketing and Channel plans for pre- and post-release activities
- Conducting product training
- Completing product packaging and collateral materials
- Conducting Final System Testing
- Conducting Release Readiness Checkpoint

At the conclusion of this phase, a final readiness checkpoint will be conducted to ensure that all activities conducted in the Release Preparation and previous phases have been satisfactorily completed. Additionally, the final release checkpoint will ensure that any outstanding issues have been addressed and that project stakeholders are in agreement regarding the acceptable quality of the product and the readiness of the organization for the product's release. Immediately upon completion of the Release Preparation phase, the product will enter the Release phase and be released to its intended audience; therefore, all activities required to ensure optimal product quality must be completed during this phase.

The following diagram indicates the activities and deliverables necessary to exit the Release Preparation phase. The suggested contents of each activity and deliverable and the recommended process for producing them are defined in the following sections.

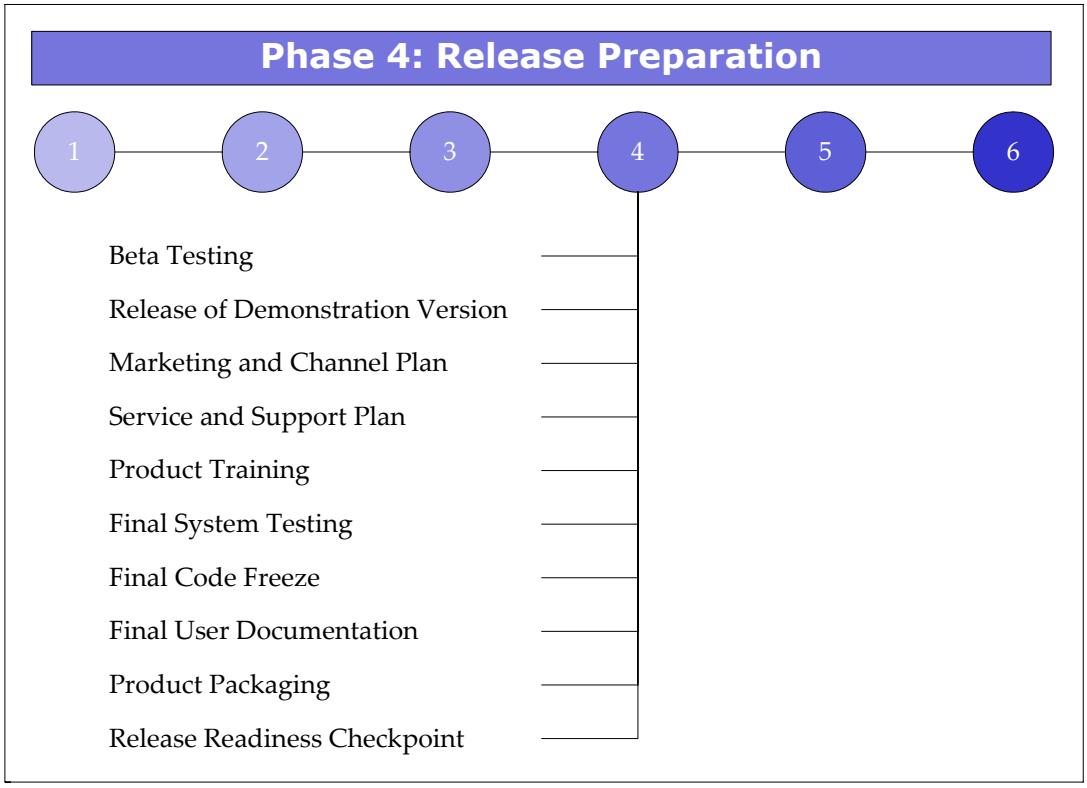


Figure 12. Release Preparation Phase Deliverables

The Release Preparation Phase Workflow Process

The following diagram illustrates the recommended best practice workflow process for the Release Preparation phase.

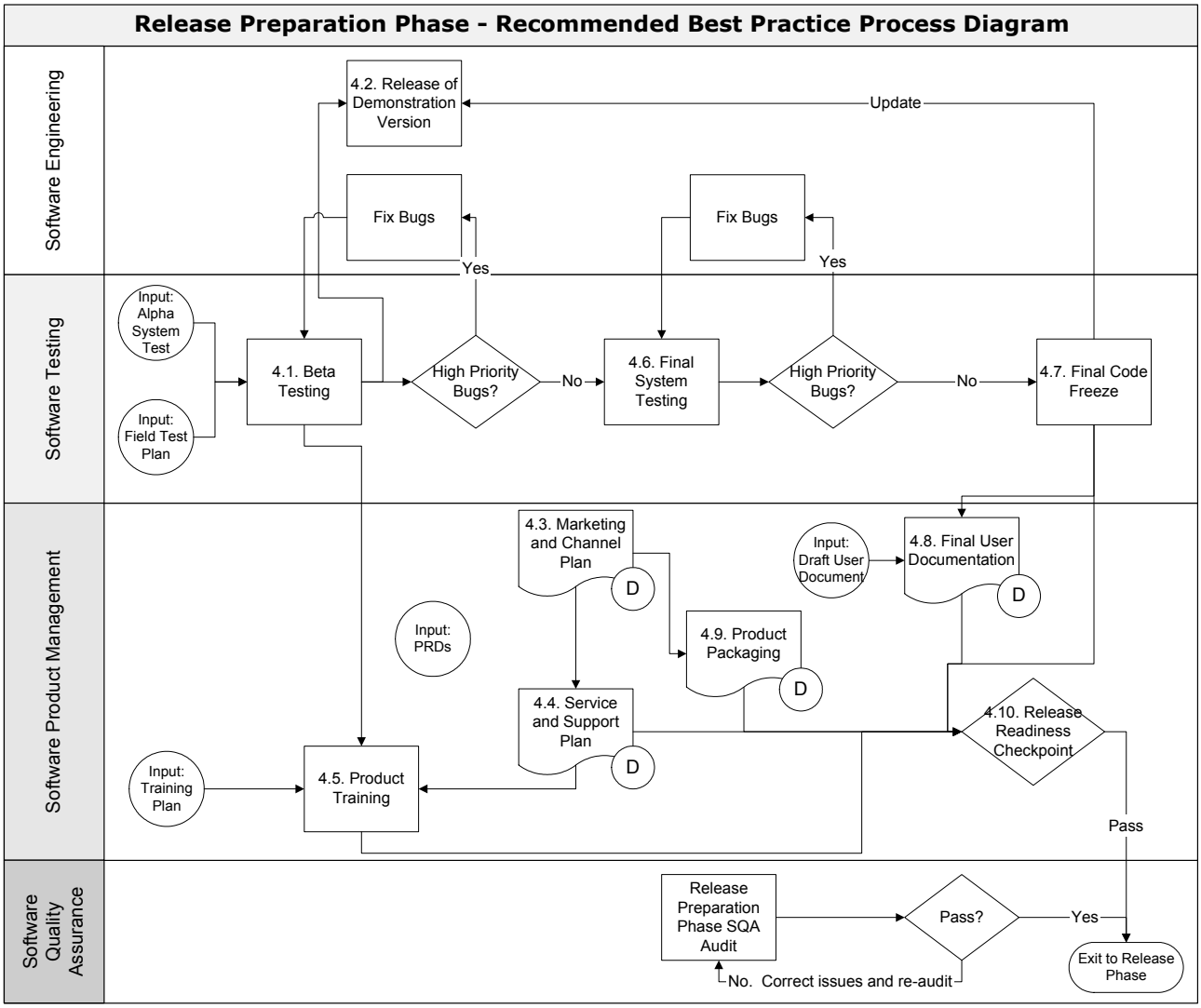


Figure 13. Release Preparation Phase Workflow Process

Please see Please see "Workflow Processes" on page 11 for a full description of all flowchart notation.

Release Preparation Phase Inputs

The following inputs are required to enter the Release Preparation phase:

Completion of Alpha System Testing

Completion of Alpha System Testing is the key prerequisite for entering Beta Testing in the Release Preparation phase. Upon entering Beta Testing, code must be of sufficiently high quality to release to an audience of Beta participants external to the organization, whether or not an external Beta program will be conducted.

Product Training Plan

The Product Training Plan will be followed duing the Release Preparation phase to conduct all product-related training activities.

Draft User Documentation

The User Documentation completed in draft form in the Development phase will be reviewed, revised, and completed during the Release Preparation phase.

Product Launch Strategy

The Product Launch Strategy will be followed during the Release Preparation phase to prepare all functional areas of the organization for the product release. The Product Launch Strategy will also be used in the Release and Sustaining Engineering phases.

Product Plan, Product Requirements Documents, and Technical Design Documents

The Business Requirements Document , Product Requirements Documents, and Technical Design Documents are referenced in the Release Preparation phase when creating detailed plans for supporting and marketing the product.

SourceForge Project

The SourceForge project created prior to beginning the Product Definition phase should now be populated with the materials completed in the Product Definition, Design, and Development phases.

SourceForge Component	Contents
At the beginning of the phase	
Project Information	Project-level information such as project description, members, and roles.
Document Manager	A Document Manager folder hierarchy containing final versions of all documents created in the Product Definition, Design, and Development phases, including automatically stored revision and review histories. The Launch Team folder now containing all meeting minutes, notes, and other documentation generated to date. Any formal change requests submitted following Code Freeze. Supplementary documentation, such as research or other reference material. The appropriate CDP process templates retrieved from your Software Process Database and stored in your project's Document Manager.
Software Process Database	A Software Process Database used for managing all standard procedures, process templates, checklists, and other materials.
Trackers	Trackers containing all feature requests, bugs, and other artifacts that will be addressed in the product. A Project Logistics Tracker used to track critical estimates, resources, and risks. Trackers containing all product defects identified during Development phase testing activities. Tracker artifacts linked to code modules where appropriate. Closure of all product requirements Tracker artifacts (with the exception of any items deferred per formal change request.)
Discussion Forums and Mailing Lists	One or more Discussion Forums and Mailing Lists for discussion of project-related topics, including collaborative decision making on design challenges.
SCM Repository	An SCM repository containing all code developed during the Development phase, including migration scripts. Code modules linked with Tracker artifacts where appropriate.
File Release System	One or more completed product build/s released to Software Testing in the File Release System.

SourceForge Component	Contents
Tracker / Code Associations	Code commits associated with Tracker artifacts for traceability of all relevant product requirements.
Task Manager	One or more projects in the Task Manager to manage various pieces of your Final Project Plan, including any activities and deliverables added upon completion of the Product Training Plan and Product Launch Strategy.

SourceForge Component	Contents
By the end of the phase	
Document Manager	Final, approved versions of all required Release Preparation phase documents stored in the Document Manager All applicable procedures, process templates, and other materials retrieved from your Software Process Database and stored in the project's Document Manager.
Task Manager	One or more projects in the Task Manager to manage various pieces of your Final Project Plan, including any activities and deliverables added upon completion of the Marketing and Channel Plan and Service and Support Plan. All Task Manager tasks scheduled for completion prior to product release now marked as completed.
SCM Repository	SCM repository containing completed, frozen, and fully-tested code. The contents of the SCM repository should be archived in case of system failure.
File Release System	Final product builds ready to release to their intended audience in the File Release System.

Release Preparation Phase Activities and Deliverables

All activities and deliverables are identified by number in the Release Preparation Phase Workflow Process Diagram.

Descriptions of each activity or deliverable, suggested contents and acceptance criteria, and dependencies on other activities or deliverables are provided with each item. Activities or deliverables for which peer review or a Software Quality Assurance audit is recommended are identified. Considerations for tailoring deliverables to meet specific organizational needs are indicated where appropriate, and recommended uses for SourceForge are included at the end of each section.

Icons

Activities and deliverables for which templates are provided are identified by the symbol shown here. Please see “Activities and Deliverables” on page 13 for a complete list of all symbols used to identify recommended activities and uses for SourceForge.

4.1. Beta Testing

Suggested owner: Software Engineering, Software Testing, or Software Product Management

During the Release Preparation phase, Beta Testing is executed as defined in the Field Test Plan. Beta Testing is one of the key activities conducted during this phase, as it provides the organization with the first opportunity to solicit input from a potential customer audience on the quality and value of the product. Participants may be internal or external to your organization, based on the intended audience of the product.

If participants are carefully selected and sufficient time is allowed to gather and respond to participant input, the Beta Testing program can provide an extremely valuable source of input, and can enable your organization to considerably refine and improve the quality of the product prior to its release. If the product is intended for sale outside the organization, the Beta Testing program can provide the opportunity to generate success stories and positive customer feedback for potential use by Marketing in developing press releases and other product marketing collateral. Technical Support and Professional Services staff can begin using the product to prepare for supporting it after its release. Partners and other third parties can use the Beta release to prepare and validate any products they are developing that integrate or work with your product. A Beta period can also support the preparation of documentation and collateral materials where product screen shots may be required.

Throughout the Beta program, product code remains subject to the change control policy initiated in the Development phase; therefore, changes to the code may only be made in accordance with this policy. Any proposed changes to the user interface should be considered carefully from this point on, as they can negatively impact test scripts and the production of documentation and collateral materials. Changes to the user interface late in the Collaborative Development Process can be extremely costly in terms of both time and resources, and should be avoided wherever possible.

Software Engineering remains actively engaged throughout the Beta period in correcting bugs, and Software Testing may continue to run Test Scripts or conduct other software testing activities, in addition to monitoring input generated by Beta participants.

The Field Test Plan, developed during the Design phase, will have defined the goals, management process, and logistical considerations such as start and end dates, product delivery method, and participant selection criteria for the Beta Testing program. Prior to beginning the Beta Testing program, identification of participants and preparation of the product and supporting materials must be completed as defined in the Field Test Plan.

Entry criteria for Beta Testing should include:

- Selection of qualified participants
- Frozen code base and completion of Alpha System Testing
- Creation of a Beta program-specific tracker to manage reported product defects
- Establishment and documentation of a Beta program support process
- Action plan for addressing reported product defects and feature requests
- Draft User Documentation
- Trained Beta support staff
- Beta Release Notes or other documentation providing participants with:
  - Software installation, upgrade, or access instructions
  - Processes for reporting product defects and other feedback
  - Commitments on the part of the organization for responding to reported issues
  - Significant known issues with the product
  - Any planned updates to the beta product, i.e. "beta refresh"
  - Requirements of participation
  - Requirements upon completion of the program (uninstalling the software, continuation or discontinuation of support, deactivation of login information, etc.)
  - Any other considerations

Acceptance criteria for Beta Testing should include:

- All beta feedback processed (product defects corrected or deferred, feature requests reviewed for inclusion in the current or a future release)
- No outstanding high priority bugs, as defined in the Product Defect Prioritization Policy
- Formal closure of the Beta program with participants as defined in the Field Test Plan

Dependencies for Beta Testing include:

- Alpha System Testing
- Field Test Plan
- Draft User Documentation
- Migration Scripts (if participants will be upgrading from an earlier version of the product.)

Software Quality Assurance Audits

Software Quality Assurance audits are recommended for a number of the components of Beta Testing. Audits of processes used during Beta Testing and activities conducted as a result of Beta Testing are also recommended.

Reference the Software Quality Assurance section of the Product Plan and the Software Quality Assurance audit checklists for each CDP phase for a complete list of all artifacts, processes, and activities to be audited.

Recommended Uses for SourceForge:

Use CVS as your Software Configuration Management (SCM) tool to manage all code developed during the Release Preparation and other phases, including product defects corrected during the Beta Testing period. Use the **SourceForge SCM interface** to browse the contents of your SCM repository.

Associate code commits with product requirements and bugs **Tracker** artifacts where applicable. Any updates to code associated with a Tracker artifact during the initial code creation stage will automatically be reflected in the history of the Tracker artifact, as will any updates to the artifact itself.

Use the **SourceForge Tracker** to manage all product defects identified during Beta Testing.

Use the **SourceForge Task Manager** to manage the schedule for all Product Release Preparation phase deliverables.

Use the **SourceForge File Release System** to release product builds to Software Testing, Beta program participants, or other audiences.

Use **SourceForge News** to announce the beginning of Beta Testing, any interim updates, and the conclusion of Beta Testing.

4.2. Release of Demonstration Version

Suggested Owner: Software Engineering, Software Product Management

Once a sufficient level of product quality is achieved, it is often desirable to prepare a demonstration version of the product to enable sales and other staff to demonstrate the new product to potential customers. A pre-release demonstration version can also facilitate pre-release Marketing activities such as tradeshows and press or analyst events, and can provide a valuable resource when conducting Product Training.

Depending on the nature of the product, the organization of your Beta program, and other considerations, you may wish to use your Beta product for demonstration purposes. You may choose to distribute or provide access to, the same product version that your Beta participants are accessing. This will allow you to leverage the work already completed for the Beta program, and simplify the process of releasing a demonstration version.

In some cases, you may wish to host or distribute a demonstration version into which some amount of sample data has been entered, in order to provide a more meaningful demonstration of the product's capabilities. You may also wish to create a pre-defined demonstration path to ensure that all staff are providing similar, high quality demonstrations to potential customers and other audiences. It is also important to provide appropriate training to staff who will conduct demonstrations.

Regardless of how the demonstration version is offered, it is important that the demonstration version is of sufficiently high quality to ensure a positive representation of the product's capabilities when demonstrating to potential customers and other audiences external to the organization, and when conducting product training. Use of Beta-quality product will ensure that sufficient testing has been completed to enable successful product demonstrations to be conducted.

Acceptance criteria for the Demonstration Version should include:

- Identification of the product build that will be used for the demonstration version
- Criteria for availability and distribution established
- Population of demonstration version with sample data and creation of demonstration path, if desired
- Completion of any necessary training and user documentation providing instructions on access and use of the demonstration version, including identification of any known issues that staff should avoid when demonstrating.
- Frequently Asked Questions (FAQ) document providing responses to questions likely to arise during demonstrations.
- Completion of peer review. The Iterative Document Review and Approval Process is recommended for the FAQ and all other user documentation. A Cross-Functional Peer Review Meeting is recommended to review the demonstration version prior to its release. This may be done in conjunction with the training.

Dependencies for the Demonstration Version include:

- Ready for Beta Testing

Recommended Uses for SourceForge:

Use the **SourceForge Document Manager** to store any documentation on use of the Demonstration Version and the demonstration path, if desired.

Use the **SourceForge Task Manager** to manage the schedule for all Product Release Preparation phase deliverables.

Use the **SourceForge File Release System** to release the Demonstration Version and other product builds.

Use **SourceForge News** to announce the availabilty of the Demonstration Version.

4.3. Marketing and Channel Plan

Suggested Owner: Software Product Management

The primary goal of the Marketing and Channel Plan is to document all marketing-related activities that will be conducted in support of any product release intended for distribution and sales outside the organization. If your product is intended for internal use only, the Marketing and Channel Plan may be reduced or eliminated entirely. If adoption of the product by an internal audience is expected, internal marketing and awareness building activities should be conducted. For products that are not expected to be used in their current form, but serve only as inputs into a larger development project, marketing activities may not be necessary at this point.

For products that will be offered for sale, the Marketing and Channel Plan should include activities associated with the adoption of the product by the sales and distribution channel, both internal and external to your organization, as well as general marketing activities such as press releases, media relations, website, print, and other advertising, and development of sales tools and marketing collateral. Strategies should be included for notifying and preparing reseller and distribution partners to sell and support the product, and key sales issues such as pricing and licensing models, upgrade and migration strategies, and identification of target customers or market segments should also be addressed. The activities defined in the Marketing and Channel Plan should remain consistent with the strategic goals of the product release as defined in the Business Requirements Document and Product Requirements Documents.

The Marketing and Channel Plan should include the following information:

- Positioning of the product within applicable market segments
- Strategy for press releases and other announcement-related activities
- Strategy for media relations, including analysts and applicable thought leaders
- Strategy for advertising via website, print media, television, radio, direct marketing, or other methods
- Strategy for informing channel and other business partners
- Strategy for developing all sales tools and marketing collateral to be available for the product release
- Strategy for developing product packaging, if applicable
- Sales strategy and identification of target customers or market segments
- Any new programs targeted at maximizing customer adoption of the product
- New product pricing or licensing models, if applicable

- Upgrade or migration strategy for existing customers with an earlier version of the product (this may be done as part of, or in conjunction with, the Service and Support Plan.)
- References to Training and release of the Demonstration Version (may be cross-references to other documents.)
- Information regarding any general corporate marketing initiatives that may impact the release, although they may not be directly related
- Strategy for conducting post-release customer satisfaction analysis
- Dates for all action items and deliverables

When the Marketing and Channel Plan is completed, the Task Manager should be used to manage all required activities and deliverables. You may choose to update your Final Project Plan and use the Task Manager project already in use, or you may create a new project or task folder to manage only marketing-related items.

Acceptance criteria for the Marketing and Channel Plan should include:

- Completion of peer review. The Iterative Document Review and Approval Process is recommended. Additional peer review criteria are included as an appendix to the document template.
- All marketing-related activities and deliverables incorporated into the Project Plan and managed using the Task Manager.

Dependencies for the Marketing and Channel Plan include:

- Product Plan
- Product Requirements Documents
- Product Launch Strategy

Recommended Uses for SourceForge:

Use the **SourceForge Document Manager** to store the Marketing and Channel Plan at all stages of its development (draft through final) and to manage the Iterative Document Review and Approval Process. See Figure 21 on page 276 for additional detail on this process.

Use the **SourceForge Task Manager** to manage any additional dates added to the Final Project Plan as a result of the Marketing and Channel Plan, or to manage a new project plan specific to marketing-related activities.

4.4. Service and Support Plan

Suggested Owner: Software Product Management

The goal of the Service and Support Plan is to define the both the strategy for supporting the product following its release, and the activities necessary to adequately prepare Technical Support and other staff for providing such support. The Service and Support Plan should therefore cover both pre- and post-release activities. Since product training is a key element of the Service and Support Plan, you may wish to begin this document in conjunction with the Product Training Plan completed during the Development phase, to ensure that training requirements for the support organizations are represented early in the planning stages.

The Service and Support Plan is also highly dependent upon other documents produced during the Release Preparation and earlier phases. Documents such as the Product Requirements Documents, Technical Design Documents, draft user documentation, Marketing and Channel Plan, and Product Launch Strategy should be reviewed by the support organizations in order to fully understand the scope of the product, any new or changed functionality, and the impact that activities planned by other functional groups within the organization may have when developing the strategy for supporting the product.

The Service and Support Plan should include the following information:

- Definition of any new services plans, including plans for Technical Support, Professional or Field Services, and any other organization providing product support.
- Identification of any third party hardware or software associated with the product.
- Strategy and budget for procurement of any additional hardware, software, or personnel required to support the product.
- Identification of training required to support the installation and use of the product.
- Identification of training required to support the installation and use of any third-party hardware or software associated with the product.
- Plans for reviewing the user documentation and other customer-facing technical materials prior to release.
- Strategy for updating a knowledge base or other similar support tools with new product information.
- Strategy for updating a support tracker or other customer support management tools if needed to support the new product.
- Strategy for involving support staff in pre-release testing activities.

- Strategy for identification of most likely initial customer concerns, and development of a FAQ document or similar predetermined responses.
- Upgrade or migration strategy for existing customers with an earlier version of the product. (This may be done as part of, or in conjunction with, the Marketing and Channel Plan.)
- Dates for all action items and deliverables.

When the Service and Support Plan has been completed, the Task Manager should be used to manage all required activities and deliverables. You can choose to update your Final Project Plan and use the Task Manager project already in use, or you may create a new project or task folder to manage only support-related items.

Acceptance criteria for the Service and Support Plan should include:

- Completion of peer review. The Iterative Document Review and Approval Process is recommended. Additional peer review criteria are included as an appendix to the document template.
- All service and support-related activities and deliverables incorporated into the Project Plan and managed using the Task Manager.

Dependencies for the Service and Support Plan include:

- Product Requirements Documents
- Technical Design Documents
- User Documentation
- Product Launch Strategy
- Marketing and Channel Plan

Recommended Uses for SourceForge:

Use the **SourceForge Document Manager** to store the Service and Support Plan at all stages of its development (draft through final) and to manage the Iterative Document Review and Approval Process. See Figure 21 on page 276 for additional detail on this process.

Use the **SourceForge Task Manager** to manage any additional dates added to the Final Project Plan as a result of the Service and Support Plan, or to manage a new project plan specific to support-related activities.

4.5. Product Training

Suggested Owner: Software Product Management

Providing comprehensive product training is one of the most critical elements in ensuring the success of your product release. During the Release Preparation phase, the training activities defined in the Product Training Plan are carried out, and post-training follow-up activities are conducted to ensure attendees' understanding of how to use, sell, and support the new product.

The Product Training described in this section refers to training conducted for an audience of participants internal to your organization. Attendees should include all staff who will be engaged in using, selling, or supporting the product, or conducting other product-related activities. In addition to establishing proficiency within your organization, another key success factor is the ability of attendees to then create and conduct training programs for customers and other audiences external to the organization, if applicable.

It is highly recommended that Product Training be conducted only after the release of a Demonstration Version of the product, in order to allow for live product demonstration and hands-on training exercises. Availability of a Demonstration Version also allows staff to continue to work with the product and increase their proficiency following the Product Training.

When completed, all attendees should complete a Training Evaluation Form to evaluate the success of the training. The Training Evaluation Forms should be posted to the Organizational Training project in SourceForge, or whatever location you have established in which to maintain training records.

Acceptance criteria for Product Training should include:

- Successful completion of all activities defined in the Product Training Plan.
- Successful completion of post-training survey, quiz or exam, or other feedback-gathering activities to ensure attendees' comprehension of the material presented, and their ability to work effectively with the product. Collecting this post-training feedback constitutes peer review for the training.
- Completion of Training Evaluation Forms by all attendees.

Dependencies for Product Training include:

- Product Training Plan
- Service and Support Plan
- Release of Demonstration Version

Recommended Uses for SourceForge:

Use the **SourceForge Document Manager** to store any training materials to be used for the Product Training, plus the results of training feedback for use in improving future materials.

Use the **SourceForge Task Manager** to manage the schedule for all Release Preparation phase deliverables.

Use the **SourceForge File Release System** to release product builds for use during Product Training.

4.6. Final System Testing

Suggested Owner: Software Testing

Upon completion of the Beta Testing period, the Software Testing organization conducts Final System Testing. Final System Testing is the last round of product testing conducted prior to release, and tests the correction of all product defects and the incorporation of any new code that may have been developed during the Beta Testing period. The test scripts executed during Alpha System Testing are re-executed against the updated product code, and generally must run completely, without halt or error, in order to consider each test passed. In addition, no high priority bugs may be outstanding, as per the Product Defect Prioritization Policy, in order to successfully complete Final System Testing. Regression testing or stress testing may also be conducted during Final System Testing if indicated in the Software Test Plan.

As with Alpha System Testing and Beta Testing, the change control policy initiated in the Development phase remains in effect throughout Final System Testing; however, new features should not be introduced at this stage, and code changes should be strictly limited to correction of high-priority product defects. The SourceForge Tracker should continue to be used to record and communicate the status of all product defects, and Software Engineering should remain actively engaged in fixing bugs. If the exit criteria for previous testing periods have been met, the number of high priority bugs remaining at this stage of testing should be minimal, limited to those identified during Final System Testing.

The process of testing, identifying and correcting high priority bugs, and retesting continues until the exit criteria for Final System Testing are met. Upon successful completion of Final System Testing, the product code is frozen and no further changes are made. Any remaining outstanding issues will be documented in the Release Notes.

Acceptance criteria for Final System Testing should include:

- All functional, performance, integration, environmental, and other test scripts passed
- No outstanding high priority bugs, as defined in the Product Defect Prioritization Policy
- All test scripts run completely without error
- Performance testing results meet or exceed requirements
- Ready for final code freeze

Dependencies for Final System Testing include:

- Completion of Beta Testing

Software Quality Assurance Audits

Software Quality Assurance audits are recommended for a number of the components of Final System Testing. Audits of processes used during Final System Testing and activities conducted as a result of Final System Testing are also recommended.

Reference the Software Quality Assurance section of the Product Plan and the Software Quality Assurance audit checklists for each CDP phase for a complete list of all artifacts, processes, and activities to be audited.

Recommended Uses for SourceForge:

Use CVS as your Software Configuration Management (SCM) tool to manage all code developed during the Release Preparation and other phases, including product defects corrected during Final System Testing. Use the **SourceForge SCM interface** to browse the contents of your SCM repository.

Associate code commits with product requirements and bugs **Tracker** artifacts where applicable. Any updates to code associated with a Tracker artifact during the initial code creation stage will automatically be reflected in the history of the Tracker artifact, as will any updates to the artifact itself.

Use the **SourceForge Tracker** to manage all product defects identified during Final System Testing.

Use the **SourceForge Task Manager** to manage the schedule for all Release Preparation phase deliverables.

Use the **SourceForge File Release System** to release product builds to Software Testing or other audiences.

Use the **SourceForge Document Manager** to store the test scripts to be executed during Final System Testing, and the results of Final System Testing.

4.7. Final Code Freeze

Suggested Owner: Software Engineering

After Final System Testing is successfully completed, all code is considered fully complete and no further modifications of any nature are made. Once this milestone is reached, tag the code as per your SCM policy and prepare final product builds for release to their intended audience.

Although complete, final product builds should not be released using the File Release System until the Release Readiness Checkpoint is successfully passed, and final approval is obtained to proceed with the product release.

Acceptance criteria for Final Code Freeze should include:

- Completion of Final System Testing (no outstanding high priority bugs as per the Product Defect Prioritization Policy)
- Ready for product release

Dependencies for Final Code Freeze include:

- Completion of Final System Testing

Software Quality Assurance Audits

Software Quality Assurance audits are recommended for a number of artifacts after being placed under the Change Control Policy. Audits of change control processes and activities conducted after formal change requests are also recommended.

Reference the Software Quality Assurance section of the Product Plan and the Software Quality Assurance audit checklists for each CDP phase for a complete list of all artifacts, processes, and activities to be audited.

Recommended Uses for SourceForge:

Use CVS as your Software Configuration Management (SCM) tool to manage all code developed during the Release Preparation and other phases. Use the **SourceForge SCM interface** to browse the contents of your SCM repository. Once the code base and user interface are frozen, the code should be tagged as per your SCM Policy.

Use the **SourceForge File Release System** to release final product builds to their intended audiences, following the Release Readiness Checkpoint

Use the **SourceForge Task Manager** to manage the schedule for all Release Preparation phase deliverables.

Use the **SourceForge Tracker** to manage any product defects or other artifacts outstanding following final code freeze. Outstanding artifacts should be transferred to another Tracker or otherwise noted for inclusion in a future release, and for inclusion in the product Release Notes.

4.8. Final User Documentation

Suggested Owner: Software Engineering, Software Quality Assurance, or Software Product Management

During the Release Preparation phase, the draft user documentation completed during the Development phase is revised and completed. Input gathered during the Iterative Document Review and Approval process should be reviewed and processed (either included in the final documentation, deferred for inclusion in a later release, or rejected for inclusion.) Additional detail may be added as the product continues to reach higher levels of stability, enabling the technical writer to more accurately describe the functionality of the product.

If the user documentation will include product screen shots, it is recommended that they be taken during the Beta Testing period, when the product has reached a high level of stability and the user interface is largely completed and changes are strictly limited. If possible, a frozen user interface milestone is highly desirable, as it provides the technical writer with a fixed point after which all product screen shots may be considered final. This is often not possible, however, as high-priority product defects, including those related to the user interface, may continue to be corrected throughout Final System Testing. If this is the case, the user documentation may not be considered fully completed until following final code freeze, when the final functionality and user interface of the product can be confirmed.

Following final code freeze and the completion of the final user documentation Iterative Document Review and Approval Process, any last-minute changes to the product, any outstanding known issues, and any other differences between the final user documentation and the final product will be detailed in the Release Notes.

Acceptance criteria for the Final User Documentation should include:

- All chapters and other documentation completed according to the Documentation Plan.
- All feedback received during the draft user documentation Iterative Document Review and Approval Process reviewed and processed.
- Completion of peer review. The Iterative Document Review and Approval Process is recommended. Additional peer review criteria are included as an appendix to the document template.

Software Quality Assurance Audits

Software Quality Assurance audits are recommended for the User Documentation. Audits of processes used to produce the User Documentation are also recommended.

Reference the Software Quality Assurance section of the Product Plan and the Software Quality Assurance audit checklists for each CDP phase for a complete list of all artifacts, processes, and activities to be audited.

Dependencies for Final User Documentation include:

- Draft User Documentation
- Beta Testing
- Final Code Freeze

Recommended Uses for SourceForge:

Use the **SourceForge Document Manager** to store the Service and Support Plan at all stages of its development (draft through final) and to manage the Iterative Document Review and Approval Process. See Figure 21 on page 276 for additional detail on this process.

Use the **SourceForge Task Manager** should be used to manage the schedule for all Release Preparation phase deliverables.

Use the **SourceForge Tracker** to manage requests for additional or modified text or other documentation-related materials.

4.9. Product Packaging and Collateral Materials

Suggested Owner: Software Product Management

In many cases, your product may be intended strictly for use within your organization, and as such, will not require formal packaging for distribution. The final product builds may be posted to the SourceForge File Release System for access by their intended audience, or distributed on CD or other portable method in an informal manner.

If the product will be offered for sale, however, or a more formal method of distribution is desired, product packaging and collateral materials must be prepared prior to the release of the product. Product packaging may include physical packaging such as boxes, CD cases, shrinkwrap, and other package components, plus collateral materials such as product information, package inserts, and other marketing-related items. It may also include electronic packaging elements such as wrappers containing click-through license agreements, software license management mechanisms, registration and other product information. Specific product packaging and collateral materials required should be described in the Marketing and Channel Plan.

Product Packaging and Collateral Materials can include:

- Bill of Materials (BOM) listing all components to be included in the product packaging and their part numbers
- Physical package components such as boxes, CDs, CD cases, box stiffeners, shrinkwrap
- Collateral materials such as manuals, registration cards, Release Notes, marketing offers, package stickers, or other materials
- Artwork for boxes, CDs, manual covers, stickers, and other components
- Text such as product information, system requirements, license information, and copyright information
- Artwork and text for electronic packaging elements such as wrappers

Due to sometimes lengthy manufacturing times, and the requirement to reach Final Code Freeze before manufacturing may be completed, the Product Packaging and Collateral Materials may not be fully completed until the Release phase. However, all materials should be prepared and approved prior to the Release Readiness Checkpoint.

Acceptance criteria for Product Packaging include:

- Completion of peer review. The Iterative Document Review and Approval Process is recommended for all packaging and collateral artwork and text. A Cross-Functional Peer Review Meeting is recommended for all other materials.
- Readiness to submit for manufacturing upon completion of Final Code Freeze and the Release Readiness Checkpoint.

Dependencies for Product Packaging include:

- Marketing and Channel Plan

Recommended Uses for SourceForge:

Use the **SourceForge Document Manager** to store the BOM, packaging collateral, artwork, and any other documents created to support the Product Packaging, and to manage the Iterative Document Review and Approval Process. See Figure 21 on page 276 for additional detail on this process.

Use the **SourceForge Task Manager** to manage the schedule for all Release Preparation phase deliverables.

4.10. Release Readiness Checkpoint

Suggested Owner: Software Product Management

Since its inception in the Design phase, the Launch Team will have met regularly to monitor the ongoing progress and status of the product and its associated development project. If the project has been well managed throughout the Release Preparation and earlier phases, the factors listed below will have been monitored and addressed and should not present any significant concerns by the time the Release Readiness Checkpoint is reached. The Release Readiness Checkpoint should serve as a final pre-release meeting of the Launch Team, plus any additional stakeholders such as the Management Approval Team to formally record approval of the product for release.

Factors to consider in the Release Readiness Checkpoint can include:

- Does the product include all features and functionality described in the Product Requirements Documents?
- Have the features been implemented according to the Technical Design Documents?
- Are there any outstanding high-priority bugs, as defined in the Product Defect Prioritization Policy?
- Have all deviations from the Product Requirements Documents, Technical Design Documents, or Product Defect Prioritization Policy been documented and approved as per the Change Control Policy?
- Has Final System Testing been completed according to established exit criteria?
- Have all required actions and deliverables been completed in the Release Preparation and earlier CDP phases?
- Are all functional units in the organization fully prepared for the product release?

After the Release Readiness Checkpoint is successfully completed, the final product builds can be released to their intended audience using the SourceForge File Release System.

Acceptance criteria for Release Readiness Checkpoint should include:

- Agreement by all Launch Team members that product is ready for release. The cross-functional Launch Team meeting serves as peer review for this activity.

Dependencies for Release Readiness Checkpoint include:

- Final Code Freeze
- Final User Documentation
- Product Training
- Service and Support Plan
- Marketing and Channel Plan
- Product Launch Strategy
- Product Packaging

Recommended Uses for SourceForge:

Use the **SourceForge Document Manager** to store all relevant documentation for review by stakeholders in preparation for the Release Readiness Checkpoint.

Use the **SourceForge Task Manager** to manage the schedule for all Release Preparation phase deliverables, and to review the conditions associated with any deviations from the Final Project Plan that may impact the Release Readiness Checkpoint.

Use the **SourceForge File Release System** to release final product builds to their intended audiences, following the Release Readiness Checkpoint.

Use **SourceForge News** to announce the completion of the Release Readiness Checkpoint.

Release Preparation Phase Exit Criteria

During the Release Preparation phase, members of all stakeholder groups will have worked closely to ensure that the product has been completed, fully tested, and is ready for release. All preparatory supporting activities will also have been completed.

A Release Preparation Phase Exit Review meeting should be held with key stakeholders from all impacted organizations. The following checklist should be used to ensure all exit criteria are met.

✓	Release Preparation Phase Exit Checklist
❑	SourceForge project now containing all elements specified on page 166.
❑	Beta Testing completed with no high priority bugs outstanding.
❑	Demonstration Version released.
❑	Service and Support Plan completed, peer-reviewed, and approved by required approvers. Final version stored in the Document Manager and locked.
❑	Actions and deliverables from the Service and Support Plan input into the Task Manager.
❑	Marketing and Channel Plan completed, peer-reviewed, and approved by required approvers. Final version stored in the Document Manager and locked.
❑	Actions and deliverables from the Marketing and Channel Plan input into the Task Manager.
❑	Product Training completed.
❑	Software Quality Assurance software baseline audit completed prior to Final System Testing.
❑	Final System Testing completed with no high priority bugs outstanding.
❑	Code, including user interface, completed, frozen, and ready for release.
❑	Final User Documentation completed, peer-reviewed, and approved by required approvers. Final version stored in the Document Manager and locked..
❑	Product Packaging completed.
❑	All project training activities scheduled for completion in the Release Preparation phase completed.

✓	Release Preparation Phase Exit Checklist
❑	All Training Evaluation Forms completed and posted to the Organizational Training project on SourceForge.
❑	Release Preparation phase Software Quality Assurance audit checklist completed. Checklist posted to the Document Manager.
❑	Periodic engineering review meetings conducted.
❑	Regular Launch Team meetings conducted, including reviews of the Task Manager and the Project Logistics Tracker.
❑	Periodic reviews of subcontractor progress completed, if applicable.
❑	Release Readiness Checkpoint successfully passed. Checklist posted to the Document Manager.
❑	Release Preparation Phase Exit Review successfully completed using this checklist. Checklist posted to the Document Manager.

Using SourceForge to Support the Release Preparation Phase

SourceForge supports the goals of the Release Preparation phase by allowing you to:

- Use CVS as your Software Configuration Management (SCM) tool to manage all code developed during the Release Preparation and other phases. Use the **SourceForge SCM interface** to browse the contents of your SCM repository.
- Associate code commits with product requirements Tracker artifacts using **Tracker / Code Associations.**
- Release product builds to Software Testing and their final intended audiences using the **File Release System.**
- Manage product defects using the **Tracker.**
- Enable management visibility and control of work being done both internally and by subcontractors using the **Task Manager.**
- Review all documentation stored in the **Document Manager** in preparation for the Release Readiness Checkpoint.
- Review the conditions associated with any deviations from the Final Project Plan that may impact the Release Readiness Checkpoint, using the **Task Manager**.
- See *SourceForge and the Capability Maturity Model for Software, A Supplement to the SourceForge Collaborative Development Process* for additional detail on using SourceForge and the CDP to satisfy SW-CMM Level 2 and 3 requirements.

Release Preparation Phase Templates

The following table provides a quick reference to all templates provided for Release Preparation phase deliverables and activities.

All templates are provided on the CD included with this manual, and at the following URL: *https://www.vasoftware.com/cdp/*

Table 18. Release Preparation Phase Templates

Ref#	Template	CD File Location
4.3.1	Marketing and Channel Plan	\relprep\[4.3.1]marketing.dot \relprep\[4.3.1]marketing.pdf
4.4.1	Service and Support Plan	\relprep\[4.4.1]service.dot \relprep\[4.4.1]service.pdf
4.10.1	Release Readiness Checkpoint	\relprep\[4.10.1]readiness.dot \relprep\[4.10.1]readiness.pdf
4.11.1	SQA Checklist Release Preparation Phase	\relprep\[4.11.1]sqarelprep.dot \relprep\[4.11.1]sqarelprep.pdf
4.12.1	Release Preparation Phase Exit Checklist	\relprep\[4.12.1]releaseexit.dot \relprep\[4.12.1]releaseexit.pdf

Please reference “Numbering” on page 10 for additional information on document numbering conventions.

CHAPTER 7

Release Phase

This chapter includes the following information:

- Release Phase Overview
- Release Phase Workflow Process Diagram
- Release Phase Inputs
- Release Phase Activities and Deliverables
- Release Phase Exit Criteria
- Using SourceForge to Support the Release Phase
- Release Phase Templates

Release Phase Overview

The primary goal of the Release phase is to distribute the product to its intended audience. The activities conducted during this phase should ensure that the product meets both internal and external customer expectations and includes all training materials, user documentation, collateral materials, and an established support infrastructure. In cases where the product is not intended for use on its own, but will serve as an input into a larger development project, sufficient technical information must also be provided to the downstream consumers of the product to enable them to successfully integrate it as intended.

To enter the Release phase, the Release Readiness Checkpoint must be successfully completed, with formally recorded approval by key stakeholders that the product meets all established functionality and quality standards, and that all groups within the organization are prepared to support the release.

Activities conducted in the Release phase include:

- Completing Product Release Notes
- Publishing Quality Statement
- Issuing the Release Announcement
- Releasing the product to its intended audience
- Confirming product acceptance

At the conclusion of this phase, the product will have been formally released to its intended audience, and release analysis activities will have confirmed initial acceptance of the product by customers, internal sales staff, resellers and other business partners. Any significant issues with the product release will have been addressed before proceeding to the Sustaining Engineering phase of the CDP.

The following diagram indicates the activities and deliverables necessary to exit the Release phase. The suggested contents of each activity and deliverable and the recommended process for producing them are defined in the following sections.

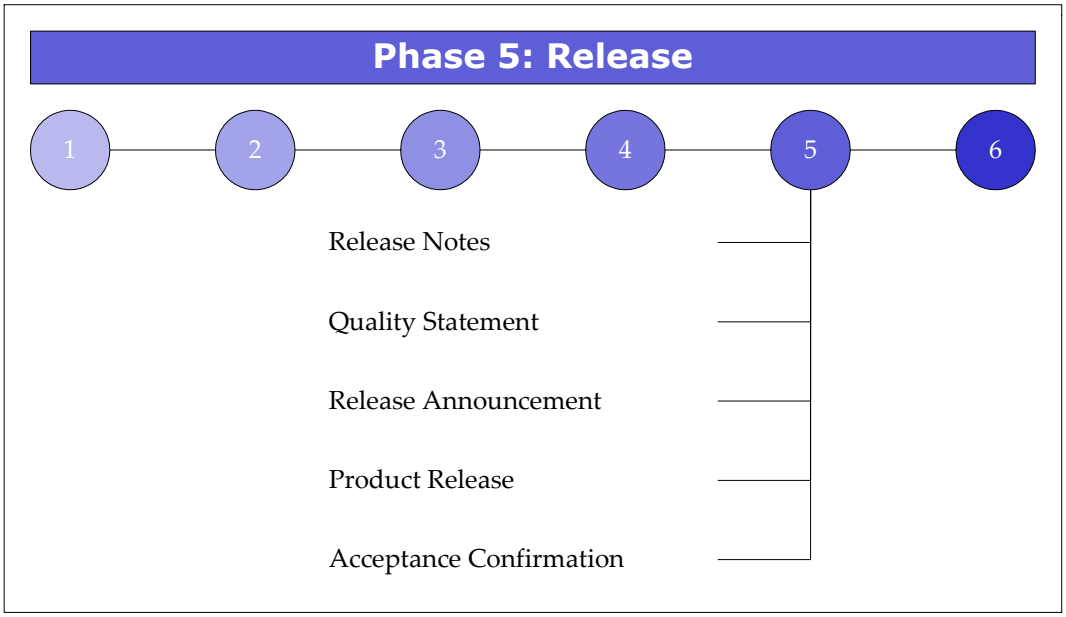


Figure 14. Release Phase Deliverables

The Release Phase Workflow Process

The following diagram illustrates the recommended best practice workflow process for the Release phase.

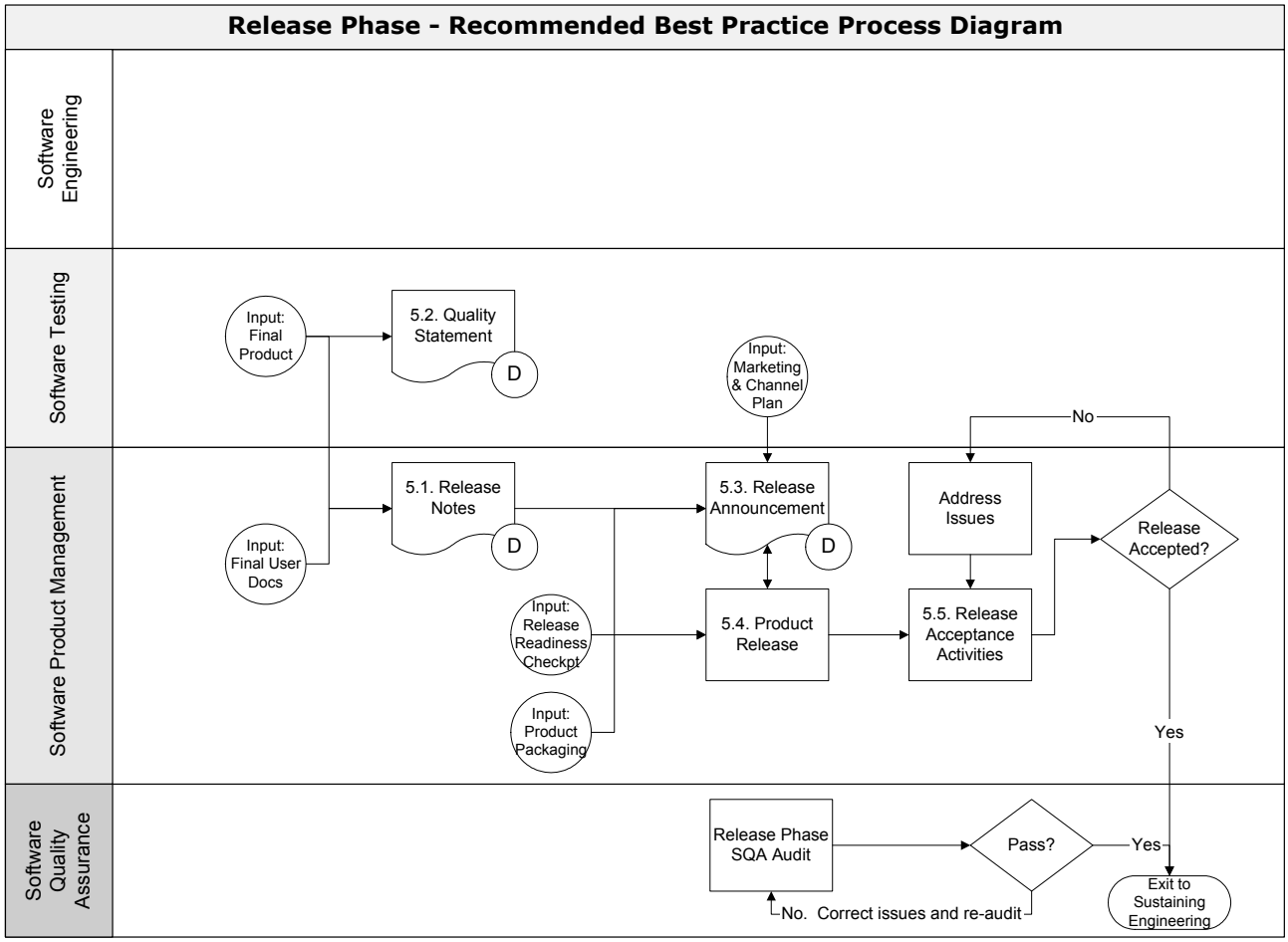


Figure 15. Release Phase Workflow Process

Please see Please see "Workflow Processes" on page 11 for a full description of all flowchart notation.

Release Phase Inputs

The following inputs are required to enter the Release phase:

Release Readiness Checkpoint

The Release Readiness Checkpoint is the key pre-requisite for entering the Release phase. The successful completion of the Release Readiness Checkpoint serves as the formal approval by key stakeholders that the product meets all established functionality and quality standards, and that all functional groups within the organization are prepared to support the release. The Release Readiness Checkpoint confirms that all product training, user documentation, collateral materials, and support infrastructure are in place to support the initial acceptance of the release, which will be measured during the Release phase.

Final Product

The final product is released to its intended audience in the Release phase, and also forms the basis for release documentation such as the product Release Notes and Quality Statement. "Final Product" refers to the final product builds prepared following Final Code Freeze and the Release Readiness Checkpoint.

Final User Documentation

The Final User Documentation is referenced, in additional to the final product, when creating the product Release Notes.

Marketing and Channel Plan

The Marketing and Channel Plan is referenced to determine the content and strategy for the Release Announcement.

SourceForge Project

The SourceForge project created during the Product Definition phase should now be populated with the materials completed in the Product Definition, Design, and Development, and Release Preparation phases.

SourceForge Component	Contents
At the beginning of the phase	
Project Information	Project-level information such as project description, members, and roles.
Document Manager	A Document Manager folder hierarchy containing final versions of all documents created in the Product Definition, Design, Development, and Release Preparation phases, including automatically stored revision and review histories. The Launch Team folder now containing all meeting minutes, notes, and other documentation generated to date. Any formal change requests submitted following Code Freeze. Supplementary documentation, such as research or other reference material. The appropriate CDP process templates retrieved from your Software Process Database and stored in your project's Document Manager.
Software Process Database	A Software Process Database used for managing all standard procedures, process templates, checklists, and other materials.
Trackers	Trackers containing all feature requests, bugs, and other artifacts that have been addressed in the product. A Project Logistics Tracker used to track critical estimates, resources, and risks. Trackers containing all feature requests, bugs, and other artifacts that have been deferred to a later product release. Closure of all product requirements Tracker artifacts (with the exception of any items deferred per formal change request.)
Discussion Forums and Mailing Lists	One or more Discussion Forums and Mailing Lists for discussion of project-related topics, including collaborative decision making on design challenges.
SCM Repository	SCM repository containing completed, frozen, and fully-tested code.
File Release System	One or more completed product build/s released to their final intended audience in the File Release System.

SourceForge Component	Contents
Tracker / Code Associations	Code commits associated with Tracker artifacts for traceability of all relevant product requirements.
Task Manager	One or more projects in the Task Manager to manage various pieces of your Final Project Plan, including any activities and deliverables added upon completion of the Marketing and Channel Plan and Service and Support Plan. All Task Manager tasks scheduled for completion prior to product release now marked as completed.

SourceForge Component	Contents
By the end of the phase	
Document Manager	Final, approved versions of all required Release phase documents stored in the Document Manager All applicable procedures, process templates, and other materials retrieved from your Software Process Database and stored in the project's Document Manager.

Release Phase Activities and Deliverables

All activities and deliverables are identified by number in the Release Phase Workflow Process Diagram.

Descriptions of each activity or deliverable, suggested contents and acceptance criteria, and dependencies on other activities or deliverables are provided with each item. Activities or deliverables for which peer review or a Software Quality Assurance audit is recommended are identified. Considerations for tailoring deliverables to meet specific organizational needs are indicated where appropriate, and recommended uses for SourceForge are included at the end of each section.

Icons

Activities and deliverables for which templates are provided are identified by the symbol shown here. Please see “Activities and Deliverables” on page 13 for a complete list of all symbols used to identify recommended activities and uses for SourceForge.

5.1. Release Notes

Suggested owner: Software Product Management

When the product is released, it is often desirable to prepare a short document providing critical information to users in an easily accessible location, such as on the product CD or referenced in the installer or product start-up. Information such as key product features, any key enhancements since a previous release, and critical information such as installation instructions or references to more comprehensive user documentation can be included, as well as any variations between the final user documentation and the final product that can exist as a result of last-minute corrections to the product code made during Final System Testing. The Release Notes should serve as a supplement to, not an alternative to, comprehensive user documentation.

The Release Notes can include:

- Key product features
- Key product enhancements
- Defects corrected since a previous release
- Known issues and workarounds
- Installation instructions
- Registration information
- References to comprehensive user documentation
- Additions or changes to the user documentation based on late changes to the code
- Final definition of Reference Platforms on which the product has been validated for installation and execution
- Any additional information you wish to display prominently to users

Acceptance criteria for the Release Notes should include:

- Completion of peer review. The Iterative Document Review and Approval Process is recommended. Additional peer review criteria are included as an appendix to the document template.
- Release Notes provided for inclusion on the product CD, website, or other means of distribution.

Dependencies for the Release Notes include:

- Final Code Freeze
- Final User Documentation

Recommended Uses for SourceForge

Use the **SourceForge Document Manager** to store the Release Notes at all stages of development (draft through final) and to manage the Iterative Document Review and Approval Process. See Figure 21 on page 276 for additional detail on this process.

Use the **SourceForge Task Manager** to manage the schedule for all Release phase deliverables.

5.2. Quality Statement

Suggested Owner: Software Testing

The goal of the Quality Statement is to summarize the outcome of the testing activities conducted during the Development and Release Preparation phases. The Quality Statement is generally an internal document used primarily for assessing the success of completed quality assurance activities, and for identifying potential areas for future process improvement. For products intended for use as components in larger products, the Quality Statement provides critical quality information to the downstream consumers of the product. This quality information can be used when planning integration and further testing activities.

The contents of the Quality Statement can vary based on the scope of the project and other considerations as defined in the Software Test Plan. The Quality Statement should serve as a follow-up to the Software Test Plan, and report the actual versus planned results of each section.

The Quality Statement can include:

Release Scope

A reiteration of the release scope defined in the Software Test Plan (summary of new features, enhanced functionality, pre-existing product defects), including any variations from the originally planned release scope, and justification for each variation.

Test Processes and Test Environment

Reiterate the test processes (manual, automated, regression, automated), and test environment (hardware and software configurations) defined in the Software Test Plan, including any variations from the originally planned test processes and test environment. Provide justification for any variations.

Tests

List all test scripts completed and executed, including justification for any variations from those identified in the Software Test Plan.

Project Goals

List the summary goal for the project, e.g. 'Complete testing according to defined acceptance criteria by x date.'

Acceptance Criteria

Define the acceptance criteria used to measure the success of each software testing activity.

Risks

List the risks and constraints applicable to the testing activities, including any that were identified following completion of the Software Test Plan. For all potential risks identified in the Software Test Plan, indicate whether or not they affected the testing activities as anticipated, and to what degree. Include the steps taken to mitigate any issues that resulted, and their consequences.

Test Cycle Milestones

Provide a milestone-level schedule of all significant testing activities, highlighting any significant deviations from the original testing project plan, and justification for each deviation. Task Manager reporting can be useful for this section.

Reporting

The Software Test Plan included a list of all reports that will be produced both during and at the conclusion of the test cycle. Copies of all final reports should be included, or referenced by SourceForge URL, in the Quality Statement.

Acceptance criteria for the Quality Statement should include:

- Completion of peer review. The Iterative Document Review and Approval Process is recommended. Additional peer review criteria are included as an appendix to the document template.

Dependencies for the Quality Statement include:

- Final Code Freeze

Recommended Uses for SourceForge

Use the **SourceForge Document Manager** to store the Quality Statement during all stages of its development (draft through final) and to manage the Iterative Document Review and Approval Process. See Figure 21 on page 276 for additional detail on this process.

Use the **SourceForge Task Manager** to manage the schedule for all Release phase deliverables. Generate Task Manager reports to justify any deviations from the original testing project plan provided in the Software Test Plan.

Use **SourceForge Reporting** to generate all reports specified in the Software Test Plan for inclusion in the Quality Statement.

5.3. Release Announcement

Suggested Owner: Software Product Management

When the Release Readiness Checkpoint, the Release Notes, and all other preparatory activities have been completed, the product is ready to be announced to its intended audiences. The contents of the Release Announcement can vary significantly depending on the intended audiences (internal distribution only, limited external distribution, available to the general public,) the scope of the release, and other considerations. The contents of the Release Announcement and the methods for its distribution should be included in the Marketing and Channel Plan and reflect the strategic goals defined in the Business Requirements Document .

In general, the primary goal of the Release Announcement is to provide all necessary information to announce the availability of the product and enable the intended audiences to obtain and begin using, selling, or supporting it. You may wish to prepare separate Release Announcements if you have different audiences to whom you wish to communicate significantly different information.

In some cases, you may wish to pre-announce the product to certain audiences to allow them sufficient time to prepare for the release, to begin generating interest in the product, or to conduct other pre-release activities. In such cases, it is important to follow up any pre-announcements with formal Release Announcements upon availability of the product.

The types of Release Announcements can include:

- Informal announcement to a limited audience
- Internal organization-wide announcement
- Notification to the reseller and distribution channel, or other business partners
- Formal press release
- Announcement on corporate Website
- Other marketing-related announcements such as print media, television, radio, Web, or other methods

The Release Announcement can include:

- Date of availability
- Summary of key product features
- Goals and intended audience for the product
- Performance, scalability, and other operational data
- Relationship to any previous versions of the product, e.g. free upgrade, patch update, etc.
- Methods for obtaining the product
- Methods for obtaining supplementary information such as user documentation, Technical Support, or other information
- Pricing, promotions, or other sales-related information

Acceptance criteria for the Release Announement should include:

- Completion of peer review. The Iterative Document Review and Approval Process is recommended. Additional peer review criteria are included as an appendix to the document template.
- Distribution of the Release Announcement to its intended audiences

Dependencies for the Release Announcement include:

- Release Readiness Checkpoint
- Release Notes
- Marketing and Channel Plan

Recommended Uses for SourceForge

Use the **SourceForge Document Manager** to store the Release Announcement during all stages of its development (draft through final) and to manage the Iterative Document Review and Approval Process. See Figure 21 on page 276 for additional detail on this process.

Use the **SourceForge Task Manager** to manage the schedule for all Release phase deliverables.

Use **SourceForge News** to announce the completion and availability of the product.

5.4. Product Release

Suggested Owner: Software Product Management

Once the Release Announcement is completed, the product should be simultaneously announced and released to its intended audiences. Depending on the scope of the release, its intended audience, and other considerations, the methods for releasing the product can vary significantly.

Once Final System Testing and the Release Readiness Checkpoint are successfully completed, Software Engineering or Software Testing will have released final product builds to the SourceForge File Release System. The Maturity Level feature of the File Release System should be used to help ensure that final versions are correctly identified. If the intended audience for the product is internal to the organization, releasing the final product builds to the File Release System may be the only necessary step. The Release Announcement should simply include a reference to the location of the files (plus any other supplementary information). For users monitoring the File Release package containing the final product builds, SourceForge will also provide an automatic notification when the final product builds are posted.

If the audience for the product includes parties external to the organization, or a more formal internal distribution process is desired, the final product builds should be obtained from the SourceForge File Release System, then distributed to the parties responsible for manufacturing CDs, uploading to a website for download, or enabling other internal or external distribution mechanisms.

In addition to your regular offsite backup cycle, it is highly recommended that following the product release, you archive the final product builds, the contents of your SourceForge project, and the contents of your SCM repository for disaster recovery purposes.

Acceptance criteria for the Product Release should include:

- Successful release of the product to its intended audience.

Dependencies for the Product Release include:

- Release Announcement
- Final Product
- Release Notes

Recommended Uses for SourceForge

Use the **SourceForge Document Manager** to store the Release Announcement, Release Notes, and other supplementary documentation applicable to the Product Release.

Use the **SourceForge Task Manager** to manage the schedule for all Release phase deliverables.

Use the **SourceForge File Release System** to release product builds to their intended internal or external audiences.

5.5. Release Acceptance Activities

Suggested Owner: Software Product Management

Following the product release, it is critical to ensure its initial acceptance by recipients. Initial acceptance refers to the confirmation by a sampling of the product's intended audience that they have received sufficient information and material to use, sell, or support the product as intended. Release acceptance can be confirmed by a subset of the product's early recipients; it does not imply that all ultimate users of the product must confirm acceptability before this requirement can be satisfied and the Release phase of the CDP concluded.

The goal of Release Acceptance activities is simply to ensure that the product and supporting materials are being used successfully by the product's early recipients, and to correct any issues that can arise immediately following the product release. If the product has multiple audiences with significantly different use cases or requirements, you may wish to conduct separate Release Acceptance Activities for each audience.

Release Acceptance Activities can include:

- Informal conversation with recipients.
- Providing a structured mechanism for recipients to provide general feedback. Some examples can include website message boards, email (e.g. cdpfeedback@vasoftware.com), or comment cards.
- Conducting surveys or other formal feedback programs.
- Post-sales follow-up with recipients by Sales representatives.
- Reviewing common issues encountered by Sales representatives.
- Reviewing common issues reported to Technical Support.
- Reviewing expected versus actual product sales.
- Reviewing expected versus actual progress toward strategic goals identified in the Business Requirements Document and Marketing and Channel Plan

Based on the feedback received from participants, corrective actions may need to be taken to improve the quality and usability of the product, supporting materials, or support infrastructure. Corrective actions can range from documentation updates or the publication of additional FAQs, to the requirement in more serious cases to issue a patch release to correct user-identified product defects. Many of the Release Acceptance Activities will be continued throughout the Sustaining Engineering phase, and are described in more detail in Chapter 8, Sustaining Engineering Phase.

Acceptance criteria for Release Acceptance Activities should include:

- Completion of peer review. The Iterative Document Review and Approval Process is recommended.
- Confirmation that early recipients have received sufficient information and material to use, sell, or support the product.
- Readiness to enter the Sustaining Engineering phase.

Dependencies for Release Acceptance Activities include:

- Release Announcement
- Product Release
- Release Notes
- Marketing and Channel Plan

Recommended Uses for SourceForge

Use the **SourceForge Document Manager** to store any documentation prepared to support Release Acceptance Activities.

Use the **SourceForge Task Manager** to manage the schedule for all Release phase deliverables.

Release Phase Exit Criteria

During the Release phase, members of all stakeholder groups will have worked closely to ensure that the product is released and deemed acceptable by its intended audiences. Any significant issues will have been corrected prior to proceeding to the Sustaining Engineering phase of the CDP.

A Release Phase Exit Review meeting should be held with key stakeholders from all impacted organizations. The following checklist should be used to ensure all exit criteria are met.

✓	Release Phase Exit Checklist
❑	SourceForge project now containing all elements specified on page 200.
❑	Release Notes completed, peer-reviewed, approved by required approvers, and submitted for inclusion on the product CD or other distribution method. Final version stored in the Document Manager.
❑	Quality Statement completed, peer-reviewed, and approved by required approvers. Final version stored in the Document Manager.
❑	Release Announcement completed, peer-reviewed, and approved by required approver, and distributed to its intended audience. Final version stored in the Document Manager.
❑	Product released to its intended audience.
❑	All project training activities scheduled for completion in the Release phase completed.
❑	All Training Evaluation Forms completed and posted to the Organizational Training project on SourceForge.
❑	Release phase Software Quality Assurance audit checklist completed. Checklist posted to the Document Manager.
❑	Periodic engineering review meetings conducted.
❑	Regular Launch Team meetings conducted, including reviews of the Task Manager and the Project Logistics Tracker.
❑	Periodic reviews of subcontractor progress completed, if applicable.
❑	Release Acceptance activities conducted confirming initial acceptance of the product by its intended audience.
❑	Release Phase Exit Review successfully completed using this checklist. Checklist posted to the Document Manager.

Using SourceForge to Support the Release Phase

SourceForge supports the goals of the Release phase by allowing you to:

- Release product builds to their final intended audiences using the **File Release System.**
- Manage post-release product defects using the **Tracker.**
- Enable management visibility and control of work being done both internally and by subcontractors using the **Task Manager.**
- Manage the document creation and approval process using the **Document Manager**.
- See *SourceForge and the Capability Maturity Model for Software, A Supplement to the SourceForge Collaborative Development Process* for additional detail on using SourceForge and the CDP to satisfy SW-CMM Level 2 and 3 requirements.

Release Phase Templates

The following table provides a quick reference to all templates used for the Release phase deliverables and activities.

All templates are provided on the CD included with this manual, and at the following URL: *https://www.vasoftware.com/cdp/*

Table 19. Release Phase Templates

Ref#	Template	File Location
5.1.1	Release Notes	\release\[5.1.1]releasenotes.dot \release\[5.1.1]releasenotes.pdf
5.2.1	Quality Statement	\release\[5.2.1]qualitystatement.dot \release\[5.2.1]qualitystatement.pdf
5.6.1	SQA Checklist Release Phase	\release\[5.6.1]sqarelease.dot \release\[5.6.1]sqarelease.pdf
5.7.1	Release Phase Exit Checklist	\release\[5.7.1]releaseexit.dot \release\[5.7.1]releaseexit.pdf

Please reference “Numbering” on page 10 for additional information on document numbering conventions.

CHAPTER 8

Sustaining Engineering Phase

This chapter includes the following information:

- Sustaining Engineering Phase Overview
- Sustaining Engineering Phase Workflow Process Diagram
- Sustaining Engineering Phase Inputs
- Sustaining Engineering Phase Activities and Deliverables
- Sustaining Engineering Phase Exit Criteria
- Using SourceForge to Support the Sustaining Engineering Phase
- Sustaining Engineering Phase Templates

Sustaining Engineering Phase Overview

The primary goal of the Sustaining Engineering phase is to support and maintain the product throughout the remainder of its lifecycle. The post-release analysis activities begun in the Release phase are continued, adoption by customers and other users is monitored closely, and bug fixes, patches, or other supporting materials are issued where needed. Formal analysis of the development process is also conducted to improve internal processes and procedures for future projects.

To enter the Sustaining Engineering phase, the initial Release Acceptance activities must have been successfully concluded in the Release phase. Many of them will be continued throughout the Sustaining Engineering phase.

Activities conducted in the Sustaining Engineering phase include:

- Monitoring customer adoption of the product
- Monitoring reported issues and feature requests
- Issuing bug fixes or patches as needed
- Conducting formal post-release analysis of the development project
- Modifying and improving internal CDP processes

The Sustaining Engineering phase is not formally concluded until the product reaches its end-of-life, which may be months or years from the date the product is released. The point at which you should formally quantify your release results and modify your CDP processes will be based largely on the timeline for beginning the next project. It is very common to enter the Product Definition phase of a new project while still in the Sustaining Engineering phase of one or more previous projects.

The following diagram indicates the activities and deliverables conducted during the Sustaining Engineering phase. The suggested contents of each activity and deliverable and the recommended process for producing them are defined in the following sections.

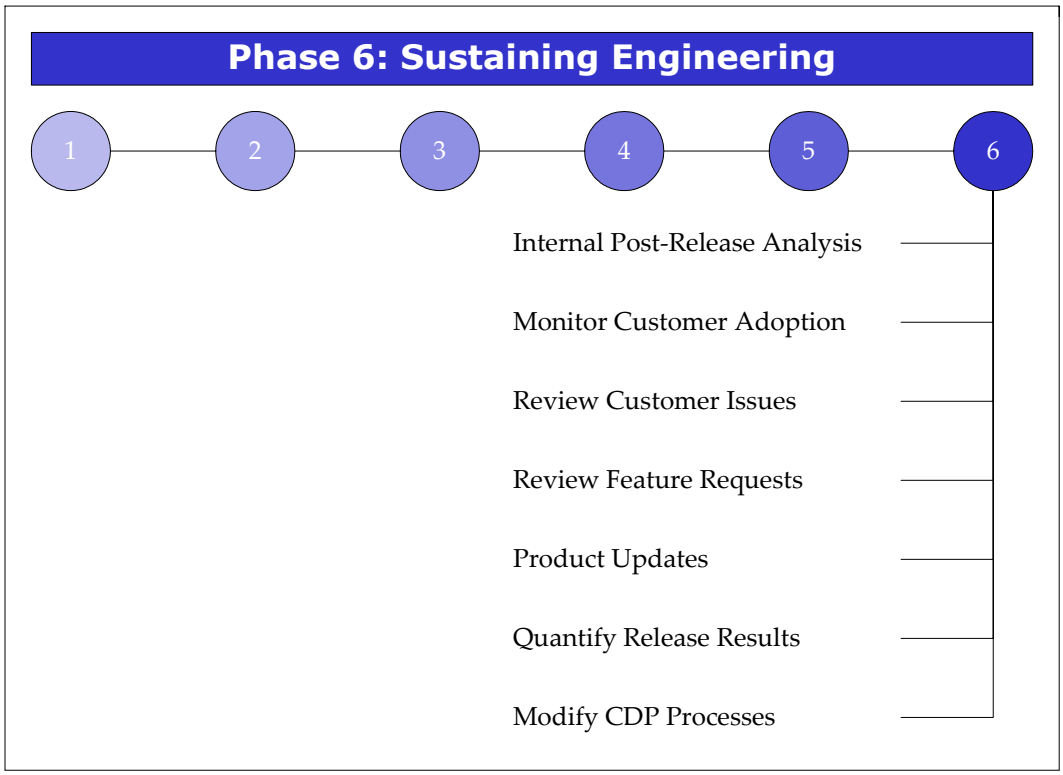


Figure 16. Sustaining Engineering Phase Deliverables

The Sustaining Engineering Phase Workflow Process

The following diagram illustrates the recommended best practice workflow process for the Sustaining Engineering phase.

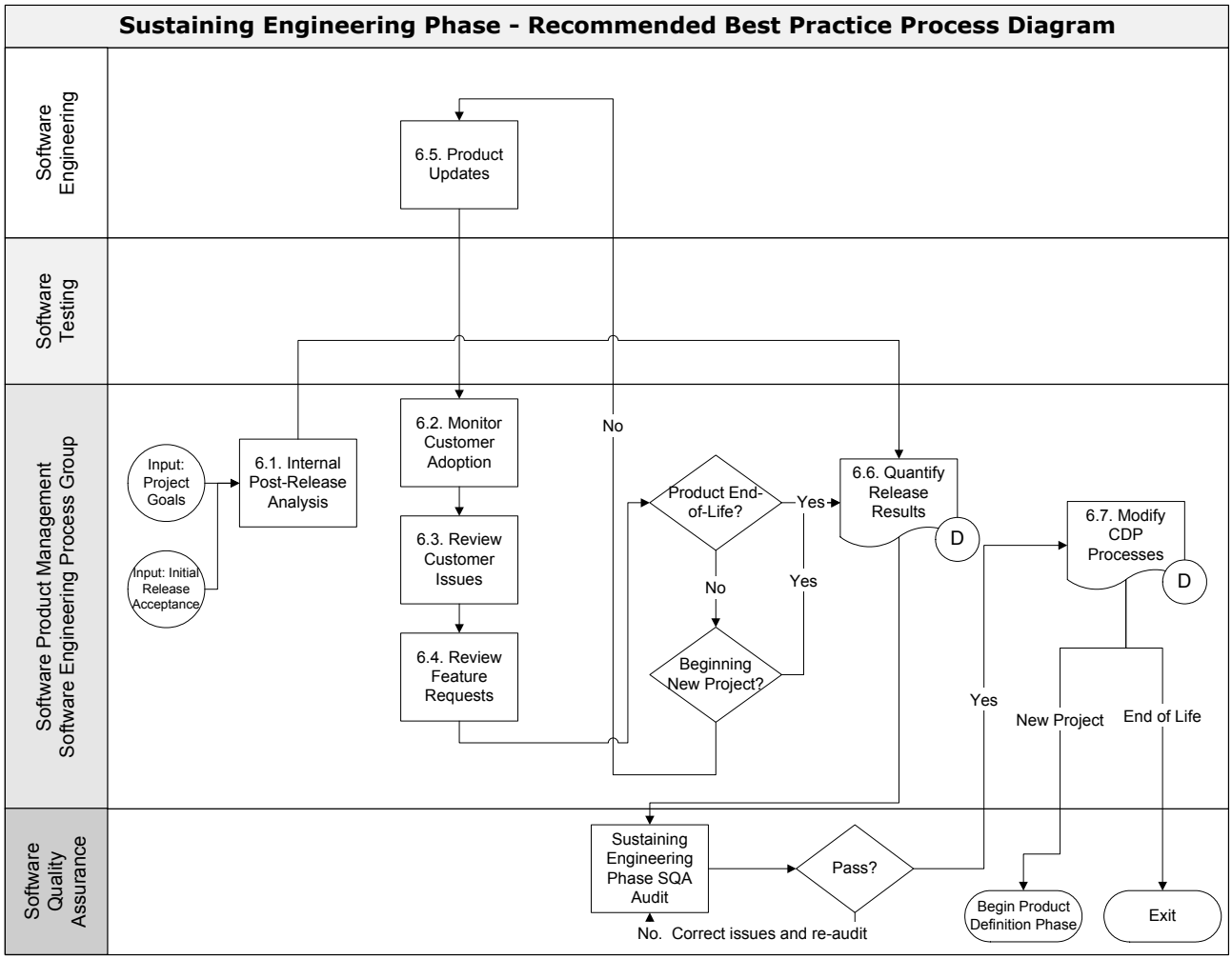


Figure 17. Sustaining Engineering Phase Workflow Process

Please see "Workflow Processes" on page 11 for a full description of all flowchart notation.

Sustaining Engineering Phase Inputs

The following inputs are required to enter the Sustaining Engineering phase:

Successful Completion of Initial Release Acceptance Activities

Many of the Release Acceptance activities begun in the Release phase will be continued and expanded during the Sustaining Engineering phase. However, it is important to confirm the general acceptability of the product and its supporting materials and infrastructure prior to considering the product successfully released. The focus of the Sustaining Engineering phase is on analysis and maintenance; a significant amount of development work is not expected.

Project Goals

In order to effectively analyze the successes and challenges of the development project, the project goals must be measured against the actual results achieved. The documentation created in the Product Definition, Design, Development, and Release Preparation phases defines both strategic and tactical goals for various elements of the project, and should be referenced when conducting Post-Release Analysis activities.

Any goals associated with process improvement initiatives should also be referenced, whether or not they are specific to the current project.

SourceForge Project

The SourceForge project created during the Product Defintion phase should now be populated with the materials completed in the Product Definition, Design, Development, Release Preparation, and Release phases.

SourceForge Component	Contents
At the beginning of the phase	
Project Information	Project-level information such as project description, members, and roles.
Document Manager	A Document Manager folder hierarchy containing final versions of all documents created in the Product Definition through Release phases, including automatically stored revision and review histories for each. All documentation generated as a result of activities conducted during these phases, such as meeting minutes and change requests, should also be stored. The appropriate CDP process templates retrieved from your Software Process Database and stored in your project's Document Manager.
Software Process Database	A Software Process Database used for managing all standard procedures, process templates, checklists, and other materials.
Trackers	Trackers containing all feature requests, bugs, and other artifacts that have been addressed in the product. A Project Logistics Tracker containing the change history of all critical estimates, resources, and risks. Trackers containing all feature requests, bugs, and other artifacts that have been deferred to a later product release. Closure of all product requirements Tracker artifacts (with the exception of any items deferred per formal change request.)
Discussion Forums and Mailing Lists	One or more Discussion Forums and Mailing Lists for discussion of project-related topics, including collaborative decision making on design challenges.
SCM Repository	SCM repository containing completed, frozen, and fully-tested code (with offsite backup for disaster recovery purposes.)
File Release System	One or more completed product build/s released to their final intended audience in the File Release System.
Tracker / Code Associations	Code commits associated with Tracker artifacts for traceability of all relevant product requirements.
Task Manager	One or more projects in the Task Manager to manage various pieces of your Final Project Plan. All Task Manager tasks scheduled for completion prior to product release now marked as completed.

SourceForge Component	Contents
By the end of the phase	
Document Manager	Final, approved versions of all required Sustaining Engineering phase documents stored in the Document Manager.
Reporting	All reports generated using SourceForge Reporting archived in the Document Manager.
Task Manager	All Task Manager tasks scheduled for completion through the duration of the product lifecycle now marked as completed.
Software Process Database	Software Process Database updated with any new or revised procedures, process templates, or other process-related materials.
Project Template	Template project created for re-use in future development projects.

Sustaining Engineering Phase Activities and Deliverables

All activities and deliverables are identified by number in the Sustaining Engineering Phase Workflow Process Diagram.

Descriptions of each activity or deliverable, suggested contents and acceptance criteria, and dependencies on other activities or deliverables are provided with each item. Activities or deliverables for which peer review or a Software Quality Assurance audit is recommended are identified. Considerations for tailoring deliverables to meet specific organizational needs are indicated where appropriate, and recommended uses for SourceForge are included at the end of each section.

Icons

Activities and deliverables for which templates are provided are identified by the symbol shown here. Please see “Activities and Deliverables” on page 13 for a complete list of all symbols used to identify recommended activities and uses for SourceForge.

6.1. Internal Post-Release Analysis

Suggested owner: Software Product Management

After the product is released and the initial acceptance activities are completed, an Internal Post-Release Analysis (also known as a postmortem) should be conducted. The goal of this activity is to analyze the successes of the development project, measure planned versus actual results, and identify areas for future improvement. Specific Internal Post-Release Analysis activities should be described in the Product Launch Strategy document, and should involve all functional groups involved in the project. Internal Post-Release Analysis activities are generally conducted by the Launch Team or similar cross-functional team.

Several types of activities are often conducted during the Internal Post-Release Analysis. The first involves surveying project participants to gather individual feedback on the successes and challenges of the project. This type of input is useful because it provides a broad range of opinions from participants involved in a variety of project activities, at different stages of the CDP, and with differing degrees of responsibility. Feedback can be solicited on specific questions, as well as more general open-ended subjects. Questioning project participants is also an effective way to gather input into subjective topics such as general team dynamics and employee satisfaction, which cannot easily be measured by more formal reporting methods. You may wish to provide anonymous surveys to participants or conduct an open meeting to review issues and successes in a group forum. You can also establish a "process feedback" forum to collect and discuss input throughout the product lifecycle.

In addition to gathering participant input, it is also highly recommended that more formal, quantitative analysis be conducted. Many of the documents completed in the earlier phases of the CDP provide specific project goals that can be objectively measured at this point. Goals established in the Business Requirements Document , Product Requirements Documents, Software Test Plan, Product Launch Strategy, Marketing and Channel Plan, Final Project Plan, and other documents should be assessed against the actual results achieved. If process improvement goals have been set, progress against these goals should also be measured.

SourceForge provides a broad range of reporting functionality to enable this type of Post-Release Analysis. The SourceForge Tracker can provide comprehensive reporting on bug, product requirements, and other artifact activity, such as numbers of bugs opened and closed during each phase of the CDP, average time to closure, and distribution of work among team members. The SourceForge Task Manager can provide comprehensive reporting on adherence to the project schedule, including justifications for any or all deviations. SourceForge Reporting provides additional options. (For additional detail on SourceForge reporting capabilities, please see the *SourceForge Enterprise Edition 4.2 User Guide*.)

Internal Post-Release Analysis can include:

- Surveying project participants
- Issuing anonymous questionnaires
- Conducting open forum meetings to discuss successes and issues
- Discussing issues using SourceForge Forums
- Reviewing actual versus planned progress toward goals established in project documents
- Reviewing progress toward process improvement goals
- Generating and reviewing Tracker, Task Manager, and other SourceForge reports

It is important to note that many of the goals established for the project cannot be measured immediately upon conclusion of the Release phase. Some, such as long-term strategic goals and sales targets, cannot be measured until a pre-defined period of time has passed. Others, such as continuing to monitor customer adoption, bugs, and feature requests, should be done on an ongoing basis throughout the life of the product. These are not considered part of the Internal Post-Release Analysis, and are described as separate activities later in this chapter.

Acceptance criteria for Internal Post-Release Analysis should include:

- Completion of all Internal Post-Release Analysis activities as described in the Product Launch Strategy.
- Audit of Software Quality Assurance function completed.
- Completion of peer review. A Cross-Functional Peer Review Meeting is recommended. The Iterative Document Review and Approval Process is recommended for the Internal Post-Release Analysis document. Surveying project participants also satisfies the peer review requirement.
- Internal Post-Release Analysis document stored in the Document Manager.

Dependencies for Internal Post-Release Analysis include:

- Project Goals
- Product Launch Strategy
- Initial Release Acceptance Activities completed in the Release phase

Recommended Uses for SourceForge:

Use the **SourceForge Document Manager** to store all project documentation containing project goals to be measured during the Internal Post-Release Analysis.

Use **SourceForge Tracker Reporting** to generate any reports related to Tracker artifact activity.

Use **SourceForge Task Manager Reporting** to generate any Task Manager reports measuring adherence to the project plan.

Use the **SourceForge Task Manager** to manage the schedule for all Sustaining Engineering phase deliverables.

Use **SourceForge Discussion Forums and Mailing Lists** to provide and discuss input into Internal Post-Release Analysis issues.

6.2. Monitor Customer Adoption

Suggested Owner: Software Product Management

At the conclusion of the Release phase, activities were conducted to confirm that the product and its supporting materials were being used successfully by the product's early recipients. Activities designed to monitor and facilitate customer adoption of the product, such as conducting surveys, reviewing email or message board feedback, monitoring customer support requests, and conducting post-sales follow-up activities should be continued throughout the Sustaining Engineering phase. For products intended for use by small groups of internal customers only, or as components of larger development projects, informal conversations regarding product satisfaction should also be continued.

As time elapses, additional methods of monitoring customer adoption can be added, such as analyzing sales results, market share, and progress toward achieving other strategic and tactical goals defined in the Business Requirements Document, Marketing and Channel Plan, and other product documents. Sales, Professional Services, and other field staff in direct communication with customers can also be a valuable asset when analyzing adoption of the product. A channel should be provided for field staff to report any issues encountered when selling, installing, upgrading, or supporting the product. Quantitative data such as number of units sold or evaluated should also be reported.

Monitoring Customer Adoption activities can include:

- Continuation of all Release Acceptance Activities begun in the Release phase
- Analyzing sales results, market share, and other quantitative data
- Analyzing progress toward other strategic goals
- Reviewing customer-reported bugs, technical support requests, and feature requests (Described in more detail in sections 6.3 and 6.4 of this chapter.)
- Reviewing issues encountered by Sales and other field staff

Monitoring Customer Adoption, in conjunction with close analysis of Bugs, Support Requests, and Feature Requests will enable you to determine if and when additional supporting materials, support infrastructure, or product patches need to be provided. This information will also be used when quantifying the overall release results, making any necessary modifications to your CDP processes, and planning your next project.

Acceptance criteria for Monitoring Customer Adoption should include:

- Continuation of monitoring activities throughout the product's lifecycle.
- Distribution of additional supporting materials, support infrastructure, and product patches as necessary.
- Publication of periodic reports to the Document Manager.

Dependencies for Monitoring Customer Adoption include:

- Project goals

Recommended Uses for SourceForge:

Use the **SourceForge Task Manager** to manage the schedule for all Sustaining Engineering phase deliverables.

Use the **SourceForge Document Manager** to store all project documentation containing customer adoption targets and other project goals to be measured during Monitoring Customer Adoption.

Use the **SourceForge Tracker** to manage all customer-reported bugs and feature requests.

Use **SourceForge Discussion Forums and Mailing Lists** to provide and discuss input into Customer Adoption issues.

6.3. Review of Customer Issues

Suggested Owner: Software Product Management, Software Testing, or Software Engineering

A key element in analyzing the ongoing success of a product release is monitoring the bugs, technical support requests, and other issues identified by customers. Analyzing the issues raised by users of the product serves several purposes. First, monitoring difficulties encountered by customers when using the product provides you the opportunity to address them by providing new or updated documentation, enhancing support infrastructure, issuing a product update, or performing other remedial actions. Reviewing bugs will help determine whether a patch should be issued in order to correct critical defects. Other types of support requests may be indicators that the user documentation or other materials are confusing or inadequate. They may also highlight technical challenges that are unrelated to your product, but are providing a barrier to successful customer adoption. All of these issues provide an opportunity for you to address them, thereby increasing current customer satisfaction and facilitating future customer adoption.

Reviewing customer-reported issues also enables you to measure the success of the software testing activities conducted during the Development and Release Preparation phases of the CDP. If critical defects are reported that were not identified during your planned testing activities, it may be an indicator that testing processes need to be improved or expanded. Root cause analysis should be used to identify why a bug occured and to ensure the appropriate CDP processes are revised to prevent repetition of the circumstances leading to the issue. Reviewing customer-reported issues also provides data when planning your next release.

Standard Technical Support policies, plus any product-specific considerations defined in the Service and Support Plan, should cover general procedures for addressing customer reported bugs and support requests. Procedures for escalating severe, high-priority bugs should be included, as well as response times and other administrative considerations. While Technical Support is generally responsible for providing customer-facing support, it is important that Software Product Management and other members of the software development team remain involved in the process to ensure that issues encountered with the current product are addressed when planning the next.

Acceptance criteria for Review of Customer Issues should include:

- Review of all customer issues throughout the product's lifecycle.
- Identification and execution of resolution strategies.
- Distribution of additional supporting materials, support infrastructure, and product patches as necessary.
- Identification of customer issues to be addressed in future releases.

Dependencies for Review of Customer Issues include:

- Project goals
- Service and Support Plan

Recommended Uses for SourceForge:

Use the **SourceForge Task Manager** to manage the schedule for all Sustaining Engineering phase deliverables.

Use the **SourceForge Tracker** to manage all customer-reported bugs and feature requests.

Use **SourceForge Discussion Forums and Mailing Lists** to discuss customer issues and resolution strategies.

6.4. Review of Feature Requests

Suggested Owner: Software Product Management

Periodic reviews of customer-submitted feature requests, or requests for product enhancements, should be conducted throughout the Sustaining Engineering and other phases of the CDP. Feature requests submitted by customers provide extremely valuable data when considering or planning your next release. They are a direct reflection of the desires of your target customer base, provide a clear path to increasing sales and customer satisfaction, and can greatly reduce the amount of market research required when planning your next release.

Analyzing the nature of customer feature requests can also help you identify whether they reflect shortcomings in delivered functionality, or simply the desire for expanded product capabilities. Viewed in this way, the number and nature of customer feature requests can help you gauge the success of your product planning processes, and improve future planning efforts. Receiving a large number of feature requests associated with a new product feature, for example, may indicate a fault in your product planning processes, just as a high number of reported bugs may indicate a fault in your development or software testing processes. You may need to increase the amount of customer participation when planning your next release, conduct more lengthy research, or modify some other component of your requirements definition process in order to more accurately capture customer needs.

It is important to encourage users to define the problem they wish to solve, rather than specify a feature. This way you are free to propose the best solution that is consistent with your product design strategy. Please reference “Managing Product Requirements Using the SourceForge Tracker” on page 280 for recommended best practices on managing feature requests and product requirements using the SourceForge Tracker.

Acceptance criteria for Review of Feature Requests should include:

- Periodic review of feature requests throughout the product’s lifecycle.
- Identification of customer issues to be addressed in future releases.

Dependencies for Review of Feature Requests include:

- Project goals

Recommended Uses for SourceForge:

Use the **SourceForge Task Manager** to manage the schedule for all Sustaining Engineering phase deliverables.

Use the **SourceForge Tracker** to manage all customer-reported bugs and feature requests.

Use **SourceForge Discussion Forums and Mailing Lists** to discuss customer feature requests and resolution strategies.

6.5. Product Updates

Suggested Owner: Software Engineering

Throughout the Sustaining Engineering phase, severe bugs or other issues may arise requiring a product update to be issued. These are also known as service packs, patches, Temporary Program Fixes (TPFs), bug fix updates, or simply updates. They may replace small sections of code or entire product components. They generally consist of new or modified code that the user can apply over an existing installation, without having to reinstall the current version or install a new version of the product. If the update replaces the whole product and cannot be delivered without requiring the product to be reinstalled and reconfigured, it is generally referred to as a new release, not a product update.

Based on ongoing analysis of user issues and feature requests, strategic considerations, and resource availability, you may choose to create and release product updates in response to critical issues as they arise, or release planned, periodic updates on a regular schedule. Issue-driven product updates are often appropriate when a critical bug is identified that is preventing or severely limiting customer adoption of the product. In such cases, the product update may be provided only to the customer reporting the issue, to a wider, selected audience, or to the entire customer base. Considerations such as the number of customers reporting the issue, its specificity to a particular customer's use case or customized installation, and its potential to impact a wide audience of users can help you determine the best method of distribution.

Alternatively or additionally, you may plan to issue product updates on a regular basis. This method will enable you to more proactively address customer-reported issues, without requiring customers to wait a lengthy period of time for the next full product release. Issuing regular product updates is also common with products whose value is limited unless they are continually updated, such as antivirus products.

A product update policy document should be created, or incorporated into your Software Test Plan or Product Defect Prioritization Policy to define how customer-reported issues will be prioritized and addressed during all phases of the CDP. The policy document should also cover the process by which product updates will or will not be incorporated into an upcoming release.

Acceptance criteria for Product Updates should include:

- Completion of peer review. Code review is recommended. Additional peer review criteria are included in the Code Review Checklist.
- Product updates issued as needed throughout the Sustaining Engineering phase

Dependencies for Product Updates include:

- Review of Customer Issues
- Review of Feature Requests
- Project goals

Recommended Uses for SourceForge:

Use the **SourceForge Task Manager** to manage the schedule for all Sustaining Engineering phase deliverables.

Use CVS as your Software Configuration Management (SCM) tool to manage all code developed during the Sustaining Engineering and other phases, including code developed for product updates. Use the **SourceForge SCM interface** to browse the contents of your SCM repository.

Associate code commits for product updates with the applicable bugs **Tracker** artifacts.

Use the **SourceForge Tracker** to manage all customer-reported bugs and feature requests that will be reviewed as the basis for issuing product updates.

Use **SourceForge Discussion Forums and Mailing Lists** to discuss customer feature requests and resolution strategies.

Use the **SourceForge File Release System** to release product updates for distribution to their intended audiences.

6.6. Quantify Release Results

Suggested Owner: Software Product Management

A significant goal of the Sustaining Engineering phase is to identify opportunities to improve your development and other processes. Since the Sustaining Engineering phase of the CDP will continue until the product's end-of-life, which may span a period of many years, the point at which you should formally quantify your release results and modify your CDP processes will be largely based on the timeline for beginning the next project.

Conducting the post-release analysis activities described in the previous sections of this chapter will provide a broad range of qualitative and quantitative data. Once you have completed all activities to a satisfactory degree (keeping in mind that many activities are continued throughout the life of the product), and are ready to begin developing an action plan for specific process improvement activities, a report should be prepared consolidating the results of all post-release analysis activities into a single location. This location may be a single document, or a Document Manager folder containing a series of individual documents in an easily identifiable location.

The goal of this activity is to provide source data for use in developing process improvement plans. Include proposed solutions to identified issues, if available, as well as summary information describing trends, highlights, and any significant accomplishments or issues. A summary document of this type is often provided to the Management Approval Team and other key stakeholders as a final post-project report. Once completed the summary report should be stored in the Software Process Database for easy reference by all staff.

Quantification of Release Results can include:

Results of Internal Post-Release Analysis

For participant survey responses, you may wish to aggregate data and provide summary results, or provide raw data with all participant responses. Include progress toward achieving project goals, and reference any Forums used for discussion. Include or reference reports generated using the SourceForge Tracker, Task Manager, or Reporting. In all cases, provide overview information describing trends, highlights, and proposed solutions to significant issues.

Customer Adoption Trends and Statistics

Report all quantitative data such as sales figures and market share, as well as overview information summarizing issues frequently encountered by Sales and other field staff. Include progress toward achieving project goals associated with customer adoption, as well as any remedial actions already taken to respond to identified issues.

Summary Results of Reported Customer Issues and Feature Requests

Include summary information such as number and severity of issues, categorization information (by feature, customer segment, or other categories), and any analysis of the nature and sources of the issues (requests for new functionality, defects in existing features). If analysis of customer issues and feature requests indicates a potential area for improvement in your product planning, development, or software testing processes, include the proposed solution.

Analysis of Planning Data versus Actual Results

Each project generates size, effort, and cost estimates during the planning stages. These estimates are then tracked using the Project Logistics Tracker throughout the project. Include an analysis of the estimates versus the actual size, effort, and cost of each project. This information is used when planning future projects.

Product Updates

Include any product updates either planned or already distributed, as well as the reason for the update.

Since many of the post-release analysis activities are ongoing, it is highly recommended that release results be quantified periodically throughout the Sustaining Engineering phase. At minimum, you should plan on consolidating and reporting the results of your post-release analysis activities twice: once in preparation for the Product Definition phase of your next project, and again at the product's end of life. All such reports should be stored in the Software Process Database.

Acceptance criteria for Quantification of Release Results should include:

- Consolidation and reporting the results of all post-release analysis activities.
- Completion of peer review. The Iterative Document Review and Approval Process is recommended.
- Periodic renewal of results.

Dependencies for Quantification of Release Results include:

- Internal Post-Release Analysis
- Monitoring of Customer Adoption
- Review of Customer Issues
- Review of Feature Requests
- Product Updates

Software Quality Assurance Audits

Software Quality Assurance audits are recommended for a number of the components of quantified release results. Audits of processes used to quantify release results and activities conducted as a result of the quantified release results are also recommended.

Reference the Software Quality Assurance section of the Product Plan and the Software Quality Assurance audit checklists for each CDP phase for a complete list of all artifacts, processes, and activities to be audited.

Recommended Uses for SourceForge:

Use the **SourceForge Task Manager** to manage the schedule for all Sustaining Engineering phase deliverables.

Use the **SourceForge Document Manager** to store the final Quantification of Release Results documents, and to manage the Iterative Document Review and Approval Process. See Figure 21 on page 276 for additional detail on this process.

6.7. Modify CDP Processes

Suggested Owner: Software Product Management, Software Engineering, Software Testing, Software Quality Assurance, Software Engineering Process Group

Many of the activities described in this chapter involve comparing the planned versus actual results of your development project. The goal of these activities is to identify areas where making improvements to your existing processes will help you increase the predictability of your projects, improve efficiency and productivity, decrease costs and time-to-market, and better satisfy the needs of your internal and external customers. By translating the results of your analysis efforts into an action plan for process improvement, the time, effort and expense you have invested will pay off in the greater success of future projects. A continuous cycle of process improvement is achieved. (See Figure 2, “SourceForge Collaborative Development Process,” on page 9.)

The CDP provides a comprehensive model against which to measure the results of your current project. Reviewing the activities, deliverables, and goals for each phase of the CDP with the activities, deliverables, and results achieved in your project is a good way to identify areas for future process improvement. If you have followed the CDP in your previous project, this can also be an opportunity to modify any aspects of it where customization will better meet the specific needs of your organization.

Once you have analyzed the available data and prepared an action plan for improving the organization’s set of standard processes, you should present your proposal to the Management Approval Team, or whoever is responsible for sponsoring organizational process improvement activities. This will ensure that you have the support necessary to implement your proposed changes. Other groups or individuals impacted by the proposed change should also peer review the proposal. The project plan for implementing the proposed changes should be managed in the Software Process Database using the Task Manager to provide organization-wide visibility into the status of process-improvement activities.

For organizations pursuing SW-CMM goals, the CDP provides SW-CMM Level 2 and 3 compliant processes that will enable you to leverage SourceForge ensure compliance. Now is an ideal time to review your progress against SW-CMM goals, and engage the services of an SEI-authorized lead assessor for consultation or formal assessment services if desired. Please see “Preparing for a CMM-Based Appraisal using SourceForge” on page 28 for additional information on preparing for a CMM-Based Appraisal for Internal Process Improvement (CBA-IPI.)

After you have completed your first development project in SourceForge, you may wish to create a template project from your completed project. If you are satisfied that the project structure you established meets your needs, you can create a template project to enforce standardization and simplify new project creation going forward.

Acceptance criteria for Modification of CDP Processes should include:

- Completed action plan for process improvement activities.
- Peer review by groups and individuals who may be impacted by the proposed process changes. The Iterative Document Review and Approval and the Cross-Functional Peer Review Meeting methods of peer review are recommended.
- Presentation to Management Approval Team and approval to move forward with proposed changes. This presentation constitutes peer review for this activity.
- Management of all process improvement activities and deliverables in the Software Process Database project.

Dependencies for Modification of CDP Processes include:

- Quantification of Release Results

Software Quality Assurance Audits

Software Quality Assurance audits are recommended for a number of the components of modified CDP processes. Audits of processes used to modify CDP processes and activities conducted as a result of the modified CDP processes are also recommended.

Reference the Software Quality Assurance section of the Product Plan and the Software Quality Assurance audit checklists for each CDP phase for a complete list of all artifacts, processes, and activities to be audited.

Recommended Uses for SourceForge:

Use the **SourceForge Task Manager** to manage the schedule for all Sustaining Engineering phase deliverables. Manage process improvement activities in the Software Process Database project.

Use the **SourceForge Document Manager** to store the final Process Improvement Action plan, and to manage the Iterative Document Review and Approval Process. See Figure 21 on page 276 for additional detail on this process.

Use **SourceForge project templating** to create a template project to enforce standardization and simplify future new project creation.

Use **all SourceForge features** as descibed thoughout the CDP when planning future process improvement activities.

Sustaining Engineering Phase Exit Criteria

Post-release analysis activities are conducted on an ongoing basis with the goal of identifying areas for future process improvement. Exit criteria activities should be conducted a minimum of twice: once in preparation for the Product Definition phase of your next project, and again at the product's end of life.

A Sustaining Engineering Phase Exit Review meeting should be held with key stakeholders from all impacted organizations. The following checklist should be used to ensure all exit criteria are met.

✓	Sustaining Engineering Phase Exit Checklist
❑	SourceForge project now containing all elements specified on page 222.
❑	Internal Post-Release Analysis completed and appropriate artifacts peer-reviewed Final checklists and other documents stored in the Document Manager.
❑	Customer adoption monitoring activities completed.
❑	Customer issues reviewed and addressed.
❑	Customer feature requests reviewed and addressed.
❑	Product updates issued as needed. Code review completed for each product update prior to issue.
❑	Release results quantified, consolidated, peer-reviewed using the Iterative Document Review and Approval Process, published to the SourceForge Document Manager and locked.
❑	CDP processes modified as needed. Process improvement action plan completed, presented to the Management Approval Team, approved, and published to the SourceForge Document Manager.
❑	The schedule for all process improvement activities managed in the Task Manager of the Software Process Database.
❑	All project training activities scheduled for completion in the Sustaining Engineering phase completed.
❑	All Training Evaluation Forms completed and posted to the Organizational Training project on SourceForge.
❑	Sustaining Engineering phase Software Quality Assurance audit checklist completed. Checklist posted to the Document Manager.
❑	Audit of the Software Quality Assurance function completed by the SEPG.

✓	Sustaining Engineering Phase Exit Checklist
☐	Periodic engineering review meetings conducted.
☐	Regular Launch Team meetings conducted, including reviews of the Task Manager and the Project Logistics Tracker.
☐	Periodic reviews of subcontractor progress completed, if applicable.
☐	Sustaining Engineering Phase Exit Review successfully completed using this checklist. Checklist posted to the Document Manager.

Using SourceForge to Support the Sustaining Engineering Phase

SourceForge supports the goals of the Sustaining Engineering phase by allowing you to:

- Use CVS as your Software Configuration Management (SCM) tool to manage all code developed for patches. Use the **SourceForge SCM interface** to browse the contents of your SCM repository.
- Associate code commits with bugs, feature requests, and other Tracker artifiacts using **Tracker / Code Associations.**
- Release product patches internally for distribution to their final intended audiences using the **File Release System.**
- Manage bugs and feature requests using the **Tracker.**
- Generate post-release analysis reports using **Tracker Reporting**, **Task Manager Reporting**, and other types of SourceForge **Reporting**.
- Enable management visibility and control of all Sustaining Engineering phase actions and deliverables using the **Task Manager.**
- Review all documentation stored in the **Document Manager** as reference for the post-release analysis activities.
- See *SourceForge and the Capability Maturity Model for Software, A Supplement to the SourceForge Collaborative Development Process* for additional detail on using SourceForge and the CDP to satisfy SW-CMM Level 2 and 3 requirements.

Sustaining Engineering Phase Templates

The following table provides a quick reference to all templates used for the Sustaining Engineering phase deliverables and activities.

All templates are provided on the CD included with this manual, and at the following URL: *https://www.vasoftware.com/cdp/*

Table 20. Sustianing Engineering Phase Templates

Ref#	Template	File Location
6.1.1	Internal Post-Release Survey	\sustain\[6.1.1]survey.dot \sustain\[6.1.1]survey.pdf
6.8.1	SQA Checklist Sustaining Engineering Phase	\sustain\[6.8.1]sqasustain.dot \sustain\[6.8.1]sqasustain.pdf
6.9.1	SQA Checklist SQA Function	\sustain\[6.9.1]sqasqafunction.dot \sustain\[6.9.1]sqasqafunction.pdf
6.10.1	Sustaining Engineering Phase Exit Checklist	\sustain\[6.10.1]sustainexit.dot \sustain\[6.10.1]sustainexit.pdf

Please reference “Numbering” on page 10 for additional information on document numbering conventions.

CHAPTER 9

SourceForge Best Practices

This chapter includes the following information:

- Best Practices Overview
- Managing your Software Process Database using SourceForge
- Managing your Peer Review Processes using SourceForge
- Managing Software Quality Assurance Audits using SourceForge
- Managing your Organization's Training Programs using SourceForge
- Managing Project Logistics using the SourceForge Tracker and the Monitoring Function
- Managing Subcontractors using SourceForge
- Managing the Document Lifecycle using the SourceForge Document Manager
- Managing Product Requirements using the SourceForge Tracker
- Managing your Project Plan using the SourceForge Task Manager
- Best Practices for Software Configuration Management using SourceForge
- Best Practices for Software Development and Quality Assurance

Best Practices Overview

The materials in this chapter provide extensive guided best practices for using SourceForge to achieve specific SourceForge CDP and SW-CMM process objectives. These guided best practices are intended as a supplement to the standard SourceForge user documentation. They assume the reader has a working knowledge of the fundamental uses of SourceForge. If you need help understanding the basic functions of SourceForge, please consult the *SourceForge Enterprise Edition 4.2 User Guide* and the *SourceForge Enterprise Edition 4.2 Installation and Administration Guide* or contact your VA Software representative regarding customized training programs.

The guided best practices in this chapter suggest ways to use SourceForge to maximize ease of use, leverage the full power of the application, and facilitate process improvement. They are recommendations only; you may wish to modify some or all of the suggested practices to better meet the specific needs of your organization.

Sample Policies and Procedures

Policies and procedures for conducting various activities are included with many of the best practices in this chapter. Procedures are identified by the following boxed notation:

PROCEDURE FOR CONDUCTING THE ITERATIVE DOCUMENT REVIEW AND APPROVAL PROCESS

Overview

The Iterative Document Review and Approval Process should be conducted for all documents created throughout the phases of the CDP.

For Owners of Documents Subject to Iterative Document Review and Approval

1. The My Tasks section of your My Page will indicate when peer reviews assigned to you are scheduled.
2. The Task details should indicate that the type of peer review to be conducted is Iterative Document Review and Approval Process. If it does not, check the Product Plan peer review section to confirm.

Figure 18. Sample Procedure Notation

The procedures provide recommendations for leveraging the functionality of SourceForge to support various activities, and represent the procedures used internally by the VA Software Product Development organization and other software development organizations using SourceForge. However, they are intended to be flexible enough to provide guidance, yet be customizable to the specific needs of your organization. Customizable document templates are provided for each procedure described in the CDP materials.

Managing your Software Process Database using the SourceForge Document Manager

One of the most valuable resources for an organization undertaking a formal process improvement program is a Software Process Database. The purpose of a Software Process Database is to collect and maintain all of the organization's process-related materials in a centralized location, enabling project members to quickly and easily locate the correct process templates, instructions, and other materials needed to perform their job functions. The Software Process Database can also be used as a repository for a variety of analytical data collected from previous projects. It then becomes a resource from which data can be gathered to help increase the accuracy of future planning efforts. The SourceForge Document Manager provides an excellent resource for establishing and maintaining your Software Process Database.

The Software Process Database is generally managed by the person or group responsible for the organization's process improvement program. This group is sometimes referred to as a Software Engineering Process Group. (See "Software Engineering Process Group" on page 50 for additional information on this function.) In addition to storing the organization's standard process library, the Software Process Database can be used to actively manage the process-related activities of the Software Engineering Process Group. In this way, it becomes the central repository for all staff to access both process-related materials and the latest news and other information on the progress of the organization's process improvement program.

The contents stored in the Software Process Database may include:

- CDP materials
- Process templates
- Checklists
- Procedures and policy documents
- Instructions on completing various process-related activities
- Analytical data collected from prior projects
- News and Forums used to communicate status of process-related activities and communicate with the Software Engineering Process Group
- Current status of the Software Engineering Process Group's activities and deliverables in the Task Manager
- References to the planning and estimation data for each development project

The Software Process Database is best managed as a separate SourceForge project, to easily keep the process templates in the organization's standard process library separate from those containing project-specific data. All managers responsible for process-related activities on a project level should be made project members and assigned appropriate permissions using Role-based Access Control. You may wish to use the Lock Document feature on all standard

process templates and organizational procedures to prevent unauthorized staff from making modifications.

When beginning each new development project, the responsible managers should retrieve the appropriate templates from the Software Process Database and store them in the SourceForge project that will be used to manage the product's development.

Role-based Access Control

It is important that all staff who will be responsible for process-related activities are made project members in the Software Process Database. They should be assigned appropriate permissions, including access to the Document Manager at minimum. Senior management or other staff who are sponsoring, monitoring, or otherwise interested in the process-related activities of the organization should also be made project members and assigned appropriate roles. You can also set default application access permissions to grant view and/or submit access to all members of specific user classes, regardless of their project membership.

Since the Software Engineering Process Group has oversight responsiblity for the process activities of all development projects, it is also important that representatives from this group are made project members in each development project's SourceForge project.

SW-CMM NOTE

The SW-CMM requires that the organization's software process database be established and maintained. Following the guidelines in this section satisfy this requirement.

See *SourceForge and the Capability Maturity Model for Software, A Supplement to the SourceForge Collaborative Development Process* for specific SW-CMM requirements satisfied by a Software Process Database.

Managing your Peer Review Processes using SourceForge

Peer review is a key element throughout every phase of the CDP and is applied to nearly all documents, software code, and other artifacts. Conducting peer review has many benefits, including collecting input from stakeholders in a variety of functional areas, identifying defects in work products, and facilitating intergroup coordination.

SourceForge supports a variety of types of peer review, each of which is recommended for different types of artifacts. The type of peer review you choose for each artifact will depend on criteria such as the importance of the document, the number of desired reviewers, the time available for review, and the software and other tools available to support review processes.

The following procedure provides recommendations for managing the peer review process on an organizational level.

ORGANIZATIONAL PEER REVIEW POLICY
Overview

Artifacts that will undergo peer review are identified in the Product Plan. All development projects must identify, in advance, artifacts to undergo peer review. Generally, all documents and a sampling of software code and other artifacts are identified for peer review. It is the responsibility of the author of the Product Plan to indicate the level of peer review each artifact is expected to undergo. Through peer review of the Product Plan itself, the peer review proposal is approved by the key stakeholders throughout the organization. The Software Engineering Process Group is a key reviewer to ensure adherance to peer review standards.

For Product Plan Owners

1. Familiarize yourself with the various types of peer reviews and the suggested uses for each.
2. Reference the CDP section describing the artifacts you wish to review.
3. Determine the most appropriate method of peer review for each artifact and indicate the selected methods in the Product Plan.
4. Through peer review of the Product Plan itself, ensure that key stakeholders support the peer review proposal. Make any necessary changes until the document is finalized.
5. Ensure that the required peer reviews are included in the Project Plan and managed to completion using the Task Manager. The type of peer review should be included in the Task Details for easy reference by the owner of the artifacts to be peer reviewed.

For Owners of Artifacts Subject to Peer Review

1. The My Tasks section of your My Page will indicate when peer reviews assigned to you are scheduled.
2. The Task Details section should indicate the type of peer review to be conducted. If it does not, check the Product Plan peer review section.
3. Review the Peer Review Procedure for the type of peer review you will be conducting. Peer Review Procedures are posted in the Software Process Database.
4. All document templates contain peer review checklists within the templates themselves, plus instructions for completing them. For documents subject to the Iterative Document Review and Approval process, simply reference the Peer Review and Acceptance Criteria section for your document.
5. For code reviews or walkthroughs, obtain the correct peer review checklist from your Software Process Database. Each checklist contains instructions for completing it.
6. Once your peer review is completed, ensure that the results are posted to SourceForge according to the instructions provided.
7. Update the task in the Task Manager to indicate that your peer review is completed. Also record the time you spent managing the peer review process.
8. If you need training on how to conduct peer reviews, inform your manager.

Types of Peer Review

Iterative Document Review and Approval Process

The Iterative Document Review and Approval process describes a method for using the SourceForge Document Manager to actively manage a document's review and approval. Upon submission of any new document, you can indicate whether you would like the document routed for review. If so, you will then be prompted to provide the list of required and optional reviewers, the timeline for returning review comments, and other details about the review process. Reviewers should be instructed to complete the peer review checklists embedded in each CDP document template to ensure consistent reviews.

The Document Manager automatically archives all review comments for later reference, maintains a log of the document version with which review comments are associated, and will notify the submitter when new review comments are posted. The Iterative Document Review and Approval Process is highly recommended for all documents created throughout the phases of the CDP.

PROCEDURE FOR CONDUCTING THE ITERATIVE DOCUMENT REVIEW AND APPROVAL PROCESS

Overview

The Iterative Document Review and Approval Process should be conducted for all documents created throughout the phases of the CDP.

For Owners of Documents Subject to Iterative Document Review and Approval

1. The My Tasks section on your My Page will indicate when peer reviews assigned to you are scheduled.
2. The Task Details should indicate that the type of peer review to be conducted is Iterative Document Review and Approval Process. If it does not, check the Product Plan peer review section to confirm.
3. When your document is ready for review, submit it to the Document Manager.
4. After the document is submitted, navigate to the Review tab and click Start a Review.
5. The Document Manager will prompt you to select required and optional reviewers. Ensure that all required reviewers indicated in the Required Reviewers section of your document are selected and identified as required reviewers. If a reviewer does not appear in the drop-down list on this page, contact a SourceForge project administrator. All reviewers must have the document edit permission.
6. Fill in all remaining fields to provide email message text, review due date, and other details.
7. In the message text, instruct reviewers to attach review copies to the review.
8. When all reviews are completed, update the task in the Task Manager to indicate that your peer review is completed. Also record the time you spent managing the peer review process.

For Reviewers of Documents Subject to Iterative Document Review and Approval

1. You will receive an email when your review of a document has been requested. The email will provide the relevant details such as document URL and review due date.
2. A list of all documents requiring your review also appears on your My Page in the Documents Awaiting Review section.
3. Review the document and complete the Peer Review appendix.
4. Follow the document link from your My Page to access the Submit a Response area in the Document Manager.
5. Attach your review copy and submit any additional review comments if desired.

Cross-Functional Peer Review Meetings

In some cases, you may wish to supplement the Iterative Document Review and Approval Process with a live review meeting involving key stakeholders from all impacted functional groups. This method is recommended for critical documents with a broad organizational impact such as the Product Plan. In such cases where it is critical to gain consensus on important issues from a large group of stakeholders, conducting a live meeting is often beneficial.

It may be necessary to repeat the Iterative Document Review and Approval Process and conduct one or more Cross-Functional Peer Review Meetings to finalize complex documents. Due to the time and resources required for cross-functional review meetings, however, this method is not prescribed for all documents.

PROCEDURE FOR CONDUCTING CROSS-FUNCTIONAL PEER REVIEW MEETINGS

Overview

Cross-functional review meetings are the recommended form of Peer Review for critical documents with broad organizational impact such as the Product Plan.

For Owners of Artifacts Subject to Cross-Functional Peer Review Meetings

1. The My Tasks section of your My Page will indicate when peer reviews assigned to you are scheduled.
2. The Task Details should indicate that the type of peer review to be conducted is Cross-Functional Peer Review Meeting. If it does not, check the Product Plan Peer Review section to confirm.
3. Begin by distributing your document for review following the Iterative Document Review and Approval method of peer review. Incorporate preliminary feedback and identify any critical, unresolved issues to address during the meeting.
4. Schedule the meeting, ensuring that all required reviewers of your artifact confirm attendance, including representation from the Management Approval Team.
5. Retrieve the Cross-Functional Peer Review Meeting checklist from the Software Process Database. Fill in the checklist with all outstanding issues remaining from the Iterative Document Review and Approval Process.
6. Identify a moderator and a scribe for the meeting.
7. When conducting the meeting, the goal is to discuss and resolve each item on the checklist.
8. Note any issues that cannot be resolved. These should be escalated to the Management Approval Team for resolution.
9. Incorporate any changes to the document and resubmit for a final Iterative Document Review and Approval Process.
10. This process may need to be repeated for especially complex documents.
11. When completed, post the Cross-Functional Peer Review Meeting checklist to the Document Manager.
12. Update the task in the Task Manager to indicate that your peer review is completed. Also record the time you spent managing the peer review process.

Specific peer review questions are included as an appendix to each CDP document template.

Code Review

The code review form of Peer Review consists of one or more software engineers reviewing and rating another software engineer's code on an established list of criteria. Following the code review, the software engineer then makes any necessary corrections to their code. If it was deemed necessary during the first review (based on severity of issues or other criteria), a follow-up peer review is then conducted. This process is repeated until the quality of all code is determined to be satisfactory by the peer review team.

PROCEDURE FOR CONDUCTING CODE REVIEW

Overview

Code review is the recommended method of Peer Review for reviewing each module of software code produced during the Development and Release Preparation phases of the CDP. It is a required acceptance criteria for Technical Unit Testing and may be used as needed during other stages of development and testing to resolve bugs and other code-related issues.

For Software Engineers Conducting Code Review

1. The My Tasks section of your My Page will indicate when peer reviews assigned to you are scheduled.
2. The Task Details should indicate that the type of peer review to be conducted is code review. If it does not, check the Product Plan Peer Review section to confirm.
3. Retrieve the Code Review Checklist from the Software Process Database.
4. The Code Review Checklist provides the Pre-Review, Review, and Post-Review activities to be conducted, as well as a checklist to capture results of the code review.
5. Conduct the code review, ensuring that all activities indicated in the checklist are completed.
6. You may need to repeat the code review based on the results captured in the checklist.
7. When completed, post each code review checklist to the Document Manager.
8. Update the task in the Task Manager to indicate that your peer review is completed. Also record the time you spent managing the peer review process.

Buddy Reviews

"Buddy reviews" are often useful when a limited amount of review time is available. This form of review is more informal and involves identifying one or more peers to review a document or other artifact without following one of the formal processes described above.

"Buddy reviews" can also be highly effective when the content to be reviewed is in its early conceptual stages, when the quantity of material to be reviewed is limited, or when the selected reviewer is a subject matter expert. Due to the more comprehensive review offered by the other processes described in this section, "buddy reviews" are not specified as the recommended form of review for any artifacts produced throughout the CDP; however, you may choose to substitute this form of review as appropriate.

SW-CMM NOTE

The SW-CMM Level 3 Key Process Area Peer Reviews requires peer review of a list of specific artifacts. Given the ease of using the SourceForge Document Manager to facilitate document review and approval, and its capacity to manage the peer review checklists and other artifacts associated with Peer Review, VA Software highly recommends conducting some form of peer review on all documents and work products produced throughout the CDP.

See *SourceForge and the Capability Maturity Model for Software, A Supplement to the SourceForge Collaborative Development Process* for specific SW-CMM requirements satisfied by an effective peer review process.

Table 21. Peer Review Templates

Ref#	Template	File Location
7.1.1	Procedure for Conducting Peer Review	\crosslife\[7.1.1]peerreviewpolicy.dot \crosslife\[7.1.1]peerreviewpolicy.pdf
7.2.1	Procedure for Conducting Iterative Document Review and Approval Process	\crosslife\[7.2.1]peerreviewdoc.dot \crosslife\[7.2.1]peerreviewdoc.pdf
7.3.1	Procedure for Conducting Cross-Functional Peer Review Meetings	\crosslife\[7.3.1]peerreviewmtg.dot \crosslife\[7.3.1]peerreviewmtg.pdf
7.4.1	Procedure for Conducting Code Review	\crosslife\[7.4.1]peerreviewcode.dot \crosslife\[7.4.1]peerreviewcode.pdf
7.5.1	Cross-Functional Peer Review Meeting Checklist	\crosslife\[7.5.1]crossfun.dot \crosslife\[7.5.1]crossfun.pdf
7.6.1	Code Review Checklist	\crosslife\[7.6.1]codereview.dot \crosslife\[7.6.1]codereview.pdf

Managing Software Quality Assurance Audits using SourceForge

Staff performing the Software Quality Assurance function are responsible for conducting periodic audits on the quality of various project artifacts. The purpose of these periodic audits is to confirm that the quality of the processes, activities and deliverables produced throughout the CDP is sufficiently high to meet the specifications set forth by company standards and established project-specific criteria. Audits are also conducted to assess how well the organization's standard processes, or project-specific tailored versions of these processes, are being followed. The Software Quality Assurance function can be viewed as a continuous monitor of planned versus actual results.

Results from Software Quality Assurance audits are distributed to the manager responsible for the project. They are also reviewed on a periodic basis with the Management Approval Team. It is important that the Software Quality Assurance team have a direct reporting channel to the Management Approval Team, independent of any other reporting structure. This independence allows the Software Quality Assurance team to act as the eyes and ears of senior management, and ensures that audit results and potential issues are quickly communicated at the appropriate level.

In some organizations, for example large aerospace and defense contractors, a group dedicated to conducting quality audits and similar activities on a full-time basis exists. In others where a full-time Software Quality Assurance organization is not necessary or cost-effective, the audit function may be performed on an as-needed basis by staff normally assigned to other functions.

The Software Quality Assurance plan for each project is completed in the Product Plan. At that time, the owner of the Product Plan is responsible for ensuring that both the artifacts to be audited and the staff members who will perform the audits are identified. Training should be secured if necessary for staff to perform the audit function.

Artifacts for which Software Quality Assurance audits are recommended are identified throughout the CDP manual by the icon shown above. It should be noted, however, that Software Quality Assurance audits are conducted on a number of activities and processes that are not associated with a specific deliverable, such as effective ongoing management of subcontractors. Refer to the Product Plan template and the Software Quality Assurance audit checklists provided for each CDP phase for a comprehensive list of activities, deliverables, and processes for which audits are recommended.

The following procedure provides recommendations for managing the Software Quality Assurance process on an organizational level.

ORGANIZATIONAL SOFTWARE QUALITY ASSURANCE POLICY

Overview

Software Quality Assurance audits are conducted on a variety of work products produced throughout the product lifecycle. A sampling of documents, code, and other artifacts are selected for review and identified in the Product Plan. Through Peer Review of the Product Plan, the Software Quality Assurance proposal is approved by the key stakeholders throughout the organization.

For Product Plan Owners

1. Familiarize yourself with the Software Quality Assurance processes and checklists, and the standard set of artifacts subject to audits.
2. Based on the tailoring guidelines in the Product Plan, identify the specific audits that will be conducted during each phase of the project.
3. Identify the staff who will be responsible for performing the audits and related activities.
4. Through peer review of the Product Plan, ensure that key stakeholders support the Software Quality Assurance proposal. Make any necessary changes until the document is finalized.
5. Ensure that the required audits are included in the Project Plan and managed to completion using the Task Manager.

For Software Quality Assurance Team Members

1. The My Tasks section of your My Page will indicate when audits assigned to you are scheduled.
2. Review the Software Quality Assurance procedure and retrieve the appropriate checklists from the Software Process Database.
3. Each checklist includes instructions for completing it. Perform the audit and record your results.
4. After the audit is completed, post the completed checklist to the Document Manager and update the associated task in the Task Manager with the document reference.
5. Report the results as follows:
    - If any issues were identified, inform the Product Manager and the person or team responsible for the item in question immediately.
    - If any high-severity issues were identified, also inform the Management Approval Team.
    - Ensure that the action items required to resolve any issues are assigned.
6. Once any issues are resolved, confirm that the results are satisfactory. Update the checklist and repost. Update the task status accordingly.
7. Notify the Management Approval Team as specified in the Product Plan.
8. If you need training on conducting Software Quality Assurance audits, inform your manager.

All Software Quality Assurance audits must be successfully completed and any issues resolved before each Phase Exit Review can be passed.

SW-CMM NOTE

The SW-CMM Level 2 Key Process Area Software Quality Assurance requires periodic audits as specified in this section.

See *SourceForge and the Capability Maturity Model for Software, A Supplement to the SourceForge Collaborative Development Process* for specific SW-CMM requirements satisfied by an effective Software Quality Assurance process.

Table 22. Software Quality Assurance Templates

Ref#	Template	File Location
7.7.1	Software Quality Assurance Policy	\crosslife\[7.7.1]sqapolicy.dot \crosslife\[7.7.1]sqapolicy.pdf
1.12.1	SQA Checklist Product Definition Phase	\proddef\[1.12.1]sqaproddef.dot \proddef\[1.12.1]sqaproddef.pdf
2.9.1	SQA Checklist Design Phase	\design\[2.9.1]sqadesign.dot \design\[2.9.1]sqadesign.pdf
3.14.1	SQA Checklist Development Phase	\dev\[3.14.1]sqadev.dot \dev\[3.14.1]sqadev.pdf
4.11.1	SQA Checklist Release Preparation Phase	\relprep\[4.11.1]sqarelprep.dot \relprep\[4.11.1]sqarelprep.pdf
5.6.1	SQA Checklist Release Phase	\release\[5.6.1]sqarelease.dot \release\[5.6.1]sqarelease.pdf
6.8.1	SQA Checklist Sustaining Engineering Phase	\sustain\[6.8.1]sqasustain.dot \sustain\[6.8.1]sqasustain.pdf
6.9.1	SQA Checklist SQA Function	\sustain\[6.9.1]sqasqafunction.dot \sustain\[6.9.1]sqasqafunction.pdf

Managing your Organization's Training Programs using SourceForge

Providing comprehensive training for product development and other staff is an important function in any organization. The availability of quality training programs contributes to the success of your development projects and promotes long-term staff efficiency and retention. This section discusses a model for managing both project-specific and broader organizational training needs using SourceForge.

Project-Specific Training

Every development project will have training needs specific to that project. Some common examples are skills training in such areas as coding, testing methodologies, new hardware and software platforms and similar technical areas. Training internal staff on the use, sale, and support of the finished product is also a project-specific training need. This particular training need is covered in greater detail in the Product Training Plan.

Training needs for each project are identified in the Product Plan to ensure that training requirements are identified early, understood by all key project stakeholders, and input into the Project Plan for execution.

Organizational Training

Organizational training refers to training needs that are either independent of any specific development project, or shared across more than one project simultaneously. Some examples of independent training requirements are management training, language or interpersonal skills training, or training for staff in functional areas not directly associated with development projects such as Human Resources, Legal, or Finance.

Training needs that may be shared across projects include any skills or other project-specific training that is required for staff in more than one project simultaneously. Training staff in the use of the CDP to perform process-related activities is often an organizational training need. In such cases, multiple needs for the same training can be managed on an organizational level to achieve economies of scale and to ensure a common approach to the training.

Training Group

In order to manage organizational training needs and coordinate training across projects, a dedicated individual or group is required. Many organizations have a Training Group responsible for internal and/or external course development and other training needs. In other cases with organizations who are either just beginning a formal training program, or do not have the need or the resources for a dedicated Training Group, a group or individual may be identified to perform this function on an as-needed basis. Members of your Software Engineering Process Group are often good choices as they have a high level of visibility into the activities of the organization's development projects.

Process

The flowchart below describes a process for identifying project versus organizational training needs, assigning ownership, managing all action items to completion, and collecting evaluation data following each training activity.

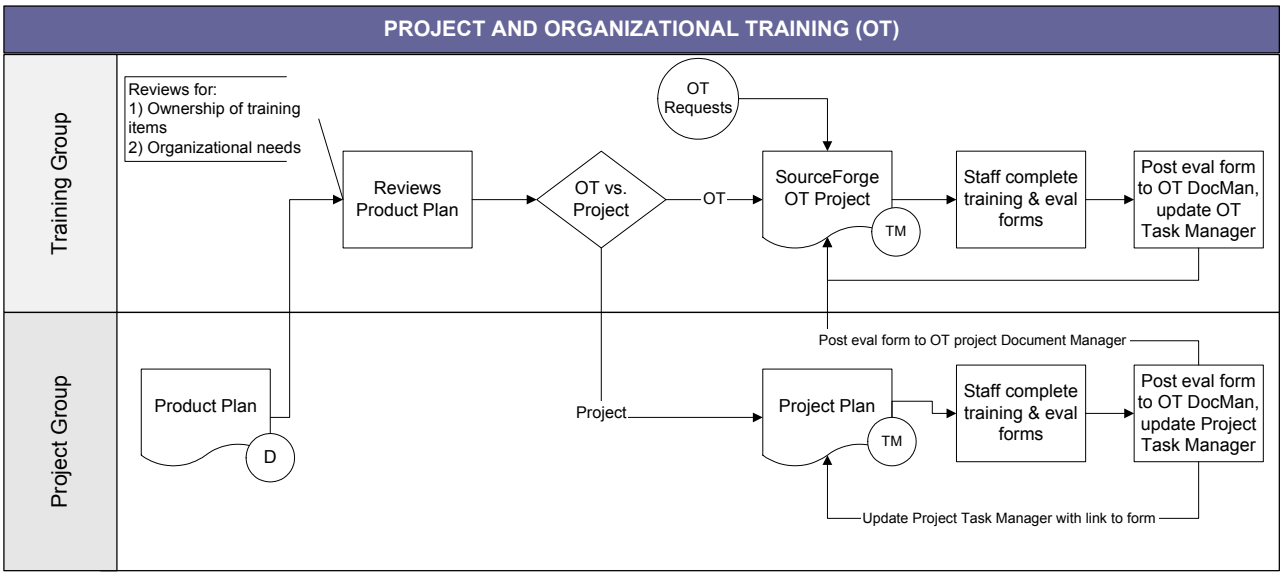


Figure 19. Project and Organizational Training

1. Identify Training Needs in the Product Plan

The first step is to identify all project-specific training requirements in the Product Plan for each project. Through peer review of the Product Plan, stakeholders from all impacted functional areas are given the opportunity to submit their training requirements.

2. Review by the Training Group

The Training Group is a required reviewer of the Product Plan. The Training Group is responsible for a) identifying training that they will be responsible for creating or delivering and b) identifying any training that can be combined with other training requirements and managed organizationally.

Before signing off their approval, the Training Group should identify each training item in the Product Plan as project or organizational training using the check boxes provided.

3. Training Worksheets

The owner of each training item identified in the Product Plan is responsible for completing a Training Worksheet. The Training Worksheet provides the owner of the Project Plan with the details of each training activity to be incorporated into the schedule and managed to completion using the Task Manager. The Training Worksheets should be stored in the Organizational Training Project on SourceForge as discussed below.

4. SourceForge Projects

All training identified as project training is incorporated into the Final Project Plan and managed to completion using the individual project's Task Manager.

All training identified as organizational training is managed in a separate SourceForge project dedicated to organizational training. This Organizational Training Project is managed by the Training Group and provides a central repository for all training materials, including Training Worksheets, Training Evaluation Forms, and the schedule for all training activities managed by the Training Group. If the Training Group is responsible for developing internal and/or external courses, development of these courses is managed in the Organizational Training Project as well.

It is important that all staff are given permission to post documents to the project's Document Manager. You may wish to use Role-based Access Control to limit general staff access to selected folders in the Document Manager.

5. Training Evaluation Forms

After each training activity is completed, all attendees are required to complete a Training Evaluation Form. This helps the Training Group and other managers measure the quality and results of each training activity and provides a record of all training completed by each staff member.

Attendees should post all Training Evaluation Forms for both project and organizational training to the Organizational Training Project in SourceForge. When updating the status of the training task in the Task Manager, the attendee should create an association to the Training Evaluation Form in the Document Manager.

6. Other Organizational Training

When training needs are identified that are clearly outside the scope of any development project, and therefore are not included in a Product Plan, the manager requesting the training should submit the request directly to the Training Group. Steps 3 through 5 are then followed to complete the training.

Development of Training Courses and other Materials

Training courses developed by the Training Group should include a description of the course that can be posted to the Document Manager in the Organizational Training Project. This allows staff to search for relevant courses when training needs arise. All training course materials prepared by the Training Group should be peer-reviewed using the Iterative Document Review and Approval Process.

Training Program Evaluation

The Software Quality Assurance group should conduct periodic audits of the training program to assess the successes and areas of potential improvement. Please see "Managing Software Quality Assurance Audits using SourceForge" on page 257 for additional information on conducting audits.

SW-CMM NOTE

The SW-CMM Level 3 Key Process Area Training Program requires an organizational training program as specified in this section.

See *SourceForge and the Capability Maturity Model for Software, A Supplement to the SourceForge Collaborative Development Process* for specific SW-CMM requirements satisfied by an effective training program.

Table 23. Training Templates

Ref#	Template	File Location
7.8.1	Training Worksheet	\crosslife\[7.8.1]trainingwork.dot \crosslife\[7.8.1]trainingwork.pdf
7.9.1	Training Evaluation Form	\crosslife\[7.9.1]trainingeval.dot \crosslife\[7.9.1]trainingeval.pdf

Managing Project Logistics using the SourceForge Tracker and the Monitoring Function

The Product Plan completed in the Product Definition phase identifies a number of factors with the potential to impact the success of the development project. These factors include estimates for size, effort, cost, critical computer resources, and risks. Other factors may also have been identified. Once identified, these factors must be actively monitored to ensure that any significant deviations are flagged early and the appropriate responses can be taken.

The SourceForge Tracker provides an excellent resource for tracking and managing these factors. After the Product Plan is completed and the size, effort, cost, critical computer resources, risks, and other critical factors are identified, trackers should be set up and populated with each item to be tracked. The monitoring function will then allow users to monitor each item and be notified via email when any changes have occured.

Each item being tracked should be entered as a unique tracker artifact and should be assigned to the individual responsible for its management. That person is then responsible for updating the artifact if and when the item's status changes to a point where it needs to be flagged as a potential issue. The artifact will then be routed to the appropriate person for review and action according to the workflow rules you establish. All users monitoring the item will also be notified, providing increased visibility into the updated status of the item.

It is recommended that the Launch Team be responsible for monitoring the project logistics tracker and managing any issues to resolution.

Size

Enter the estimated size as a tracker artifact and assign it to the person responsible for monitoring size. Use the Category field to identify the artifact as a size estimate.

If the estimated size of the product changes to the point that it represents a potential impact to the project, the owner should update the priority field and enter the applicable comments. If action is necessary, such as recalculating effort or cost, the owner should reassign the artifact to the appropriate person for action.

Effort

Enter the estimated effort as a tracker artifact and assign it to the person responsible for monitoring effort. You may choose to create dependencies between effort estimate artifacts and the artifacts for the size estimates on which they were based. Use the Category field to identify the artifact as an effort estimate.

If the estimated effort associated with the product changes to the point that it represents a potential impact to the project, the owner should update the priority field and enter the applicable comments. If action is necessary, such as a request for additional time or resources, the owner should reassign the artifact to the appropriate person for action. Use the Status field to represent status such as Action Pending, Approved, or Denied.

Cost

Enter the estimated cost as a tracker artifact and assign it to the person responsible for monitoring effort. You may choose to create dependencies between effort estimate artifacts and the artifacts for the size estimates on which they were based. Use the Category field to identify the artifact as a cost estimate.

If the estimated cost associated with the product changes to the point that it represents a potential impact to the project, the owner should update the priority field and enter the applicable comments. If action is necessary, such as a request for additional budget, the owner should reassign the artifact to the appropriate person for action. Use the Status field to represent status such as Action Pending, Approved, or Denied.

If size, effort, and cost estimates were calculated on a per-feature basis, you may choose to enter separate artifacts for each feature.

If you are subcontracting any part of the project, include the budget for the subcontracted work as separate cost artifacts.

Critical Computer Resources

Create a separate tracker artifact for each critical computer resource identified in the Product Plan. Use the Category field to identify the artifact as a critical computer resource.

If the resource becomes unavailable or otherwise at risk, the owner should update the priority field and enter the applicable comments. If action is necessary, such as a request for additional funds for computer resources, the owner should reassign the artifact to the appropriate person for action. Use the Status field to represent status such as Action Pending, Approved, or Denied.

Risks and Contingency Plans

You may wish to create a separate tracker to manage risks. Using a separate tracker will allow you to use the custom fields feature to set up required fields for Likelihood, Impact, and

Contingency Plan. Create a separate artifact for each risk identified in the Product Plan. For risks with associated contengency plans, include the contingency plan in the tracker artifact. If that risk reaches a high likelihood of occurance, the contingency plan can be triggered by setting the status of the artifact to Initiate Contingency Plan.

By tracking and managing critical estimates, resources, and risks using the tracker as described above, deviations will be identified as soon as they become imminent. The occurance or increased likelihood of a risk will be flagged and the appropriate contingency plan can be implemented. Visibility is also provided at any time into the status of these critical factors.

SW-CMM NOTE

The SW-CMM Level 2 Key Process Areas Software Project Planning and Software Project Tracking and Oversight, and the Level 3 Key Process Area Integrated Software Management require documented procedures for tracking and managing estimates, resources, and risks. Following the processes specified in this section satisfies these requirements.

See *SourceForge and the Capability Maturity Model for Software, A Supplement to the SourceForge Collaborative Development Process* for specific SW-CMM requirements satisfied by effective processes for tracking and managing estimates, resources, and risks.

Managing Subcontractors using SourceForge

With ever-changing market conditions affecting businesses worldwide, development organizations are looking more and more to both domestic subcontracting and offshore outsourcing as viable alternatives. A primary focus of subcontracting is cutting costs without sacrificing quality. Additional reasons for considering subcontracting portions of your development project include:

- Lack of appropriately skilled internal staff
- Ability to improve quality or service level
- Ability to accelerate time to market

This section will discuss considerations when making the decision to subcontract, criteria for evaluating and selecting appropriate subcontractors, and ongoing management of subcontracted work using SourceForge.

Make or Buy Decision

Subcontracting has the potential to help overcome staff shortages and reduce application management costs, but only if managed correctly. Not all development projects are suitable for outsourcing to a remote site. The following criteria should be considered when evaluating the make or buy decision.

Size of Project

Not all application development projects are suited to outsourcing, offshore or on. The most successful are generally large projects that don't require your extensive day-to-day management, and have been well defined and planned.

Critical Mass

A certain size team is generally necessary to make subcontracting worthwhile. By evaluating costs versus benefits upfront, you may find that upfront fixed costs may not justify establishing a small subcontracted team for a small or short-term project.

Initial Ramp-up Time

Ramping the subcontracting team up to speed involves not only the application requirements, but also and understanding of process and organizational culture. Getting a new team up to speed involves relationship building; often a team will not become truly productive for at least three to four months or longer for complex projects.

Knowledge Retention and Transfer

Subcontracted staff working on your projects will develop skills and experiences that will be valuable in the future. Determine up front where such skills and experiences should reside. Also consider whether these skills are tactical or strategic, and what best supports your business goals.

Confidentiality and Security of Intellectual Property

It is critical to establish an environment of trust with the selected vendor, while executing the necessary legal protections. Make sure there are protections in place in the event of employee turnover on the part of the subcontractor. Consider outsourcing only development work that will have little impact if it is appropriated. Groundbreaking research may be better maintained internally.

Subcontracting provides opportunities to lower development costs, improve software quality, and accelerate your time to market. The key is to explore all the risk factors as well as the advantages before engaging in subcontracted development.

Subcontractor Evaluation and Selection

Once you have decided to subcontract portions of your development project, it is important to conduct a careful evaluation and selection process. A reputable development organization will be willing to provide a prospective customer with samples of their work, documentation of their internal methodology, and existing customer references. In this way, you can best evaluate their programming skills, management skills, and language skills.

The following steps are recommended to help you select the most appropriate subcontractor.

1. Select an Appropriate Engagement Model

There are several types of subcontracting engagement models, each of which has advantages and disadvantages.

- Staff augmentation, or a body shop, refers to a small company with specialized developers. Your internal managers are responsible for managing this staff; therefore, costs are less than other options, but communication and visibility into the status of the project may be minimized.
- A managed team consists of a project manager employed by the subcontractor who works jointly with a project manager internal to your organization. This team of managers manages the relationship plus project-specific milestones and daily activities. You may wish to have the subcontractor's project manager work on-site at your location, particularly if you are working with an offshore partner. This on-site presence helps both parties maintain a more holistic view of the overall project, leverage all available assets, and implement or proliferate best practices.
- An offshore development center (ODC) can be a subsidiary of your US-based company or a totally independent entity. An ODC can help you leverage some of the many advantages of offshore development.

You may also wish to evaluate working with an offshore versus domestic subcontractor. Some considerations when selecting a country or territory for offshore development general economy, geopolitical stability, government support, educational system, language skills, security, infrastructure, availability of skilled resources, legal and cultural issues.

2. Define Vendor Requirements

Vendor requirements fall into two categories: qualifications from a business and relationship standpoint, and project-specific technical qualifications.

A Subcontractor Selection Worksheet is provided to help you evaluate potential subcontractors on both categories of requirements. It should be completed in the Product Definition phase as part of the Product Plan. Before beginning the evaluation process, you will need to define the project-specific technical qualifications necessary for the work to be subcontracted. The business and relationship requirements are generally consistent across projects.

3. Evaluate Individual Providers against Key Requirements

Fill out the Subcontractor Selection Worksheet for each project or subproject being subcontracted.

4. Create a Short-list of Candidates who Meet Key Requirements

Use the data from the completed worksheets to identify a short list of candidates to whom a Request for Proposal (RFP) will be issued.

5. Issue a Request for Proposal (RFP) to Short-listed Providers

The RFP should include your organizational policy for managing subcontractors. It is recommended that this policy state that any subcontractor will be treated as a partner, and that the same development processes used internally to your organization will be followed by the subcontractor. The subcontractor will be required to use SourceForge as would internal staff for communication, tracking of progress and issues, management of documents, source code, and other work products. Ownership of specific deliverables will be identified when release responsibilities are assigned, and periodic reviews of their work will be conducted. The subcontractor will also be subject to Software Quality Assurance audits of various processes and work products.

Acceptance criteria for all work to be performed and a method for resolving conflicts and issues that may have cost or contractual impact should be included. You may also wish to establish milestones that will trigger payment upon completion.

Site visits for live interviews are recommended for the final several candidates.

The Subcontractor Selection Worksheet includes a section to track RFPs and proposals received.

6. Make Selection and Execute Agreement

Once you have selected the subcontractors for the project, include this information in the Subcontractor Selection section of the Product Plan.

Subcontractor Management

SourceForge provides all the tools needed to effectively manage subcontractors. By treating subcontractors as extensions of your own development organization and providing access to SourceForge, you can greatly increase the visibility, ease of management, and smooth transition of work produced by subcontractors. If you are concerned about protecting your organization's intellectual property, you can use project access controls to limit the access your subcontractors have to sensitive company materials. For example, you may wish to make your subcontractors restricted users, who will only be able to view those projects of which they are members.

Provide training to new subcontractors on the use of the various SourceForge tools and your standard development processes. Based on their specific responsibilities, ensure that they receive adequate training on how to perform their daily activities using SourceForge. Some specific items may include:

- Understanding requirements using the Tracker. These can include features, bugs, and other artifacts assigned to the subcontractor for execution.
- Using the SourceForge SCM repository for code management.
- Integrating the schedule for their work into the overall Project Plan.
- Using the Task Manager to manage their schedules.
- Providing regular status updates using the Task Manager.
- Using the Document Manager for posting and referencing all project documentation.
- Retrieving process templates and related documentation from the Software Process Database.

Subcontractors should be included in regular staff meetings. Based on the significance of the work being subcontracted, you may wish to have a member of the subcontracting team participate in the Launch Team.

Subcontractors should also be made aware of the way in which their work will be tracked and managed. Some specific items may include:

- Reporting responsibilities. (An internal manager is identified in the Product Plan to manage each subcontractor's work.)
- Managing status changes, escalations, and other exceptions using the Task Manager.
- Peer reviews and Software Quality Assurance audits.
- Methods for resolving any conflicts and issues.
- Periodic meetings for contractual reviews.
- Any changes to the budget managed through the Project Logistics Tracker.

By managing subcontractors as regular staff members with regard to using SourceForge and your standard set of development processes, you can maximize the cost and productivity

benefits that subcontracting offers, while minimizing miscommunication, poor visibility, and other common risks.

SW-CMM NOTE

The SW-CMM Level 2 Key Process Area Software Subcontract Management requires documented procedures for selecting and managing subcontractors. Following the processes specified in this section satisfies these requirements.

See *SourceForge and the Capability Maturity Model for Software, A Supplement to the SourceForge Collaborative Development Process* for specific SW-CMM requirements satisfied by effective processes for selecting and managing subcontractors.

Table 24. Subcontractor Management Templates

Ref#	Template	File Location
1.8.1	Subcontractor Selection Worksheet	\proddef\[1.8.1]subcontractor.dot \proddef\[1.8.1]subcontractor.pdf

Managing the Document Lifecycle using the SourceForge Document Manager

The SourceForge Document Manager provides a variety of features designed to facilitate the management of your documents through all stages of the CDP.

This section provides the following information:

- Recommended processes for using the SourceForge Document Manager to effectively manage your documents through the creation, review, and approval process
- Managing multiple projects using the Document Manager
- Best practices for setting up your Document Manager folder hierarchy
- Iterative Document Review and Approval Process Flowchart

Managing the Document Review and Approval Process

A key element in the CDP is ensuring that stakeholders from all functional groups are involved in an iterative document review and approval process, prior to the commitment of resources and the initiation of the work contemplated in the document. It is also critical that any changes to key documents are recorded and the new information quickly communicated to all concerned parties.

The SourceForge Document Manager provides a variety of features designed to facilitate the management of your documents throughout their lifecycle and ensure the appropriate level of involvement of other project members. In addition to storing documents for reference purposes, the Document Manager provides a document review workflow process to allow you to actively engage other users in the review and approval of a document.

Simple Notification Process

If you would like to keep other SourceForge users informed when the status of a document is updated, e.g. when a new document is posted, or there is a change in the contents or status of an existing document, but do not require a formal review and approval process, the following steps provide a best practice for this level of document management:

- Post the document to the Document Manager in the desired location.
  - If you would like to prevent other users from editing, moving, or deleting the document, use the "Lock Document" feature. (If you choose not to lock the document, any user with Edit permission may edit and update the document.)
- Notify other users that the document is available and request that they begin monitoring it.

All users monitoring an item receive email notification whenever the item is updated.

Document Workflow Review and Approval Process

In cases where more formal document review is desired, the SourceForge Document Manager provides an easy-to-use workflow process for actively managing a document's review and approval. After submitting a document, you can start a document review. When submitting a document for review, you will be prompted to provide the following information:

- The names of all required and optional reviewers (all reviewers must have permission to access the document)
- The date by which the review must be completed
- Email message text

You will be notified each time a reviewer submits review comments.

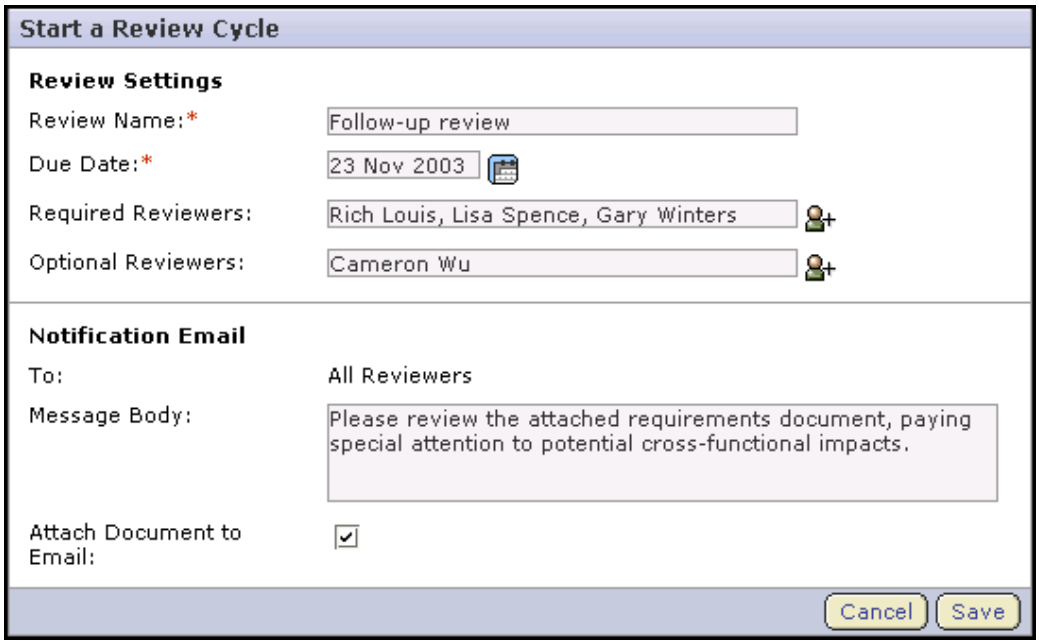


Figure 20. Start a Review Cycle page

When selected to review a document, users will receive an email notification with the relevant details and a link to the document and/or the document as an attachment. An item will also appear in the Documents Awaiting Review section of each reviewer's My Page, including the due date.

An Submit a Response section is provided within the Document Manager where reviewers can enter their review comments. Reviewers should be instructed to include "I approve this document" or similar text in the Submit a Response section to indicate that they have approved the document.

All reviewers and the document submitter will be able to read the reviews of all other reviewers once they have been submitted.

Iterative Document Review and Approval Process

The following flowchart describes a specific process for iteratively reviewing and gaining approval on documents described in the CDP, using the SourceForge Document Manager features described in the previous section.

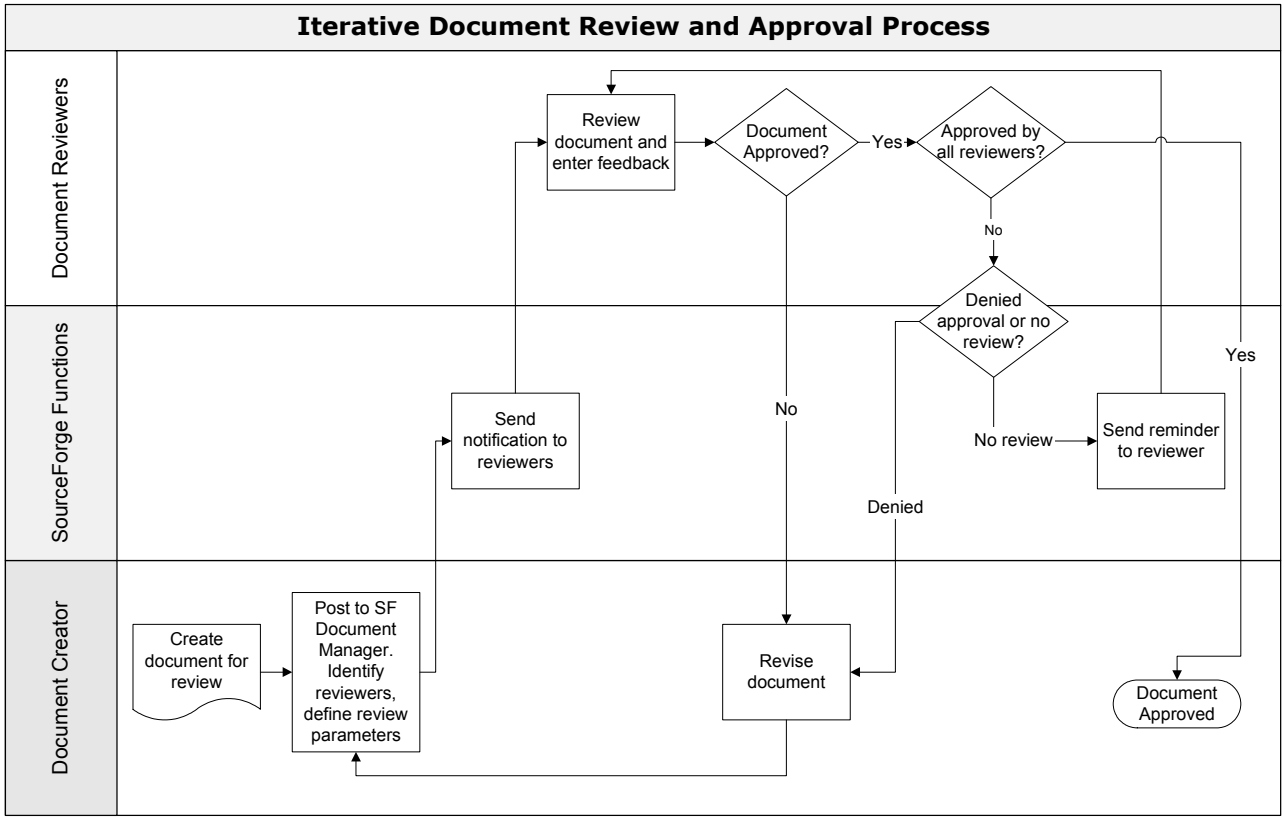


Figure 21. Iterative Document Review and Approval Process

Following the two SourceForge document management best practices, Simple Notification and Document Workflow Review and Approval, along with the Iterative Document Review and Approval Process, will help you ensure that stakeholders from all impacted functional groups are involved at the appropriate stages of the CDP, and are continually kept aware of critical updates. This, in turn, will contribute to smoother execution of all stages of the CDP, and more predictable project results.

Managing Multiple Projects

Managing a single development project using SourceForge is straightforward; a single SourceForge project provides all the tools you need for your project activities in a centralized location.

However, a much more common scenario is one in which multiple projects are managed simultaneously. This situation requires you to select from several options for managing your projects and your Document Manager hierarchies. You may choose to maintain a separate SourceForge project for each of your projects, or it may be more effective to manage your projects if they are grouped in one of several ways. Each of these options has its own advantages.

Here are some questions you may wish to consider when managing multiple projects:

Are the projects related or completely independent of one another?

- If your projects are not related to one another, and will not produce common documents or other artifacts such as feature requests, tasks or source code, it is a good idea to create a separate SourceForge project for each of your projects.
- This approach enables you to easily manage each project independently, and avoid creating confusion with other, unrelated material.
- You will still be able to move selected documents, tasks, and tracker artifacts among projects if you find that you have data that needs to be reorganized.

Are the projects highly dependent on one another, with the need to share common documents or other artifacts?

Some examples are:

- A major software release, with a series of subsequent minor and patch releases, e.g. Software Product v1, v1.1, v1.2, v1.2.1, etc.
- Projects that have close interdependencies, e.g. shared documents, source code, or tasks.
- If your projects are very closely related, you may wish to create a single SourceForge project, and use your Document Manager folder hierarchy to manage the documents associated with each sub-project separately.
- Managing closely related projects using a single SourceForge project allows you to simplify access controls and maintain all common data in a single location.

It is important to note that your documents should always be maintained according to the SourceForge project structure you have established. For example, if you have chosen to manage three closely related projects using a single SourceForge project, the documents for all three of your sub-projects should be managed using the single SourceForge Document Manager. Managing documents in this way will facilitate ease of use and enable users to quickly and easily locate project documents.

Best Practices for Setting Up Your Document Manager Folder Hierarchy

A well-structured Document Manager folder hierarchy is an important element in an effective document management process. Following are some examples intended to provide a basis for structuring your Document Manager folder hierarchies.

Note: If you need help learning how to set up and administer the Document Manager, please reference the *SourceForge Enterprise Edition 4.2 User Guide*.

When you are managing related projects using a single SourceForge project, it is a good idea to keep your root directory simple; a folder for each major category of documentation applicable across all sub-projects, plus a folder for each sub-project.

Document categories applicable across all sub-projects may include:

- Process Documentation and Templates (retrieved from the Software Process Database)
- Engineering and Product Management documentation
- Software Quality Assurance documentation
- High-level Project Plan
- Any high-level documentation that covers all sub-projects, e.g. Business Strategy or Product Roadmap.

Figure 22. Root Directory with Sub-projects

At the sub-project level (or the Root level if you are managing a single project), additional folders should be created to enable finer categorization and easier accessibility of documents.

Document sub-categories applicable at this level may include:

- Product Requirements Documents
- Technical Design Documents
- User Documentation
- Test Plans and Results
- User Interface Documents
- Detailed Project Plans
- Third-Party Materials (useful if working with third party technologies or to consolidate material produced by subcontractors)
- Meeting Minutes

Some of these categories may or may not be applicable to a particular project. You may also wish to add, delete, change, or move a particular category to the Root level or a second layer of sub-category. .

In some cases, you may wish to add a second level of sub-directory folders in order to further refine the categorization of your documents. This is an acceptable practice and often provides additional clarity; however, adding more than two levels sub-directory folders may add such complexity that documents become difficult to locate.

Unless your document categories are extremely well-defined, follow an intuitive, logical sequence, a third level of sub-directory folders is generally not recommended. Documents that do not fit into an established category may always be placed in the Root or sub-Root directory.

SW-CMM NOTE

The SW-CMM Level 3 Key Process Area Peer Reviews requires peer review of various project documents as specified in this section.

See *SourceForge and the Capability Maturity Model for Software, A Supplement to the SourceForge Collaborative Development Process* for specific SW-CMM Level 2 and 3 requirements satisfied by an effective document review and approval process.

Managing Product Requirements Using the SourceForge Tracker

Capturing product requirements clearly, thoroughly, and accurately is a fundamental goal of any software development organization. Establishing clear goals for your product and ensuring that these goals are well defined, communicated, and understood by the organization are critical to the success of any project. Poorly defined or poorly managed requirements can result in wasted time, effort, and money, while clearly defined requirements contribute significantly to the production of high quality products, on time and on budget.

Requirements management should an ongoing process, with activities such as analysis of customer requirements, market research, competitive analysis, and research into emerging technologies being conducted continuously. Many organizations have staff in Business Strategy, Market Analysis, Product Management, or similar roles dedicated to these functions. By continually researching and analyzing ways to improve the product offerings of the organization, staff in such roles are able to define and maintain a high-level product strategy for the organization. This high-level product strategy is refered to throughout the CDP as the Product Roadmap.

From the Product Roadmap, product-specific requirements documents are created during the Product Definition phase of each project. These requirements documents are refined to a greater and greater level of detail until they are ultimately refined into detailed technical designs used by product development staff to code and test the product. Throughout this process of specifying requirements and developing products, the Product Roadmap may continue to be updated with changing requirements.

The SourceForge Tracker provides many features to help you effectively manage key aspects of the requirements management process from Product Roadmap to creation and testing of code.

This section provides the following information:

- Managing your Product Roadmap
- Recommended practices for gathering requirements input from a cross-functional audience
- Managing multiple projects using the SourceForge Tracker
- Managing feature requests, product defects (bugs) and other requirements input using the SourceForge Tracker
- Ensuring traceability of requirements
- Requirements Management Process Flowchart

Managing Your Product Roadmap

The Product Roadmap is the vehicle for defining the long-term product strategy for your organization. It may be limited to short-term goals, or span a series of product releases to be produced over a period of many years. In either case, it should be continually or periodically updated and treated as independent from individual project-specific requirements. The Product Roadmap provides a readily available source of requirements information that can be used to define requirements for a specific product release during its Product Definition phase.

The following diagram illustrates the continuous requirements management process running in parallel to the development of individual products. The diagram also indicates progress of requirements from the Product Roadmap through the detailed Technical Design Documents, and the manner in which critical requirements can be incorporated into a product using a change request process. It should also be noted that a one-to-many relationship often exists between the Product Roadmap and individual products under development.

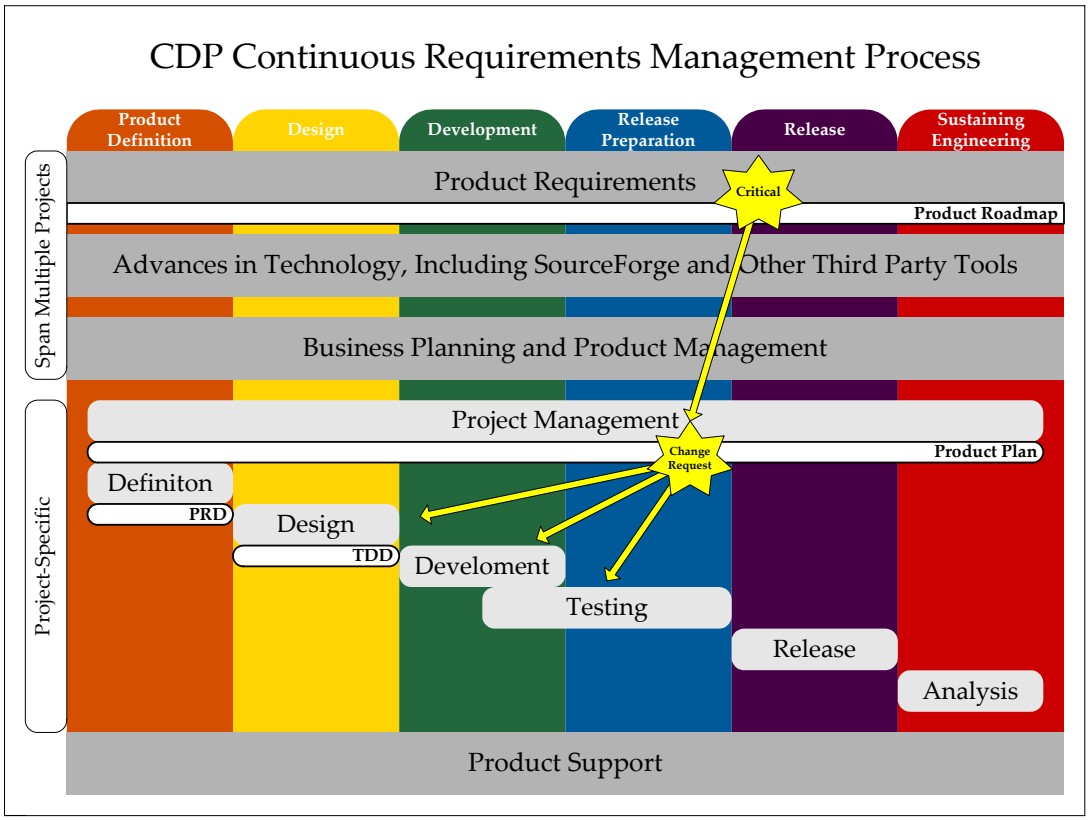


Figure 23. CDP Continuous Requirements Management Process

Gathering Requirements Input

Requirements for a product may come from a variety of sources. Development of a new product may be initiated based on requests for new or improved functionality, the discovery of a new market opportunity, the advancement of new technology, or a variety of other reasons. The Product Roadmap should be continuously updated to reflect these changing factors.

The Product Roadmap should form the starting point for identifying the nature and scope of product-specific requirements gathering activities. The degree to which product requirements have already been defined in the Product Roadmap, plus the overall scope of the project (timeline, resource availability, strategic goals and other factors) are key considerations.

After taking these considerations into account, you may determine that the requirements gathering process for your particular project need only consist of defining a single, already identified, key requirement or perhaps referencing an existing process for prioritizing product defects. In such cases, the requirements definition step of the CDP may be simplified.

However, another common scenario is one in which requirements are not yet fully defined, and a process needs to be undertaken to ensure that requirements are accurately captured from key project stakeholders. For the purposes of this guided best practice, the following assumptions are made:

- Key requirements have been identified at a high level in the Product Roadmap.
- The scope of the project allows for multiple requirements, not all of which have yet been identified.
- The product has stakeholders throughout the organization.
- The product will contain new features, enhancements to existing features, and resolution of product defects (bugs).

Soliciting product requirements input from a variety of sources (both internal and external to your organization) is generally desirable in such cases. Identifying stakeholders from whom to solicit input is based on a variety of considerations, but in general, a best practice is to engage as broad a range of stakeholders as possible. This will help ensure diversity in your feature requests, representing a wide range of viewpoints and product knowledge.

Try to include representatives from internal cross-functional teams in your requirements gathering process, including members of groups that will be responsible for selling or supporting the product after it is developed. Some examples are Technical Support, IT, Sales, Marketing, and Professional or Consulting Services. It is also critical to gather as much feedback as possible from current and potential customers, whether they are internal or external to your organization.

Once you have identified your stakeholders, some methods you may wish to use to gather product requirements input are:

- Brainstorming sessions
- Cross-functional meetings
- Review of common technical support requests
- Surveys
- Focus groups
- Customer Advisory Groups or other customer-focused events

Other sources of input may include market research, competitive analysis, and research into emerging technologies.

The activities referenced above are generally conducted to define requirements for a specific product release. In addition to performing these activities on an as-needed, project-specific basis, it is also highly recommended that processes be initiated to gather requirements input on an ongoing basis. Some examples of ongoing processes are:

- Using the SourceForge Feature Requests tracker to gather product requirements input at any time during the CDP.
- Using the SourceForge Bugs tracker to gather product defect input at any time during the CDP.
- Setting up a customer-facing tracker to gather input from external customers at any time during the CDP.

Establishing ongoing processes for gathering product requirements input will ensure that there is a pool of input available that can be leveraged whenever a new product is being planned, and supplement your project-specific requirements gathering activities. It may also help in determining when a new product should be developed. For example, a large number of product defects or outstanding customer issues may signal the need to begin work on a maintenance release. A large number of feature requests for a specific product feature may serve as the trigger for initiating a more significant release with new or enhanced functionality.

As you are gathering feature request input, it is a best practice to enter all input into the SourceForge Tracker, regardless of its source, to ensure that all feature-related ideas are captured and available for future review and prioritization. Following meetings and other requirements gathering activities, you may wish to submit tracker artifacts for each suggestion that may be considered for inclusion in a future release.

Once you have outlined your plan for gathering product requirements input, the next step (if it has not already been addressed during a previous project) is to establish and circulate guidelines that will enable you to leverage the SourceForge Tracker to efficiently manage the product requirements artifacts that will be generated.

The following sections will discuss:

1. Establishing guidelines for submitting input
2. Best practices for setting up trackers
3. Reviewing, clarifying, and prioritizing input
4. Exporting selected set of requirements into the Product Requirements Documents
5. Ensuring future traceability of product requirements

Establishing Guidelines for Submitting Input

Establishing guidelines for submitting feature request and other input, such as product defects or bugs, is important to ensure that the input is submitted in a consistent, usable manner. A policy document should be drafted and circulated including the following information:

- The SourceForge project/s that will be used to manage feature request input
- Which tracker/s should be used for input (You may wish to use one or more than one tracker)
- Which fields (category, group, custom fields, other predefined fields) should be used for various kinds of input, and which will required
- Guidelines regarding setting priority levels and communicating the business or commercial justification for feature requests
- Guidelines regarding categorization of input to prevent duplication and facilitate filtering
- Any specific detail that should be included in the tracker artifact, e.g. source of the request, specific descriptive detail, file attachments, etc.
- The process that will be used to review, clarify, and prioritize input
- If applicable, any timeline associated with submitting input for an upcoming product release

The policy document should apply to input submitted for a particular product, i.e. as part of a formal requirements gathering process, as well as to general input submitted at any time during the CDP, such as new feature requests, feature enhancement requests, and product defects or bugs. You may wish to create separate policies for feature requests and for product defects, as they are often managed by different people or functional groups, and may have different requirements, e.g. required fields, specific descriptive text, or priority levels.

Since soliciting requirements input is best managed as a continuous process, it is recommended that the policy document or documents be circulated shortly after installing SourceForge, or prior to beginning to use the Tracker for requirements management. The following section will provide a series of best practices for setting up and using SourceForge Trackers to manage product requirements. These best practices may be useful in establishing your tracker policy document.

Best Practices for Setting up Trackers

The following section provides a series of best practices when using the SourceForge Tracker to manage requirements input. The material in this section may be useful in establishing your tracker policy document.

Managing Multiple Projects

Managing a single development project using SourceForge is straightforward; a single SourceForge project will provide all the tools you need in a centralized location.

More commonly, however, multiple projects are managed simultaneously. This situation requires you to select from among several options for managing your projects and your Trackers. You may choose to maintain a separate SourceForge project for each of your projects, or it may be more effective to manage your projects if they are grouped in one of several ways. Each of these options has its advantages.

Here are some questions you may wish to consider when managing multiple projects:

Are the projects related or completely independent of one another?

- If the projects are not related to one another, and will not produce common tracker artifacts, or other artifacts such as documents, tasks or source code, it is a good idea to create a separate SourceForge project for each of your projects. This will enable you to easily manage each project independently, and avoid creating confusion with other, unrelated material.
- You will still be able to move selected tracker artifacts among projects if you find that you have data that needs to be reorganized.

Are the projects highly dependent on one another with the need to share Tracker or other artifacts?

Some examples are:

- A major software release, with a series of subsequent minor and patch releases, e.g. Software Product v1, v1.1, v1.2, v1.2.1, etc.
- A series of bug fix releases, where artifacts not addressed in one release will need to be rolled over to the next on a continual basis.
- Projects that have close interdependencies, e.g. shared tracker artifacts, source code, or tasks.
- If your projects are very closely related, you may wish to create a single SourceForge project and configure your trackers to manage the tracker artifacts associated with each sub-project separately.

Managing closely related projects using a single SourceForge project allows you to simplify access controls and maintain all common data in a single location.

Managing Multiple Trackers within a SourceForge Project

Each SourceForge project allows you to define an unlimited number of user-created trackers. When managing multiple product releases, it is generally a best practice to maintain a separate tracker or trackers for the artifacts associated with each release. This will enable you to more easily manage each release separately, and to close each tracker upon completion of the project.

If the product release will contain both new or enhanced features and resolved product defects, it is recommended that separate trackers be used for Feature Requests and Bugs to avoid confusion between the artifact types.

One solution for managing trackers and artifact input is to create one or more trackers for each product release, then enter the artifacts directly into the appropriate tracker. This method may be preferred by managers and other staff directly responsible for ongoing monitoring of requirements, who are familiar with all guidelines regarding artifact entry. The disadvantage to this method, however, is that artifacts may be unintentionally entered into the wrong tracker, and then must be moved. This may happen often in cases of feature request or product defect-related input where it is unclear to the user submitting the artifact how and when it is going to be addressed.

A more effective process that minimizes the margin for error and provides for review, clarification, and prioritization activities prior to the disposition of the artifact, is to have all feature request input entered into a single tracker first (and product defect input into another, if applicable), then sorted by management or other responsible staff into the appropriate project-specific trackers. This single tracker can be viewed as a 'holding' tracker to store artifacts until it can be determined when and how (and whether) to resolve them.

Managing product requirements input in this way will enable you to:

- Provide simple instructions to all users regarding where and how to submit feature request, product defect, and other product requirements input
- Eliminate confusion and unnecessary searching and moving of artifacts
- Eliminate the risk that artifacts will be assigned incorrectly and possibly missed
- Enable management staff to review all artifacts prior to assigning them to a particular project-specific tracker
- Enable management staff to maintain a single repository from which to easily sort, prioritize, and assign artifacts

Tracker Fields and Features

Instructing users to submit all feature request input into a single Feature Request Tracker will provide a large repository of artifacts to review as part of any requirements gathering process. In order to facilitate the extensive sorting and review that is generally associated with this process, the SourceForge Tracker provides a number of features of which you may wish to take advantage. Some of these include:

- Pre-defined, customizable fields, such as Category, Group, and Priority
- Optional user-defined Custom Fields to further refine categorization and search efforts
- Customer field to provide traceability of a request to a particular source
- File attachments to provide additional justification or clarification for a request
- Ability to auto-assign Categories to specific individuals

There are many options for using Category, Group and Custom Fields to facilitate sorting feature request input. They are highly customizable, based on the varying needs of individual projects and organizations.

Category - You should create Category values for each key functional area of your product, then auto-assign these requests to the Software Product Manager responsible for defining requirements for that functional area. If more than one person will be entering and/or reviewing artifacts, the Category field should be made required for users when entering artifacts. The Category field is also helpful in preventing duplicate entries. Users should be advised to search the tracker using the Category field for similar entries prior to entering a new artifact.

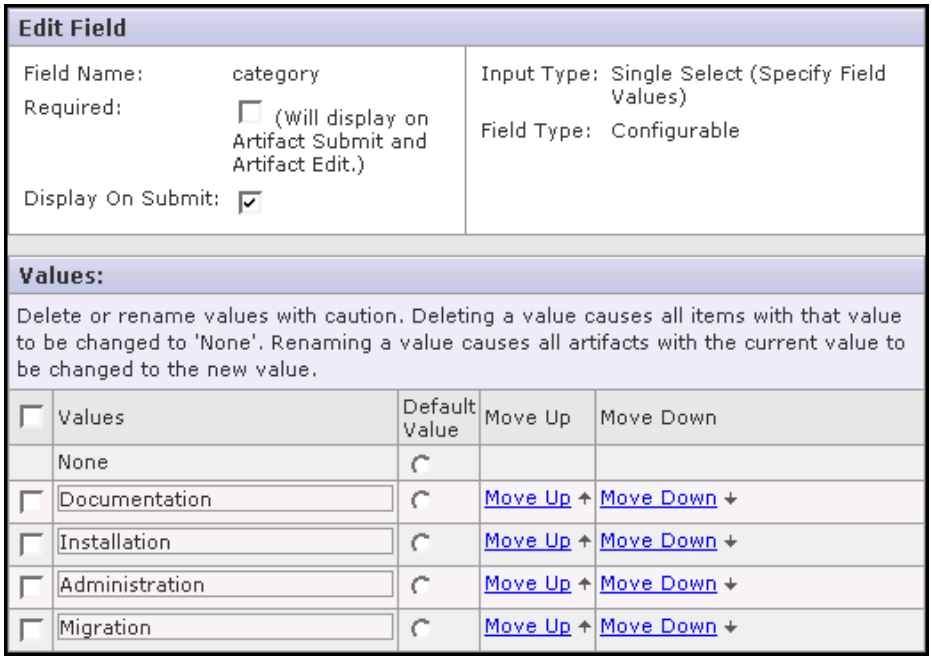


Figure 24. Edit Field page

Group - The Group field is often used to designate a specific product release. If used in this way, you may or may not wish to make this a required field for users when entering artifacts. It may be premature when the requirements definition process is still underway. It may also be more desirable for the Software Product Manager or other group or individual responsible for requirements to assign a Group value only after reviewing the artifact.

Customer - The Customer field is generally used to provide traceability of a requirement back to the customer or organization from whom it originated. This field is often used when Sales or other internal staff are entering feature requests on behalf of customers external to the organization.

Another potential use of the Customer field is to expand its values to include **all** sources of a requirement. For example, you may be conducting brainstorming sessions, cross-functional meetings, surveys, market research, or other requirements gathering activities associated with an upcoming product. Creating values to represent each source of feature request input will help ensure end-to-end traceability of all artifacts.

It is highly recommended that the Customer field be made a required field.

Completing the Tracker Policy Document

Each tracker policy document (feature requests, product defects, or other artifact types) should include all relevant detail about how you have chosen to set up and configure your SourceForge projects and trackers. The SourceForge Tracker will display only those fields you have selected (with the exception of several system-required fields), and will automatically enforce any required fields, however this information should also be included in the policy document to ensure that users are prepared with the required information prior to entering artifacts.

The policy document should also define any requirements or practices that are not enforceable by the tracker, e.g. specific levels of descriptive detail, and should inform users of the process that will be followed to review, prioritize, and resolve product requirements-related artifacts.

Reviewing, Clarifying, and Prioritizing Input

Once the input that will be used as the basis for determining the product's feature set has been collected, the next step is for the person or group responsible for managing product requirements to review each feature request, product defect, or other product requirements-related artifact and determine where it should be routed.

Use any filter or Tracker Search parameters to help sort and prioritize the artifacts. Some possible techniques include:

- Filter by Submit Date to sort out all artifacts entered since the last review period
- Filter by Priority level to review the highest priority artifacts
- Filter by Submitter to review artifacts submitted by key stakeholders
- Filter by Customer field to review customer-driven artifacts
- Filter by Category, Group, or Summary to review content-specific artifacts

If you have chosen to allow submissions to multiple trackers, instead of consolidating all input into a single Feature Requests, Bugs, or other tracker, simply repeat the same process in all applicable trackers.

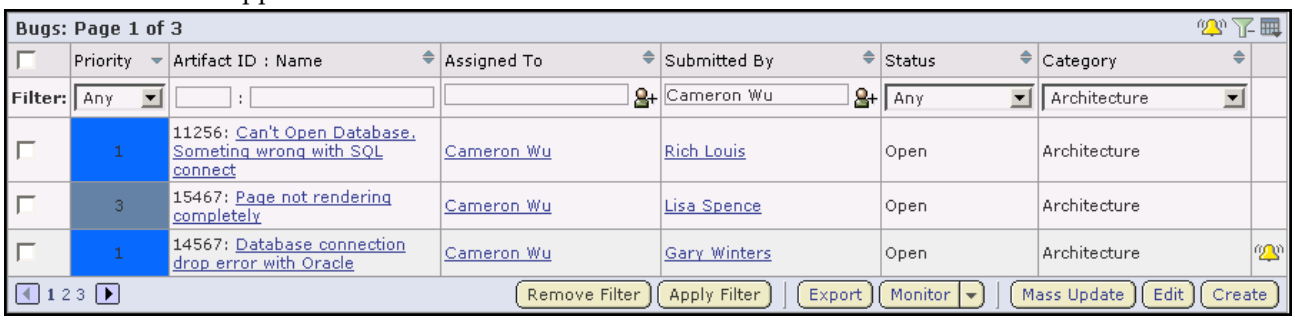


Figure 25. Filter Tracker Artifacts for Review

If additional clarification or detail is needed, route the artifact back to the submitter with a comment requesting the additional information. Once you have sufficient information to determine how to address the request, move the artifact from the holding tracker to the appropriate project-specific tracker. Once you have moved each artifact, you will need to reset the Category, Group, and other fields with tracker-specific values.

Some possible actions include:

- If the artifact will become a requirement for next product release:
  - Move the artifact into the project-specific tracker for the next product release.
- If the artifact will not become a requirement for the next product release, but will be deferred to an identified future product release:
  - Move the artifact into the project-specific tracker for a future release.
- If the disposition of the artifact has not yet been determined:
  - Leave the artifact in the holding tracker, or move to another holding tracker to indicate that it has been reviewed but its disposition has not yet been confirmed, or that the artifact has been deferred.
- If an artifact is being considered for inclusion in a product release, but additional discussion or approvals need to take place first (for example, a product feature is being proposed in the draft Product Plan):
  - You can create new Status values to represent the maturity level of the artifact, e.g. “Under Consideration”, “Included in Business Requirements Document,” “Proposed in Product Requirements Document,” etc.
- If the artifact is determined to represent a product defect rather than a feature request (or vice versa):
  - Move the artifact into the appropriate tracker for review.
- If the artifact will not be addressed:
  - Change the artifact’s status to “Deleted.” You may wish to create a new Status value such as “Denied” to avoid confusion with artifacts that have been completed and closed.

Ensuring Traceability of Requirements

SourceForge provides a series of features designed to help you ensure traceability of all requirements from inception through completion and post-release activities.

The first step toward establishing traceability is ensuring that the source of each requirement is captured during the feature request or product defect input stage. Identifying a field [Customer, User-defined Custom Field, other field] in which to capture the source of a requirement, and making it a required field, will achieve this first step.

Creating Associations

Once you have determined which feature request, product defect, and other artifacts will be addressed in the upcoming product release, the Create Association to Document feature will enable you to associate each artifact with the appropriate Product Requirements Document. This bi-directional association will enable a user to quickly identify which tracker artifacts are associated with a Product Requirements Document (or other document,) and provide an interface within the Document Manager to view a list of each associated artifact's summary information.

Creating associations between the Product Requirements Documents and their associated tracker artifacts is another way to ensure end-to-end traceability of requirements.

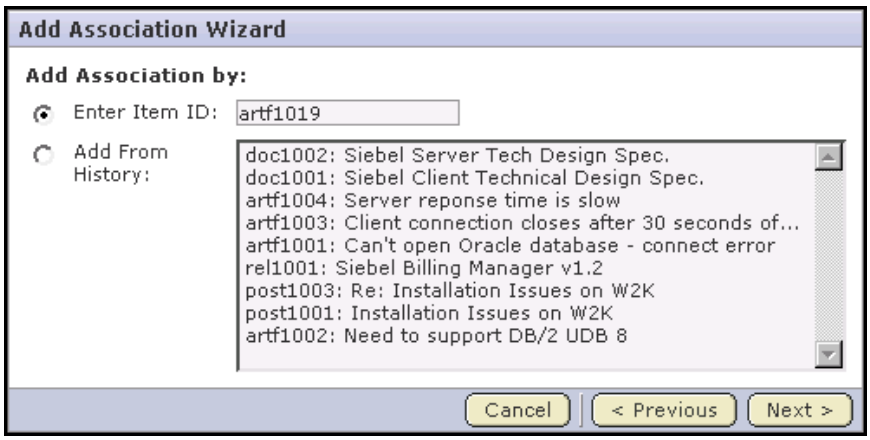


Figure 25. Add Association Wizard

Exporting Requirements

Once all feature request, product defect, and other artifacts have been reviewed and the items to be addressed in the upcoming release have been identified, you will have a project-specific Tracker containing only the artifacts that constitute the set of requirements for the release. If you have entered all product requirements input from all sources into the Tracker, this will be an all-inclusive list. The Export feature will now enable you to export selected artifact data into a .CSV, .XML, or tab-delimited file that can then be imported into the Product Requirements Documents.

If the artifacts have already been associated with the document using the Create Association feature, summary information will already be displayed in the Document Manager along with the document. If additional fields are desired, a .CSV, .XML, or tab-delimited file may be generated by using the interface provided.

If the artifacts have not been associated with the document, the manual Export feature can be used. Based on how your Product Requirements Documents are organized (one per major feature, one master Product Requirements Document, separate sections for customer-driven requirements, etc.), use the Advanced Filter to sort out only the applicable artifacts. If you have aligned the Tracker Category values with the product features covered by each Product Requirements Document, this filtering process will be greatly simplified.

Follow the prompts to define the export parameters, then import the resulting .CSV, .XML, or tab-delimited file into the desired application for inclusion in the Product Requirements Documents.

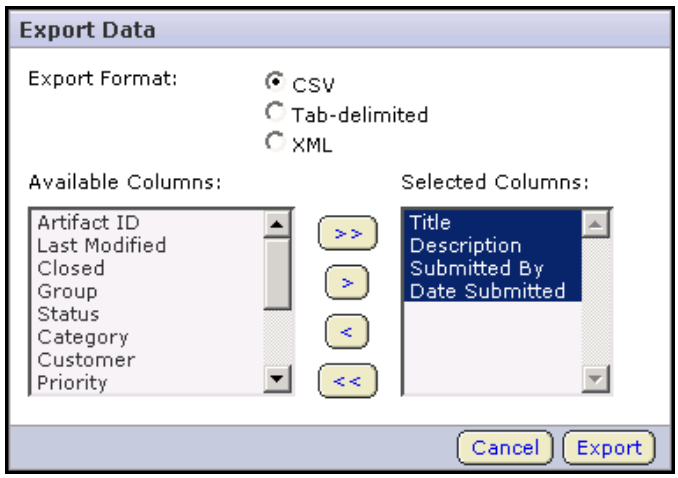


Figure 26. Export Tracker Artifact Data window

If you need to modify the list of requirements based on the results of the PRD Iterative Document Review and Approval Process, simply make the desired modifications to the affected artifacts, then repeat the export process using the new list of artifacts.

Note: If the artifacts have been associated with the Product Requirements Document using the Create Association feature, the Document Manager will now display an updated summary of associated artifacts.

Associations with Other Artifact Types

In addition to documents, the SourceForge Tracker provides the ability to associate artifacts with tasks, with file releases, with source code, and any other SourceForge artifacts.

The ability to associate your product requirements artifacts with other artifacts at any time during the Product Development Lifecycle will enable the following:

- Allow you to route a high level requirement in a project-specific tracker without assigning it to a specific owner. The person or group responsible for execution of artifacts in that tracker can then assign it in later phases.
- Allow you to include a high level requirement in the Product Definition phase that will then be split into associated smaller, task-level tracker artifacts in later phases.
- Allow you to include a high level requirement in the Product Definition phase that will then be split into associated smaller Task Manager artifacts in later phases.
- Allow you to associate source code with requirements artifacts in later phases.
- Allow you to associate requirements artifacts with specific file releases in later phases.

SW-CMM NOTE

The SW-CMM Level 2 Key Process Areas Requirements Management, Software Project Planning, and Software Project Tracking and Oversight require documented procedures for defining, managing, and tracking the status of requirements and using them as the basis for future work products. The SW-CMM Level 3 Key Process Area Software Product Engineering requires traceability of requirements. Following the guidelines in this section and using the Tracker as recommended satisfies these requirements.

See *SourceForge and the Capability Maturity Model for Software, A Supplement to the SourceForge Collaborative Development Process* for specific SW-CMM Level 3 requirements satisfied by an effective document review and approval process.

Requirements Management Process

The following flowchart describes a process for managing the requirements process using the SourceForge Tracker features described in the previous sections..

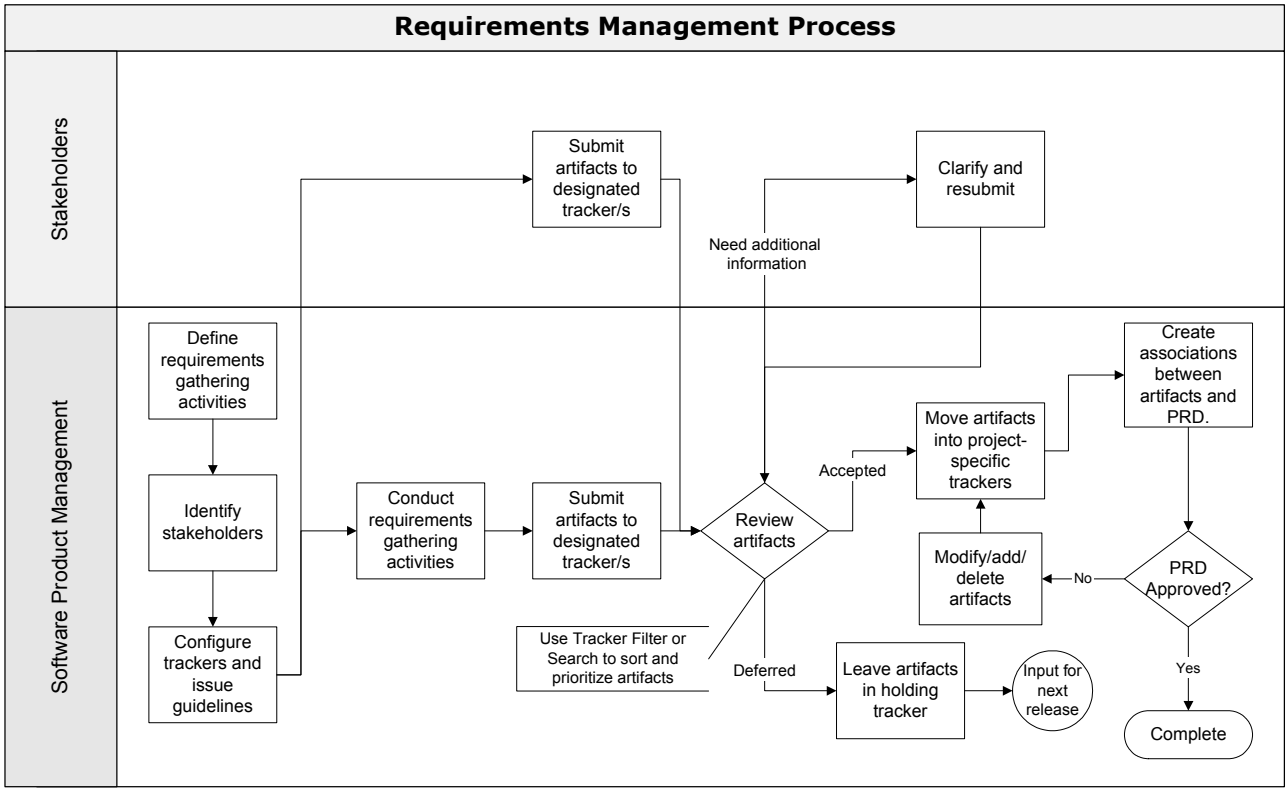


Figure 26. Requirements Management Process

Managing your Project Plan Using the SourceForge Task Manager

The SourceForge Task Manager is a powerful project and task management system with comprehensive reporting functionality that can help your organization implement and sustain the principles outlined in the CDP. The Task Manager provides increased efficiency through collaborative project planning and cross-functional status tracking throughout the development cycle. This allows organizations to maintain quality standards, reduce costs, and mitigate potential schedule delays and risks.

Much of the administrative effort required to manage your project plans throughout the various phases of the CDP can be systematically automated using the Task Manager. Global project administration settings, overdue task alerts, change approval requirements, and other sophisticated project management features provide effective project tracking, disciplined process control, and consistent adherence to defined standards.

This section provides the following information:

- Best Practices for Creating and Using your Project Plan
- Best Practices for Working with Microsoft Project, the Task Manager, and the Task Manager
- Best Practices for Managing Multiple Projects in the Task Manager
- Best Practices for Tracking Effort with the Task Manager
- Best Practices for Measuring Progress and Reporting to Management with the Task Manager

Best Practices for Creating and Using your Project Plan

The Project Plan consists of the schedule for all activities and deliverables associated with your development project. A well-constructed Project Plan provides task-level detail for all activities conducted during each phase of the CDP and by all contributing individuals and functional groups. It highlights milestones, critical path items, and dependencies among tasks. It allows you to accurately allocate resources across the project for all project activities and all functional groups. Owners for each item are explicitly assigned, and processes are in place for ongoing use, maintenance, changes, and visibility into the status of the plan.

Managers responsible for creating the Project Plan should conduct sufficient research and planning activities to ensure that the Project Plan is complete and accurate. A well-constructed Project Plan that is actively used according to defined processes can be one of the most critical factors contributing to the success of your project.

This section provides sample procedures for developing and maintaining your Project Plan using the SourceForge Task Manager.

Preliminary Project Plan

The Preliminary Project Plan establishes a milestone-level schedule to use as a basis for developing detailed, task-level schedules in the Final Project Plan. The Preliminary Project Plan in the Task Manager is intended to act primarily as a reference. Only those tasks due prior to the completion of the Final Project Plan will require updates.

PROCEDURE FOR DEVELOPING THE PRELIMINARY PROJECT PLAN

Overview

The Project Plan is developed in two phases. The Preliminary Project Plan is developed in the Product Definition Phase based on established milestones and the effort estimates developed in the Product Plan. It will be refined and expanded in the Design Phase down to task-level detail.

Preliminary Project Plan

The owner of the Preliminary Project Plan is the Product Manager responsible for the project. Input is gathered from key project stakeholders and the milestone dates and effort estimates established in the Product Plan. The goal is to establish a milestone-level schedule.

1. Create the Preliminary Project Plan as a Microsoft Project file for ease of review.
2. Include any fixed dates established in the Product Plan.
3. Include dates for all items due prior to completion of the Final Project Plan.
4. Calculate an estimated start and end date for each CDP phase. These estimates are based on the following information from the Product Plan:
    - Process tailoring section (# of documents)
    - Size and effort estimates (time for coding and testing)
    - Field testing section (time for beta and other external testing)
    - Peer review section (time for document and code reviews)
    - Software quality assurance section (time for audits)
    - Training section (time for required staff training)
    - Hardware and software, subontractor selection, and third party products sections (time needed to acquire necessary resoures or materials)
    - Delivery section (time needed to prepare for product release)
5. If available, review schedule data from previous similar projects. If needed, adjust estimated time for activities up or down based on past performance.
6. Review the risk analysis section of the Product Plan. Factor in additional time where needed to mitigate the impact of likely risks.
7. Distribute the draft Preliminary Project Plan for peer review using the Iterative Document Review and Approval Process.
    - If the time for coding & testing (or other activities) exceeds fixed completion dates, you may need to conduct Cross-Functional Peer Review Meetings to reduce the scope of the project or eliminate non-critical activities.
    - Ensure that the Product Plan is revised accordingly.
8. Once approved by required reviewers, export the Microsoft Project file into the Task Manager.

Final Project Plan

Before beginning the Final Project Plan, determine whether it will be created as a single plan or as a composite of individual functional area project plans. For more information on using task folders to manage multiple project plans, see “Best Practices for Managing Multiple Projects in the Task Manager” on page 306.

PROCEDURE FOR DEVELOPING THE FINAL PROJECT PLAN
Overview

The Project Plan is developed in two phases. The Final Project Plan is developed in the Design Phase based on the milestones established in the Preliminary Project Plan. Once completed, the Final Project Plan is then managed using the Task Manager throughout the remainder of the CDP phases.

Final Project Plan

The Final Project Plan may be created as a single plan or as a composite of individual functional area project plans. Individual project plans may be owned by functional area managers, but the owner of the Final Project Plan is the Product Manager responsible for the project.

1. Ensure that the Preliminary Project Plan Microsoft Project file is consistent with any changes that may have been made in the Task Manager.
2. Each functional area manager should create a task-level schedule using the established milestones as guidelines. Do not change any milestones. Organize task folders according to the phases of the CDP. The task-level schedule is based on:
    - Technical Design Documents (for coding tasks)
    - Any updates to effort estimates based on completing the Technical Design Documents
    - Software Testing Plan (for testing tasks)
    - User Interaction Specification (for UI development tasks)
    - Documentation Plan (for documentation tasks)
    - Field Test Plan (for beta and other field testing tasks)
    - Launch Team meeting dates
    - Any subcontractor plans
    - All peer review, training, software quality assurance audits, and other activities identified in the Product Plan. Include all items in the process tailoring section.
3. Assign owners for each task.
4. Assign appropriate items to subcontractors, if applicable.
5. Identify all milestones. All phase exit dates and milestones established in the Preliminary Project Plan should be marked as milestones.
6. Identify the critical path items.
7. Identify any dependencies among tasks.
8. Submit any proposed milestone changes to the Product Manager for review and cross-functional coordination.
9. Once any proposed milestone changes are addressed, the Product Manager should submit the Final Project Plan (either assembled into a single file or as a collection of individual files) for peer review using the the Iterative Document Review and Approval Process.
10. Once approved by required reviewers, export each Microsoft Project file into the Task Manager.

Managing the Project Plan

The Task Manager provides a number of configurable tools to enable managers to actively manage the use of the project plan. Because of the superior reporting and visibility capabilities of the Task Manager, it is highly recommended that the Task Manager be used exclusively to manage the project plan once it is completed and imported. There is no need to continue to maintain the original Microsoft Project files in parallel.

All managers are responsible for managing the activities and deliverables in their project plans throughout the life of the project. The Task Manager provides managers with the ability to receive an approval request when a task owner changes the status or the due date of a task, or any other specified field. Overdue notifications can be automatically sent when a task is overdue.

Some recommended settings include:

Approval for Changes to Task Status and Dates

Configure the Change Approval section to require management approval for all changes to task status and dates. This enables managers to control any proposed changes to the plan that may impact the overall schedule.

You may also wish to enable change management for other fields.

Overdue Tasks

Configure the Overdue Tasks sections to notify and auto-escalate tasks that become overdue. This enables managers to take prompt corrective action.

In effect, managers create their own policies for how the project plan will be managed by configuring the Task Manager Settings.

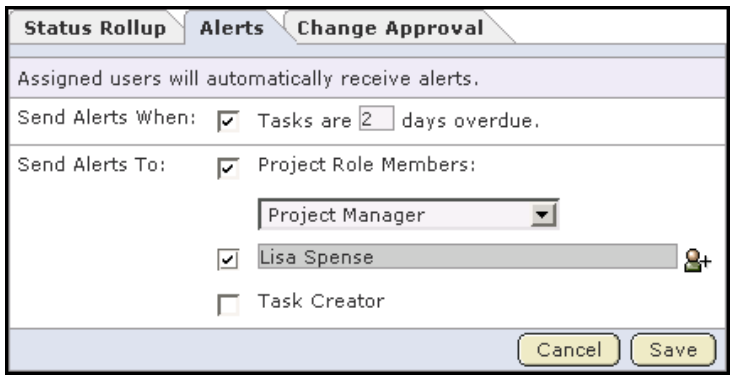


Figure 27. Configure Overdue Tasks Alerts

SW-CMM NOTE

The SW-CMM Level 2 Key Process Areas Software Project Planning and Software Project Tracking and Oversight require documented processes for creating and maintaining your project plan. Documented processes are also required for monitoring use of the plan and managing changes. Following the guidelines in this section and using the Task Manager as recommended satisfies these requirements.

See *SourceForge and the Capability Maturity Model for Software, A Supplement to the SourceForge Collaborative Development Process* for specific SW-CMM requirements satisfied by effective processes for creating and maintaining your project plan.

Table 27. Project PlanTemplates

Ref#	Template	File Location
1.5.1	Preliminary Project Plan	\proddef\[1.5.1]projplan.mpp \proddef\[1.5.1]projplan.pdf
2.7.1	Final Project Plan	\design\[2.7.1]finalprojplan.dot \design\[2.7.1]finalprojplan.pdf
7.10.1	Procedure for Developing the Project Plan	\crosslife\[7.10.1]projplanprocedure.dot \crosslife\[7.10.1]projplanprocedure.pdf

Best Practices for Working with Microsoft Project and the Task Manager

As with many SourceForge features, the Task Manager offers users a variety of choices when setting up and managing activities. This section will describe the ways in which the SourceForge Task Manager and Microsoft Project can be used individually and in combination to best manage tasks and project plans.

Microsoft Project Export

The SourceForge Task Manager Add-in for Microsoft Project enables you to export an existing Microsoft Project file from Microsoft Project into the SourceForge Task Manager.. The Export function enables you to take aa Microsoft Project plan, export it into the Task Manager, then manage the tasks more effectively using the advanced reporting and visibility features provided by the Task Manager. During the export process, the Task Manager maintains all of your task descriptions and start and end dates. It also matches resource names where possible, or allows you to select a user to whom to assign any unmatched tasks. At any time, you can re-import updated task data from SourceForge back into Microsoft Project.

It is not necessary to begin your projects using the export function; you can create all of your tasks directly in the Task Manager. However, for users who are already using Microsoft Project for project management, the export feature enables you to leverage the time and effort already invested in creating and managing your Microsoft Project plan. Creating an Microsoft Project file first, then using the export function, can also be less time-consuming than creating each individual task in the Task Manager.

Some benefits of using Microsoft Project Export are:

- Allows you to create a project plan in a familiar application.
- Allows you to use an existing project plan without having to re-enter data.
- Allows you to more quickly create a large project plan.
- Allows users who do not have Microsoft Project to access task data.

Native Project Planning

Using Microsoft Project import can greatly simplify the process of setting up a new project plan for users who are familiar with and are already using Microsoft Project. However, if you are not using Microsoft Project or do not have a large number of tasks to enter, you may wish to create your project plan directly in the Task Manager. This is easily done using either the Task Manager interface.

You may wish to use native project planning when:

- You do not use Microsoft Project.
- You have exported a project plan from Microsoft Project, but have no further need to maintain the Microsoft Project file.
- You have a relatively small number of tasks to enter.
- You are managing "one-off" tasks, or tasks that are not closely associated with a project plan.

A recommended best practice combining the two methods of project planning is to create your initial project plan in Microsoft Project, export it into the Task Manager, then continue to add, modify, and delete tasks using the Task Manager. The additional management, reporting, and visibility features provided by the Task Manager often eliminate the need to maintain a parallel plan in Microsoft Project.

It is important to note that all Task Manager functions work the same way whether you export tasks from Microsoft Project or create them directly in the Task Manager.

Best Practices for Managing Multiple Projects in the Task Manager

When setting up projects in SourceForge, some consideration must be given to how to best manage multiple development projects. You may choose to manage each development project in a separate SourceForge project, which will enable you to easily manage each project independently and avoid creating confusion with other unrelated material. You may also elect to group closely related projects into a single SourceForge project in order to share common artifacts such as tracker artifacts, documents, source code, or tasks. Please see "Managing Multiple Projects" on page 277 and on page 286 for additional considerations regarding multiple projects.

Task Folders

The Task Manager provides a one-to-one mapping between SourceForge project and Task Manager instance. While you cannot create unique Task Manager instances for separate development projects being managed within a single SourceForge project, the same goal can be accomplished using task folders.

The SourceForge Task Manager Add-in for Microsoft Project allows you to export multiple Microsoft Project files into separate task folders within the same SourceForge project, while enabling different managers to manage their own projects or sections individually. This feature allows you to do the following:

Manage sequential or otherwise related development projects in a single SourceForge project

When your projects are highly dependent on one another with the need to share artifacts, it is often beneficial to manage them in a single SourceForge project. Sequential or otherwise related development projects can be set up and managed as unique task folders. When the first project is finished, all tasks will display the 'Completed' status, and the data in that task folder is available for future reference. The next project can then be started as a new task folder.

This method is also useful when managing a number of development projects simultaneously.

Subdivide a project into multiple subprojects, each managed by a different manager

The project plan for a single development project is often comprised of multiple project plans specific to each functional area of the organization. For example, you may have an engineering project plan, a software testing project plan, a training plan, and any number of other project plans that are created and managed by individual managers. While all contribute to the Final Project Plan, and must be managed as a cohesive unit, many organizations prefer that individual managers retain the primary responsiblity for managing the daily activities of their organizations.

In such cases, you can import the project plans of each functional area into separate task folders.

Multiple Functional Area Project Plans as Task Folders

You may combine these strategies for using task folders when you are managing both multiple development projects and multiple functional area project plans in a single SourceForge project.

Task Folders and CDP Phases

In order to support adoption of CDP processes and facilitate phase exit reviews, it is highly recommended that all project plans be organized according to the phases of the CDP. If you are using multiple task folders to manage functional area project plans, each task folder should be organized according to the phases of the CDP. Task folders can be created for each CDP phase to manage the activities and deliverables due for completion in that phase. When conducting each phase exit review, the Task Summary page for each functional area will provide the necessary information regarding the completion of all items required to exit that phase.

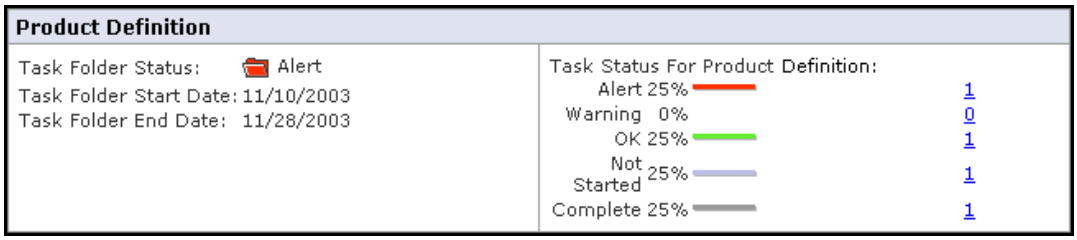


Figure 28. Task Summary for Product Definition Phase

Best Practices for Tracking Effort with the Task Manager

SourceForge provides several alternatives for tracking the effort expended on individual tasks. This section will cover best practices for using the Task Manager and the Tracker for measuring effort.

Task Manager Hours Fields

The Task Manager provides Estimated and Actual Hours field in the Task Details section of each task that can be used to track the number of hours spent.

Tracker Custom Fields

Another more detailed alternative for tracking effort expended on management versus execution activities is to use tracker artifact with custom fields to track each type of activity separately. For example, you can create custom fields for Management Hours and Execution Hours. For additional detail, create additional custom fields for Projected Management and Projected Execution Hours. You can then create an association between the Task Manager task and the Tracker artifact to allow them to be easily updated together..

SW-CMM NOTE

Several of the SW-CMM Level 2 and Level 3 Key Process Areas require that planned versus actual effort be tracked for executing and managing various activities. Following the guidelines in this section and using the Task Manager as recommended satisfies these requirements.

See *SourceForge and the Capability Maturity Model for Software, A Supplement to the SourceForge Collaborative Development Process* for specific SW-CMM requirements satisfied by effective processes for creating and maintaining your project plan.

Best Practices for Measuring Progress and Reporting to Management with the Task Manager

The Task Manager is designed to simplify reporting by providing continuous access to real-time project status data. At any time, managers can access the Task Manager to get instant updates ranging from high level summary information about the general health of a project to task level detail complete with comments, status change history, and action plans for correcting slippages.

Using the Task Manager to manage your project plan allows managers to greatly reduce the amount of time and effort needed to collect status updates, prepare, and distribute periodic progress reports. The following section covers best practices for using the Task Manager to satisfy your reporting requirements.

Regular Status Reports with the Global Development Dashboard™/ Project Dashboard

Many organizations have regular status meetings to review status, progress, and issues. When using Microsoft Project to manage your project plan, it is a common practice to prepare for each meeting by updating the project plan, distributing it to meeting attendees, and printing out hardcopies for reference during the meeting. After the meeting, the project plan must be updated manually, re-distributed to attendees, and often posted to a website or shared directory for future reference.

When using the Task Manager to manage your project plan, the time-consuming and manual processes associated with reliance on Microsoft Project are eliminated. The Task Summary page and the SourceForge Global Development Dashboard/Project Dashboard provide all of the information you need to conduct status meetings with little or no additional preparation. The Project Dashboard provides managers with an at-a-glance overview of the status of each of their projects. It provides summary data on the number and status of the tasks and tracker artifacts in each project, and calculates project overrun and underrun statistics , based on the difference between estimated and actual hours spent on project tasks and tracker artifacts.

If your meeting location provides network access, connect to SourceForge and access the Task Summary page or the Project Dashboard for the projects that you want to discuss. A projector is recommended, if available, to allow all attendees to view the Task Summary page or the Project Dashboard simultaneously. If network access is not available, use the print function to print copies of the pages for reference during the meeting.

.

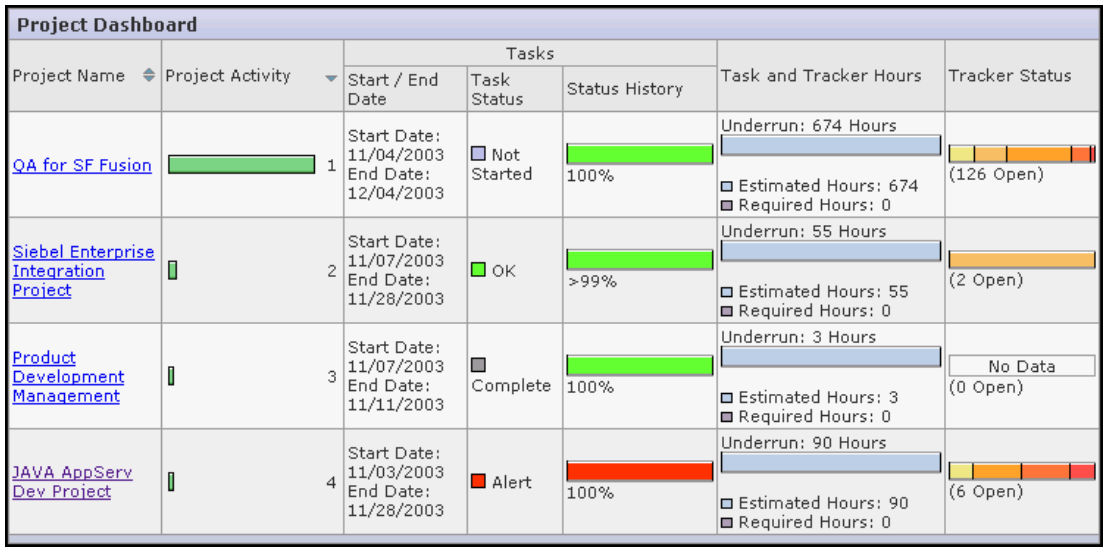


Figure 29. Project Dashboard

Many organizations also require periodic written status reports that are manually rolled up to senior management. Using the Task Manager and the Project Status Report, these written reports can be minimized or eliminated.

Phase Exit Reviews

For Phase Exit Reviews, the Task Summary page and the Project Dashboard can be used along with a checklist of specific items that must be completed prior to exiting the phase. Items on the checklist associated with the completions of specific deliverables can be confirmed by checking status of each item in the Task Manager. Other items such as numbers of outstanding issues or review of risks and resources can be confirmed using the Tracker or other SourceForge tool.

Checklists are provided in this manual at the end of each CDP phase.

Reporting to Senior Management

If your organization does not have regular status meetings or other reports to management, it is highly recommended that a mechanism be put in place to ensure that senior management staff receive periodic updates into the status of each project. Using the Task Manager and the Project Dashboard as described above provides the means for senior management to access status reports at any time. You may wish to initiate a corporate policy or similar program to encourage all staff with a need for such information to actively use the Task Manager and Project Dashboard. Such an initiative may be particularly useful in cases where senior-level

managers may not use SourceForge in their dailiy activities. A recommended best practice is to provide senior managers with project membership and Role-based Access Control permissions to access the Task Manager for all projects they are responsible for overseeing. SourceForge users automatically have access to the Project Dashboard for all projects in which they have Task Manager or Tracker Access permissions.

SW-CMM NOTE

Many of the SW-CMM Level 2 and Level 3 Key Process Areas require periodic review of various project activities by senior management. Following the guidelines in this section and using the Task Manager as recommended satisfies these requirements.

See *SourceForge and the Capability Maturity Model for Software, A Supplement to the SourceForge Collaborative Development Process* for specific SW-CMM requirements satisfied by effective processes for creating and maintaining your project plan.

Best Practices for Software Configuration Management using SourceForge

Software Configuration Management defines the practice of controlling changes to documents, source code, and other work products produced throughout the product lifecycle. SourceForge provides all the tools needed to manage and control changes to all such artifacts.

Software Configuration Management Policy

The Software Configuration Management Policy is a set of standards for establishing and maintaining the integrity of software products throughout the software development lifecycle. The SCMP breaks down into four key areas:

1. Software configuration management activities are planned.
2. Selected software work products are identified, controlled, and available.
3. Changes to identified software work products are controlled.
4. Affected groups and individuals are informed of the status and content of software baselines.

The Software Configuration Management Policy is broken into several phases as outlined below. Each phase includes entry and exit criteria.

Policy Creation and Management

The SCM group, comprised of the same membership as the Software Change Control Board (SCCB), is responsible for implementation and management of the Software Configuration Management Policy. To initiate a project, the SCCB must review and update the Software Configuration Management Policy as necessary to meet specific project needs. The SCCB must meet periodically to review phase objectives outlined below. One member of the SCCB is charged with administrative control of the Software Configuration Management Policy, designated as SCM Lead.

Entry criteria:

- Product Plan complete
- SCCB members identified
- Periodic SCCB meeting scheduled

Exit criteria:

- Updated Software Configuration Management Policy published and announced to stakeholders
- SCM Lead identified
- SourceForge project and SCM environment created under administrative control of SCM Lead

SCM Training

All members of the software project team using SCM must be trained on the Software Configuration Management Policy and particular operation of the SCM system. Training may not be required if all members possess SCM knowledge from previous training or project experience. In this case, each team member must meet the stated training objectives, whether or not they attend formal training. SCM training should be managed through your Training Program and may be project or organizational training. See "Managing your Organization's Training Programs using SourceForge" on page 260 for additional information on training programs.

Entry criteria:

- SCM project team identified
- Team members requiring training identified
- Training plan and objectives created

Exit criteria:

- Entire SCM project team meets training objectives

SCM Identification

The SCCB must identify resources (code, documentation, configuration items, supporting libraries and material) to be placed under SCM control at project inception. This list must encompass the entire project lifecycle and be published to stakeholders via SourceForge. The preferred form is a complete list of deliverables associated with storage and publication mechanisms, as well as responsible parties.

SCM Administration

The SCM Lead is charged with setting SCM user access based on the SCM project team makeup.

SCM Control

Role-based Access Control will allow you to permit access to the SCM system only by the SCM team.

SCM Status

Resources under SCM control will be periodically tagged with an identifier in the form: sf_major_minor_point-name. Each tag signifies a baselined version of code and supporting materials suitable for the Product Build Process. Each tag is created by agreement of the SCCB.

Product Build Process

The product must build from SCM to final distribution media via a programmatic mechanism. Each build will be based on an SCM tag as specified in the SCM Status section.

Document Change Control

A document change control policy should be written that describes the process by which changes can be made to critical project documents after change control has been initiated. The goal of this change control policy is to ensure complete visibility of document changes to all concerned parties and to track changes in a consistent manner for future reference.

Documents that will be subject to this change control policy are identified in the Product Plan. Generally, only key project documents that affect the ability of a group to meet release-related deliverables are placed under change control. Documents with significant cross-functional impact should also be placed under change control. This set of documents must include, but is not limited to, the Product Plan, Product Requirements Documents, and Technical Design Documents. Other documents may be placed under change control if desired.

Change control goes into effect after all required reviewers of the document have provided their final sign off.

Change Request Tracker

Change requests are made via the change request tracker for each project. It is recommended that the Launch Team be responsible for monitoring the change request tracker and addressing all change requests in a timely fashion according to their priority level. Monitoring should be used to send email notifications of all activity to project members responsible for project change control.

Priority Levels and Response Times

The time within which the Launch Team is required to respond to all change requests should be established in the document change control policy according to the priority level attached to each request. The table below provides an example response time matrix.

Table 28. Required Change Request Response Times

Priority	Criteria	Response Time
1	Urgent. Has critical impact on other deliverables and/ or schedule.	1 business day
2	High. Has critical impact but does not impact other deliverables and/or schedule.	1 business day
3	Medium. Workaround available and does not impact other deliverables and/or schedule.	3 business days
4	Low. Desirable, but not critical and does not impact other deliverables and/or schedule.	5 business days

DOCUMENT CHANGE CONTROL POLICY

Overview

The Document Change Control Policy describes the process by which changes can be made to critical project documents after change control has been initiated. The documents to be placed under change control are identified in the Product Plan

Process

1. Identify issue.
2. Create new artifact in the project's change request tracker.
3. Artifact must include:
    - Document title, version number, and URL
    - Document owner
    - Priority level
    - Summary of change, including cross-functional impact statement if priority level is (1) urgent or (2) high
4. The Launch Team will review each artifact and respond to the submitter according to the turnaround time specified for each priority level.
    - (1) Urgent - 1 business day
    - (2) High - 1 business day
    - (3) Medium - 3 business days
    - (4) Low - 5 business days
5. The Launch Team is responsible for escalating any (1) urgent or (2) high priority change requests to the Management Approval Team if they cannot be resolved promptly.
6. All priority level (1) urgent change requests that impact the delivery date or expected functionality of the product must be approved by the Management Approval Team.
7. If the change request is approved, the approver will change the status of the change request to "Approved" and assign it to the appropriate party for execution.
8. The resulting change to the document in question must reference the change request tracker artifact number in the revision history.
9. If the change request is denied, the approver will change the status of the change request to "Denied" and provide the reason for the denial.
10. The submitter may escalate the change request if desired by changing the status to "Escalated." The Launch Team must review all escalated change requests with the Management Approval Team within 5 business days of receipt.

SW-CMM NOTE

The SW-CMM Level 2 Key Process Area Software Configuration Management requires documented procedures for conducting SCM activities. Following the guidelines in this section and using SourceForge as recommended satisfies these requirements.

See *SourceForge and the Capability Maturity Model for Software, A Supplement to the SourceForge Collaborative Development Process* for specific SW-CMM requirements satisfied by a comprehensive Software Configuration Management policy.

Table 29. Software Configuration Management Templates

Ref#	Template	File Location
7.11.1	Software Configuration Management Policy	\crosslife\[7.11.1]scmpolicy.dot \crosslife\[7.11.1]scmpolicy.pdf
7.12.1	Document Change Control Policy	\crosslife\[7.12.1]change.dot \crosslife\[7.12.1]change.pdf

Best Practices for Software Development and Quality Assurance

Software development is a broad discipline, encompassing a wide range of coding languages, techniques, and standards. The goal of the CDP is not to attempt to identify a "best" method, nor is it to instruct users in the art and science of developing code. The focus of the CDP is to present a series of best practices covering processes, procedures, and uses of SourceForge that can be applied in any software development organization, independent of the coding methods they have chosen.

This section provides a series of best practices that the VA Software Product Development organization has found useful when developing SourceForge Enterprise Edition and other software products. They are general enough to be applied in a wide variety of development environments, and are intended to provide suggestions for supplementing your existing software development practices. The order of presentation below does not imply any priority associated with each recommendation.

Code Review

Throughout the development of source code, all code should be peer-reviewed prior to its completion. Peer review consists of reviewing and rating another software engineer's code on criteria such as correctness, performance, commenting, style, and ease of comprehension, maintenance, and extension. The VA Software Product Development organization conducts informal code reviews whenever coding issues are encountered and formal code reviews at appropriate stages throughout the development process. Code is always reviewed prior to the completion of any milestone such as Technical Unit Testing.

Please see Table 17, "Development Phase Templates," on page 159 for the CDP Code Review template.

Dedicated Code Review Environment

A dedicated physical environment should be provided that can be used at any time for code review. The conference room or other location should contain all the tools needed to quickly and easily conduct code review and other impromptu collaborative meetings, such as network access, projectors, and whiteboards. It should be open at all times, require no reservations, and be accessible to all software engineering staff.

The conference room used by VA Software Product Development is equipped with two LCD projectors. This enables engineering staff to display the TDD or other design materials on one, and the code being reviewed on the other. A shared whiteboard application, e.g. Placeware or WebEx, is used when conducting code reviews with our remote teams in India and the Ukraine. This allows all staff to see the same code and to collaborate on real-time changes, eliminating the possibility of omissions or errors due to lack of shared understanding.

Coding Standards

A coding standards document should be drafted describing the internally agreed-upon set of standards to which all developers must adhere. In order to ensure compliance with the established coding standards, VA Software recommends the use of filter software to scan code to validate its adherence to standards before allowing check in to your chosen Software Configuration Management tool. There are numerous open source utilities that can be used for this purpose.

The filter in use by VA Software Product Development is configured to provide two possible outputs. A warning is retuned when the filter detects issues that are not critical but should be addressed at some point. Issues of this nature do not block code check in. An error is returned when the filter detects issues that are critical and must be resolved before the code can be checked in. These errors block code check in.

Unit Testing

Unit test cases should be written prior to developing the code. This ensures that the test case matches the TDD or other specification. It also reduces the possibility that the test case will be written in accordance with code already completed which may or may not adhere to the design specification. To further ensure objective adherence to the design specification, each unit test case should be written by a software engineer other than the one writing the code to be tested.

An automated unit testing framework should be used to enable any software engineer to check out all applicable code in the SCM repository and automatically run all unit test cases. A VA Software software engineer performs this process daily each afternoon. All software engineers are then required to correct any issues with their code prior to leaving the office for the night. Another option is to require each software engineer to successfully perform this process prior to being allowed to check in code.

Running unit testing and requiring code fixes to be completed daily ensures that the code in your SCM repository can be successfully built at any time. This is especially valuable for geographically dispersed development teams. For example, as the VA Software development team in California is leaving for the day, the team in India is just beginning work. The Indian team can work with confidence knowing that they have been left with a stable platform on which to work.

One-Step Build Process

A one-step build process should be available to anyone in the software engineering organization. A one-step process enables any user to check out all applicable code in the SCM repository and build it with a single command. This concept of an "always buildable product" means that a broad set of internal staff can easily take a snapshot of the nascent product for purposes specific to their function or team. It also frees the release engineer, or other responsible individual or team, from continually having to build and provide product to anyone in need of it.

SourceForge Tracker / Code Associations

The SourceForge Tracker / Code Associations functionality is used extensively by VA Software Product Development staff in building SourceForge and other software products. All code check ins reference the artifact number of the bug or product requirement with which they are associated. This enables traceability from the requirements level all the way through the CDP to the code level. When looking at a code module, developers see references to bug and enhancement request tracker artifacts that are associated with the module. This significantly aids comprehension regarding the motivation for design and implementation decisions.

Early Engineering Estimates

As early as possible in the Product Definition phase, the Software Engineering team should be engaged to review the proposed feature set and provide preliminary effort estimates. This allows staff responsible for requirements management to better estimate a realistic scope for the project and reduce the proposed feature set as necessary. Accurately scoping the project early in the CDP greatly reduces the amount of unnecessary work that might otherwise be done defining product requirements that ultimately cannot be implemented.

As defined in the Product Definition phase of the CDP, VA Software conducts this activity in conjunction with the completion of the Product Plan. Once the full product feature set is identified and the first review copy of the Product Plan is distributed, each software engineer reviews the proposal for his or her functional areas and provides preliminary estimates as to the engineering effort involved. Software Product Management is then able to prioritize features and adjust the scope of the project before completing the Product Plan.

More detailed engineering estimates are done later in the CDP when the TDD are being written, and additional adjustments to the release scope may need to be made. However, early engineering estimates help considerably in ensuring an accurate release scope. They also provide a baseline for comparison against the actual effort required to implement each feature.

Please see *Chapter 3, 1.3. Product Plan* on page 76 for additional detail on completing the Product Plan.

Formal Document Handover

When key documents are completed, a formal document handover meeting should be conducted. The meeting does not need to be lengthy, its goal is simply to ensure that everyone understands that a milestone has been achieved (the completion of the document) and that the document in question is either closed, placed under change control, or transfered to the ownership of another team or individual.

At VA Software, formal document handovers are often conducted as part of the weekly Launch Team meetings. For lengthy or more complex documents such as the Product Requirements Documents or Technical Design Documents, separate meetings are often called to allow sufficient time to go over any questions or issues regarding the content or status of the documents. All status changes are noted in the corresponding Task Manager task, and if applicable, the status of the document is changed in the Document Manager.

Document Status

Policy at VA Software requires that Product Development staff post all draft documents to the Document Manager as soon as they are begun. The Draft, Review, and Final attributes are assigned to every document version posted. The goal of posting early and frequent drafts is to provide early visibility into the content of the document, allow other staff to easily identify the owner of a document early in its lifecycle, and permit early input into critical document content. It is also a good way to monitor and support the progress of offsite staff.

It is important to provide guidelines for reviewing and using the content in draft documents. All staff should understand that documents marked Draft are to be reviewed for informational purposes only. The content should never be used as the basis for beginning coding or taking other actions. Review comments should also be limited to identification of potentially critical issues until a document review is initiated and the document status is changed to Review.

Instant Messaging

VA Software Product Development staff consistently use an instant messaging tool or IRC channel for quick internal communication. Policy requires that all staff log into the channel whenever they are working. Using an instant communication tool is valuable for answering quick questions and to determine availability for impromptu meetings. It is also a valuable tool for enabling communication among geographically disperse teams.

Your chosen instant messaging or IRC tool should be hosted on an internal server, and communication between locations should be done via secure, encrypted means. Recording all communication for future reference is also recommended.

Daily Bug Charts

Using Tracker Reporting, daily bug activity charts should be generated to provide visibility into the status of all bugs associated with the product under development. Software Testing staff at VA Software generate daily bug activity charts early each morning before the majority of the Product Development staff are in the office. Weekly reports are also generated for a weekly Product Development Status report that is distributed to the entire organization. The report in Figure 30 on page 322 is generated daily by using the Export feature of the SourceForge Tracker, then importing into a Microsoft Excel spreadsheet.

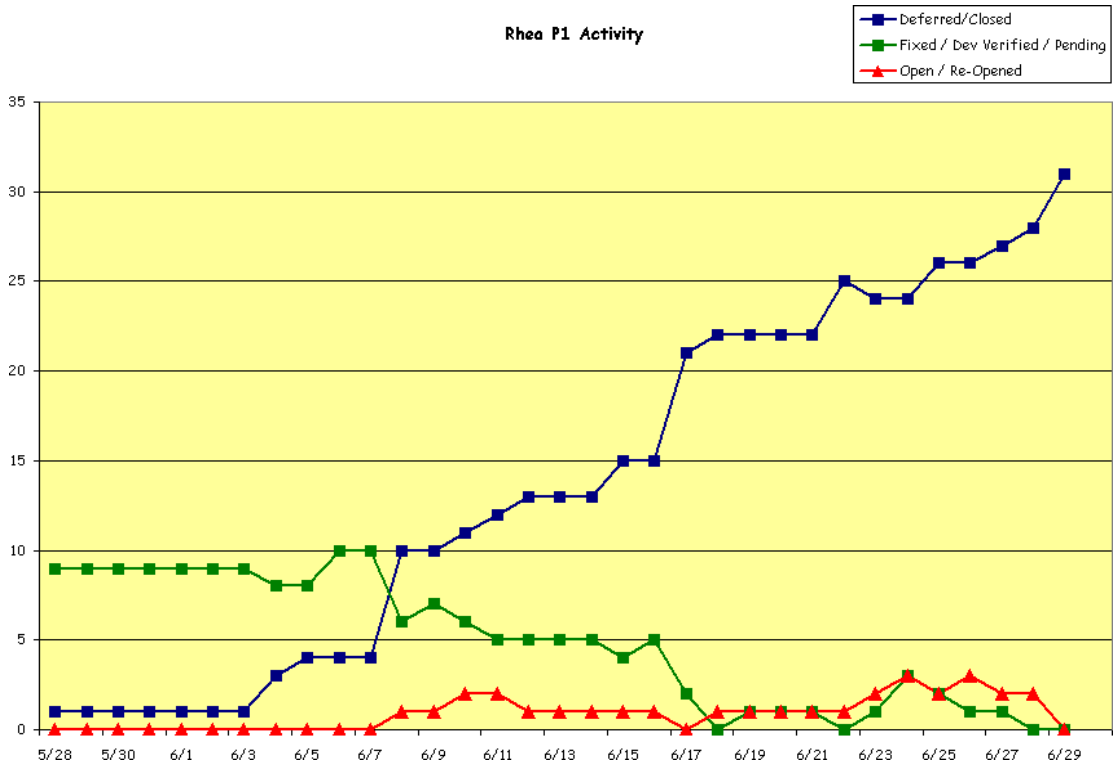


Figure 30. Daily Bug Activity Report

Bug Handling Processes

The VA Software Testing organization has implemented a series best practices for handling bugs. In addition to publishing daily bug activity reports, the Software Testing team actively uses the Status field of the SourceForge Tracker to identify the status of all bugs. The artifact status designations are used as follows:

- Pending: Used for bugs that the Software Testing organization cannot reproduce and where additional information is needed.
- Deferred: Used for bugs that have been deferred for correction in a future release. These are usually low severity bugs.
- Fixed: Used for bugs once they have been corrected by a software engineer.
- Verified: Used for bugs once they've been verified by a different software engineer. All bugs are also verified on both a standard product installation and an installation that is configured to match the one on which the bug was initially identified.
- Closed: Used for bugs that have been tested and confirmed as fixed by Software Testing.

All Product Development staff are required to re-verify a bug before entering it into the SourceForge Bugs Tracker. The bug must be reproduced in the environment in which it was originally identified, plus reproduced on a standard product installation. This helps identify whether the bug is specific to a particular product configuration and reduces the amount of testing required of the Software Testing organization.

Once Final System Testing has begun in the Release Preparation phase, the Software Testing organization begins holding daily bug resolution meetings. This enables swift resolution of new and existing bugs, and ensures that all impacted functional groups understand which bugs will be fixed and which will be deferred to a future release.

Please see "Managing Product Requirements Using the SourceForge Tracker" on page 280 for additional bug handling best practices.

Dedicated Development and QA Environments

Dedicated machines, or system configurations, should always be provided for software engineers to use when developing product code. Similar dedicated machines should be provided for software testing activities. The applications, operating systems, and other software loaded on these machines should be strictly controlled to ensure that only appropriate, non-conflicting software is present. This also ensures consistency when testing the product on different machines. Development and testing should never be done on personal machines, i.e. those assigned to individuals for general use.

A similar best practice in use at VA Software is to always maintain installations of all previous, supported versions of the product. Installations of all currently supported configurations of SourceForge, operating systems, and databases are also maintained (the

product's reference environment). This allows the Software Testing team to quickly verify bugs that are reported against earlier product versions. It also enables staff to confirm whether a bug was present in any supported reference configurations other than the one in which it was reported.

Self-Residency

VA Software strongly believes in using our own product as much as possible. As "power users" of SourceForge, VA staff consistently submit as many bugs and enhancement requests as possible, in many cases exceeding those submitted by users external to the organization. By remaining actively engaged in continually using and improving our product, we are able to thoroughly understand users' issues and requirements.

Customer Advisory Council

VA Software has created a Customer Advisory Council (CAC) consisting of selected customers who participate in a variety of product lifecycle activities. The CAC comes together periodically with VA Staff to discuss the benefits, challenges and future direction of the SourceForge product. Members participate in product definition activities, field testing, post-release analysis, and other activities. Benefits to members include frequent opportunities to communicate directly with VA senior staff, to directly input enhancement and other requests, and to participate in defining future VA product strategy. Members are invited to attend one or more off-site events annually, and to participate in quarterly Webinars covering a variety of topics. Members may also be invited to participate in additional activities from time to time.

Members are responsible for making efforts to attend all scheduled CAC events and participate as requested by VA Software. For membership information, please contact cac@vasoftware.com.

Glossary

Administrator An administator is a SourceForge user that has initiated, or controls, a SourceForge-hosted project. The administrator may or may not be the Project Manager, although typically they are.

Best Practices Best practices are recommended uses of SourceForge to enable the achievement of process improvement and other goals.

Capability Maturity Model for Software (SW-CMM) The Software Engineering Institute's SW-CMM is a framework that describes the key elements of an effective software process. It defines five process maturity levels ranging from Level 1 to Level 5.

Capability Maturity Model Integration (CMMI) The Software Engineering Institute's CMMI is an integrated framework that builds on and extends the best practices of the Capability Maturity Model for Software, the Systems Engineering Capability Model, and the Integrated Product Development Capability Maturity Model.

CMM-Based Appraisal A CMM-Based Appraisal for Internal Process Improvement (CBA-IPI) is the formal appraisal method recognized by the Software Engineering Institue for assessing progress against SW-CMM goals.

Cross-Lifecycle Principles Cross-lifecycle principles are processes and activities that are conducted continuously, either in parallel to or throughout the CDP phases applied to individual projects.

Design Phase The Design phase is the second phase of the CDP, during which technical and other design documents are created and approved.

Developer A developer is a SourceForge user that is part of a project team, typically involved to contribute code.

Development Phase	The Development phase is the third phase of the CDP, during which code and supporting materials are created. The first round of unit and system testing are also conducted during this phase.
Document Manager	The SourceForge Document Manager provides users with a repository for storing, reviewing, managing, and archiving all product and related documentation.
FAQ	Frequently Asked Questions. An FAQ is typically prepared as a supplement to product user documentation.
Forums	SourceForge Forums provides users with an unlimited number of customizable discussion forums in which to discuss a variety of topics.
ISO 9000	A family of standards representing an international consensus on good management practices.
Key Process Area (KPA)	Each maturity level of the SW-CMM is comprised of KPAs that define a series of activities that when performed consistently and effectively, help organizations meet established goals for process capability.
Management Approval Team (MAT)	The Management Approval Team is defined as the group of managers who initiate, sponsor, approve, and fund the project.
Methodology	An organized, documented set of procedures, processes and guidelines for one or more phases of the software development life cycle, such as analysis or design.
Milestone	A milestone is a significant activity or deliverable upon which other activities or deliverables are dependent.
Monitoring	The SourceForge monitoring function enables users to subscribe to email notifications generated when the status of a monitored artifact changes.
Peer Review	Peer review is the review of a document, code module, or other work product by its creator's peers. The goal of peer review is to identify defects, obtain input, and facilitate intergroup coordination.
Phase Deliverables	Phase deliverables are the series of deliverables recommended by the CDP for completion prior to exiting the phase.
Phase Exit Criteria	Phase exit criteria are the activities and deliverables that must be completed before a CDP phase can be exited.
Policy	A policy defines the accepted organizational standards for various actions.

Procedure	A procedure is an established set of steps for completing an activity. In the CDP, procedure is used to describe documented organizational standards for conducting process-related activities.
Process	A process is an established set of steps for completing an activity.
Process Certification	Process certification refers to achieving a formal rating by a certified assessor against an established methodology. SW-CMM process certification consists of self-assessment with the assistance of an SEI authorized lead assessor.
Process Maturity Models	Process maturity models provide defined models for functioning at a high level of process efficiency. The SEI SW-CMM is an example of a process maturity model.
Process Methodology	Process methodology refers to the methods used to achieve process improvement goals associated with process maturity models. The CDP is an example of process methodology.
Product	Product refers to the software product being developed.
Product Definition Phase	The Product Definition phase is the first phase of the CDP, during which the product requirements are fully defined and agreed to by product stakeholders.
Product Roadmap	The Product Roadmap is the vehicle for defining the long-term product strategy of an organization.
Project	Project refers to the set of activites required to develop, test, release, and support a product. Each product has an associated development project.
Task Manager	The SourceForge Task Manager provides managers with visibility into and control of their project plans. It provides real-time reporting, project metrics, and visual representations of the status of a project.
Project Member	Project member is the SourceForge default user class for users who are members of a particular project.
Release Phase	The Release phase is the fifth phase of the CDP, during which the product is released to its intended audience.
Release Preparation Phase	The Release Preparation phase is the fourth phase of the CDP, during which software testing activities are conducted, product defects are corrected, and the product and supporting materials are prepared for the product release.
Role	A role is a special identity that can be assumed by assigned users only. Roles are created using Role-based Access Control.

Role-based Access Control	Role-based Access Control (RBAC) is the SourceForge user permissions system enabling an administrator to define and assign roles that provide customized access to specific tools and functions.
Software Engineering	Software Engineering refers to all staff primarily engaged in technical activities related to software development.
Software Engineering Institute (SEI)	Carnegie Mellon's Software Engineering Institute developed and maintains the Capability Maturity Model for Software and other process improvement models.
Software Engineering Process Group	The Software Engineering Process Group is the team responsible for the organization's proess-improvement activities.
Software Process Database	A Software Process Database is a repository where the organization's standard process templates, policies and procedures, guidelines, and other process-related materials are stored for easy access by all staff members.
Software Product Management	The Software Product Management category, also known as Project or Program Management, refers to all staff engaged in overall management of the product and its associated development project.
Software Quality Assurance	The Software Quality Assurance function refers to periodic quality audits of CDP activities and deliverables.
Software Testing	The Software Testing category refers to all staff primarily engaged in software quality and testing-related activities.
SourceForge Collaborative Development Process (CDP)	The CDP is a comprehensive end-to-end software development lifecycle model. It provides a process methodology for implementing or improving software development processes and supports the goals associated with the SW-CMM Levels 2 and 3.
SourceForge Enterprise Edition	SourceForge Enterprise Edition is VA Software's flagship collaborative software development product. SourceForge provides a Global Development Platform that combines collaborative development tools with real-time metrics and reporting.
Sustaining Engineering Phase	The Sustaining Engineering phase is the sixth phase of the CDP, during which product maintenance and other post-release activities are conducted.
Tracker	The SourceForge Tracker is a system for tracking data related to bugs, feature requests, support requests, patches, and other data.

Tracker Artifact A tracker artifact is an individual record inside a tracker such as a bug or feature request.

User A user is a person who has registered for an account with SourceForge. This does not imply developer status or affiliation to any particular project(s).

Work Products Work products are the artifacts of a particular work flow in the development lifecycle. Some examples of work products include requirement specifications, use case diagrams or specifications, architectural scope definitions, flow charts, or data flow diagrams.

Workflow A workflow is the set of relationships between all the activities in a project, from start to finish. Activities are related by different types of trigger relationships. Activities may be triggered by external events or by other activities.

Index

E

F

G

I

K

L

M

P

Q

R

S

T

U