Configuration Management Process

1. Table of Contents

Copyright © 1997, Floyd W. Shackelford. All Rights Reserved

2. Introduction

A Configuration Management Process "is the process which controls the changes made to a system and manages the different versions of the evolving software product" [Sommerville 1996]. Configuration Management involves more than source code control, although that is an important part of it. "The purpose of Software Configuration Management is to establish and maintain the integrity of the products of the software project throughout the project's software life cycle" [Paulk *et al.* 1993]. Configuration Management includes control over "all documents which may be necessary for future system maintenance" [Sommerville 1996] as well. It is important that we distinguish between software development products and software itself. For instance, the Requirements Document is a software development product and is just as much a part of the system configuration as the actual software which implements those requirements. Configuration Management, then, is about managing the configuration of all software development products which comprise the system.

Configuration Management (CM) is one of the most fundamental processes necessary for successfully developing quality software. Sommerville [1996] goes so far as to claim that "formal CM standards and procedures [are] essential" to developing a quality software product.

Copyright © 1997, Floyd W. Shackelford. All Rights Reserved

3. Configuration Management Plan

This section essentially documents the development organization's "requirements" for Configuration Management. These requirements typically identify what we expect from a Configuration Management Process without specifying the details. Later sections will enumerate the details for implementing these policies.

3.1. Policy

It is the policy of this organization that the Configuration Management process will be used to Configuration Manage every product which qualifies as a Configuration Managed Item.

Quality Assurance will identify products which are not under Configuration Management control, but should be, and bring this to the attention of the Producer.

Process will work with the Producer to bring the identified product under Configuration Management control.

3.2. Responsibility

Here are the roles which appear in this Configuration Management process and the responsibilities assigned to each role.

Build and Integration. Build and Integration is responsible for initiating the nightly builds of the software product, following up on all build errors, and helping the developers integrate their new software into the existing product.

Configuration Management. [1] Configuration Management is responsible for creating and maintaining the Configuration Management Repository and for initiating and administering the Configuration Managed Item Identification Scheme.

Process. This person is responsible for leading the creation of the Configuration Management process and to educate development people in its use.

Producers. Producers are responsible for identifying which of their products qualify as Configuration Managed Items and using the Configuration Management process to configuration manage those products.

Quality Assurance. Quality Assurance monitors the use of this process, analyzes resultant metrics, and assists in its institutionalization.

1. For purposes of this process, it is expected that, initially, the Build and Integration and Configuration Management roles will be filled by the same person.

Copyright © 1997, Floyd W. Shackelford. All Rights Reserved

3.3. Configuration Managed Items

Generally, items which are subject to change over the life of the project (i.e. mutable) are good candidates for being classified as Configuration Managed Items. Items which are typically produced and remain unchanged should be archived but not necessarily Configuration Managed.

We will Configuration Manage the following work products:

- Needs Analysis
- Requirements Analysis
- Architecture documents
- Design documents
- Test plans, test results, run log files
- Software source code
- Build scripts, make files, build options
- Operating systems
- Patch sets
- Project plan, schedules, and budgets
- Commercial off-the-shelf (COTS) products, both source code (e.g. Rogue Wave, ILog) and binaries (e.g. Orbix, TCP/IP, Oracle)
- Third partied software (e.g. custom vendor supplied binaries)
- User's Guides
- Administrator's Guides
- All other product documentation
- Tools
- Compilers
- Training material
- Sales material
- Policy statements
- Directory structures: develop, build, install & run
- File naming conventions
- Processes and procedures
- Standards

Unless the Producer deems otherwise, we will not Configuration Manage these work products:

- Meeting minutes
- E-mail
- News groups
- Telephone messages
- Announcements
- Invoices and payments
- Contracts (they tend to be amended rather than changed)

Although these products are not Configuration Managed Items, they should be archived as part of the project.

Copyright © 1997, Floyd W. Shackelford. All Rights Reserved

3.4. Configuration Data

We will capture the following Configuration Data for each Configuration Managed Item:

- Item Identification
- Title
- Abstract
- Version History:
  - Date
  - Version Number
  - Reason for Change (Problem Report or Change Request number)
  - Description of Change

3.5. Configuration Management Tool

We will use ClearCase to manage our Configuration Management Repository.

ClearCase is sold and supported by Rational Software.

ClearCase training is available from Rational.

3.6. Training

In order for people to use this process effectively, they will need to receive the following training:

Table 1: Training

Course Name	Course Description	Who should attend	Duration
Configuration Management Process	This course covers the content of the Configuration Management Process. Students are encouraged to identify areas in which this process might be used to enhance their ability to get their work completed. This is a required course.	• Build and Integration • Configuration Management • Product Producers • Quality Assurance	tbd

Copyright © 1997, Floyd W. Shackelford. All Rights Reserved

Table 1: Training

Course Name	Course Description	Who should attend	Duration
Using ClearCase in Our Organization	This course covers how to use ClearCase to create and manage a Configuration Managed Repository within our organization. This is a required course unless the student has had previous experience using ClearCase.	• Build and Integration • Configuration Management • Product Producers • Quality Assurance	4 hours

Copyright © 1997, Floyd W. Shackelford. All Rights Reserved

4. Configuration Managed Item Identification Scheme

Every Configuration Managed Item will be assigned a unique Configuration Managed Item Identifier which differentiates that Configuration Managed Item from all other Configuration Managed Items. The Configuration Managed Item Identifier is directly associated with the Configuration Managed Item's name and its location in the Repository. This section describes the organizational structure and naming convention to be used for all Configuration Managed Items.

4.1. Configuration Managed Item Identifier

The Configuration Managed Item Identifier has the following format:

XXXnnnnn

where:

- *XXX* is the product acronym
- *nnnnn* is a unique number, assigned by the Configuration Management Team, probably sequentially generated

4.2. Configuration Managed Item Identifier Registry

For each project, a Configuration Managed Item Identifier Registry will be maintained by the Configuration Management Team. The Configuration Managed Item Identifier Registery is a table consisting of the following fields:

Table 2: Configuration Managed Item Identifier Index Fields

Field name	Description of the field
Configuration Managed Item Identifier	This is the Configuration Managed Item Identifier assigned to this Configuration Managed Item. This field represents the unique key to this table. The table rows should be sorted in ascending order on this field.
Title	This is the title of the Configuration Managed Item
Abstract	This is a brief description of the Configuration Managed Item
Repository	This is the fully qualified path to the top of the Repository
Location	This is the path and file name of the Configuration Managed Item inside the Repository

Copyright © 1997, Floyd W. Shackelford. All Rights Reserved

5. Configuration Database

Whenever you check in a Configuration Managed Item, ClearCase requires you to supply comments for why you are checking in the item. We will use this comment field, and the Configuration Managed Item version numbering scheme, to maintain the configuration database[2] specified in “Configuration Data” on page 5 for the Configuration Managed Items.

Table 3: Configuration Database Fields

Configuration Data Element	How we will capture the data element
Configuration Managed Item Identification	Every Configuration Managed Item will contain on its cover page or in its header section the following string: “Configuration Managed Item ID: XXXnnnnn”
Title	Every Configuration Managed Item will contain on its cover page or in its header section the following string: “Title: xxxxx”
Abstract	Every Configuration Managed Item will contain on its cover page or in its header section the following string: “Abstract: xxxxx”
Version History:	A new version history record is created every time a Configuration Managed Item is checked in to the Configuration Managed Repository.
Version History: Date	The date and time when the check in ocurred is automatically stored in the version history by ClearCase.
Version History: Version Number	A new version number is automatically created by ClearCase when the Configuration Managed Item is checked in to the Configuration Managed Repository.
Version History: Who	The user id of who checked in the Configuration Managed Item is automatically captured by ClearCase.
Version History: Reason for Change	When a Configuration Managed Item is checked in to the Repository, ClearCase requires that the user enter comments about why the Item is being checked in. Place the Defect or Change Request Identifier in the comments field.[a]

2. This table should be generated directly from the Configuration Managed Items in the Configuration Managed Repository. It is to unwieldy to be maintained by hand.

Copyright © 1997, Floyd W. Shackelford. All Rights Reserved

Table 3: Configuration Database Fields

Configuration Data Element	How we will capture the data element
Version History: Description of Change	When a Configuration Managed Item is checked in to the Repository, ClearCase requires that the user enter comments about why the Item is being checked in. Place a description of the changes made to the Configuration Managed Item in the comments field.[b]

a. ClearCase does not enforce the format of the contents of the comment field. This process relies upon the user to supply this information without verification.
b. ClearCase does not enforce the format of the contents of the comment field. This process relies upon the user to supply this information without verification.

Copyright © 1997, Floyd W. Shackelford. All Rights Reserved

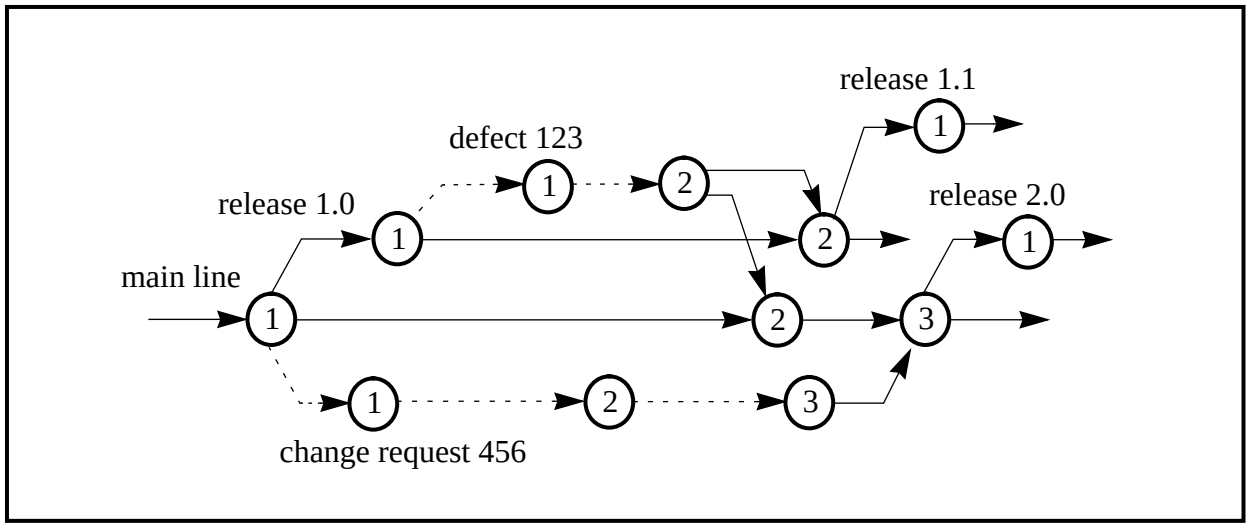
6. Configuration Managed Repository

The Configuration Managed Repository (“Repository”) is where all the Configuration Managed Items are stored. All changes to a Product will be made in that Product’s Repository. See the Procedures later in this section for details on how to manipulate the Repository.[3]

Our Repository Manager is ClearCase.

ClearCase supports parallel development via branching and merging. In our use, major releases will branch directly from the main line. Minor releases will branch from a major release line. Changes must be merged onto the main line or release line to participate in the next automatic build. This activity occurs only after the Configuration Managed Item has been built on the appropriate branch and thoroughly tested. Once merged to the main line or the release line, the new version will be labelled by the Configuration Management Team.

Figure 1. ClearCase Branching and Merging. *Circles represent build points. The numbers in the circles represent the build number for that branch and not ClearCase version numbers. Dotted lines represent change activity.*



6.1. Installing ClearCase

Contact the ClearCase Administrator to have ClearCase installed on your workstation.

3. Some commands in the following procedures begin with a “%”. The “%” represents your shell prompt and should not actually be keyed. These shell commands are in the Korn shell (ksh) format. You will need to use the syntax appropriate for your shell.

Copyright © 1997, Floyd W. Shackelford. All Rights Reserved

6.2. Creating the Repository

Contact the ClearCase Administrator to have a repository created for your product. You will need to provide the following information.

Table 4: ClearCase VOB Plan Questionaire

Product name
Number of configuration managed items in this product
Cumulative size of all configuration managed items in this product
COTS products which are used by this product
COTS products which are used to develop this product
Platforms on which this product is developed
Platforms to which this product is deployed

6.3. Initially Loading the Repository

If this is a new product, then there may be nothing to initially load into the repository. However, if this is an existing product which is not already in ClearCase, then the ClearCase repository will need to be initialized with the appropriate version of that product.

Generally, the Configuration Management Team will assist you with initially loading the Repository. If your product is stored in an SCCS, RCS, or CVS repository, then there are ClearCase tools which will migrate the product into ClearCase and preserve its change history. If your product is

Copyright © 1997, Floyd W. Shackelford. All Rights Reserved

in a “normal” file system, then the following script will load a copy of the entire product structure into ClearCase:

```
#! /bin/ksh

# You need to provide the following values . . .
PRODUCT_TOP=pathToProductTop ;
VOB_TOP=pathToVobTop ;

COMMENT=“Initial Load” ;

cd ${PRODUCT_TOP} ;

for CMITEM in ‘find . -name “*” -print‘
do
  if [ -d ${CMITEM} ]
  then
    cleartool mkdir -c ${COMMENT} ${VOB_TOP}/${CMITEM} ;
  else
    cleartool mkelem -elt file -c ${COMMENT} ${VOB_TOP}/${CMITEM} ;
    cleartool ci -c ${COMMENT} -from ${CMITEM} ${VOB_TOP}/${CMITEM} ;
    cleartool mklabel -c ${COMMENT} INITIAL_LOAD ${VOB_TOP}/${CMITEM};
  fi
done
```

6.4. Creating a Branch

To create a branch, ask the Configuration Management Team to create a config spec for the new branch. Once the config spec is created, you use the config spec when you want to reference that branch from within your View.

6.5. Creating your View

In ClearCase, Work Spaces are called Views. Every developer has a private View separate from every other deverloper. To get a View, ask the Configuration Management Team to create a View for you.

Your View may be deleted at any time. You will not impact the Repository. However, if you have any uncommitted changes in the View, they will be lost.

6.6. Adding a Configuration Managed Item to the Repository

This procedure describes how to add a new Configuration Managed Item to the Repository. This procedure assumes that you already have your View and config spec created. These steps may be performed by any Project Team member:

```
% cleartool co -c “reason”.
% cleartool mkelem -elt file -c “reason” fileName
% cleartool ci -c “reason” .
```

Copyright © 1997, Floyd W. Shackelford. All Rights Reserved

6.7. Checking out a Configuration Managed Item

This procedure describes how to check out a Configuration Managed Item from the Repository. This procedure assumes that you already have your View and config spec created.

```
% cleartool co -c "reason" fileName
```

6.8. Checking in a Configuration Managed Item

This procedure describes how to check out a Configuration Managed Item from the Repository. This procedure assumes that you already have your View and config spec created.

```
% cleartool ci -c "reason" fileName
```

Your View may be deleted at any time. You will not impact the Repository. However, if you have any uncommitted changes in the View, they will be lost.

6.9. Merging a Branch

This procedure describes how to merge a branch to another branch or the main line in the Repository. This procedure assumes that you have already made the changes you want to the Configuration Managed Items in your View and you have committed those changes to the Repository. This procedure will be performed by any Project Team member.

Table 5: Merging a Branch

Step	Activities
1. Set the merge-to branch view	`% cleartool setview viewName` `% cleartool setcs configSpec`
2. Merge the branch	1. Merge the merge-from branch into the merge-to branch: `% cleartool findmerge -ver /branchName/fileName` 2. Correct merge errors 3. Verify that the merged product is still viable

3. Check in the merged-to branch	1. Get a new revision number from the Configuration Management Team and schedule a time with them to commit this new revision to the Repository
	2. Label the View with the new revision number:
	`% cleartool mklabel -replace versionNumber`
	3. At the agreed upon time, commit your View to the Repository:
	`% for fn in 'cleartool lsco -r -s -me /vob/productName'`
	`do`
	`  cleartool ci -c "reason" ${fn}`
	`done`

Copyright © 1997, Floyd W. Shackelford. All Rights Reserved

7. Change Management

Once a Configuration Managed Item has been placed in the Repository, no changes may be made to the Configuration Managed Item without first having an approved reason for making the change. Typically, change is introduced by initiating some sort of “Change Request” and submitting it to a “Change Control Board”. Refer to the Software Development Life Cycle process for an explanation on where within the project life cycle each change control procedure is applicable.

This Change Management process recognizes four types of changes:
- Defects
- Problem Reports
- Change Requests
- System Development Requests

Defects. Defects are problems found with Configuration Managed Items for a Product which is still under development and has not yet been released to the customer. Defects are not generally written against items which are not under Configuration Management. Defects are usually generated as a result of an Inspection, a Test, or a Problem Report on a previously released version of the Product. In particular, a Defect is written when a product does not properly implement its specification.[4] Defects may result in the generation of follow-on Change Requests.[5]

Problem Reports. Problem Reports are problems found with released products and often result in changes to the project development products themselves (e.g. software or documentation). Problem Reports typically result in the generation of Defects against current development and/or Change Requests for inclusion in future development.

Change Requests. Change Requests are enhancements to system capabilities. They are generated to change the system’s requirements, which in turn should result in a changed system. Change Requests are typically applied to a Project after the Requirements Analysis has been completed and approved. In effect, the Change Request is a request to change the Requirements for the project or the Design of the product.[6]

System Development Requests. [7] System Development Requests are requests for a new product or to introduce *significant* change in a released product. They usually represent new business opportunities for the organization.

---

4. For instance: a design specification may not be in agreement with its requirements specification; a program module may not be written as designed; a test plan may not be properly testing the requirements.

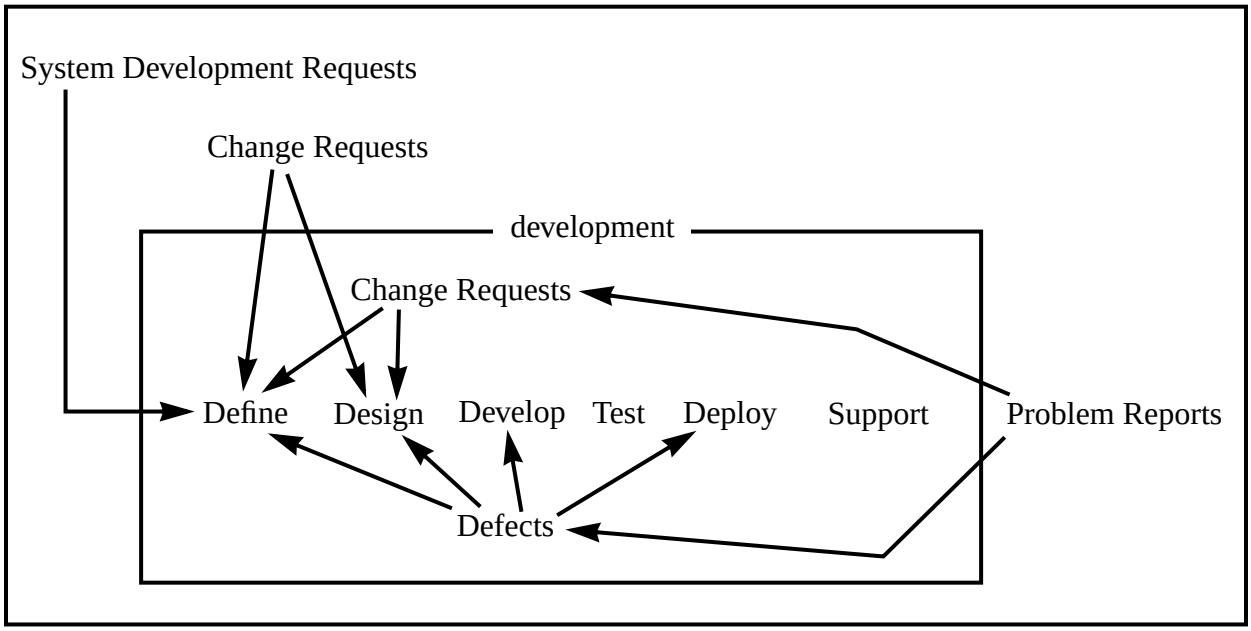
5. For instance, the Defect may specify a short-term work around or patch to be applied to the current System Development effort. The Change Request may then specify that a more strategically acceptable change will be implemented in the Product later on.

6. In this case, we discover that while the product implemented its specifications properly, so there is no Defect against the product, the result, for whatever reason, is not what we want. The business case may have changed or additional information may have become available which prompts us to need to “change course”. Therefore, we create a Change Request - there is no error, we just need to change the specification for what it is we are going to produce.

Copyright © 1997, Floyd W. Shackelford. All Rights Reserved

In essence, all four changes differ only in the scope of the change they are requesting and the point at which that type of change may be introduced. System Development Requests represent the largest, most encompassing changes and are submitted only at the very beginning of the Software Development Life Cycle. Defects, on the other hand, usually have the smallest impact and can be applied as late as the Deploy phase of the Software Development Life Cycle. Change Requests typically impact only the Requirements or the Design. Problem Reports are found in the field and result in not only an immediate fix, but may also result in one or more Defects or Change Requests against any current development activities. The following diagram demonstrates where changes originate and where those changes might be applied.[8]

Figure 2. Change Management - Sources and Targets. *There are four ways to initiate change to the System: System Development Requests, Change Requests, Problem Reports, and Defects.*



7. Technically, System Development Requests do not necessarily cause a change to an existing Product, although it may. However, once the work effort produces a Configuration Managed Item (and it will), that Configuration Managed Item is subject to Change Management just like every other Configuration Managed Item. Both the initial creation of and subsequent changes to the Configuration Managed Item must be justified. Thus, the System Development Request provides the justification for the initial creation of the Configuration Managed Entiry and for possibly some of its changes during the course of the system development life cycle.

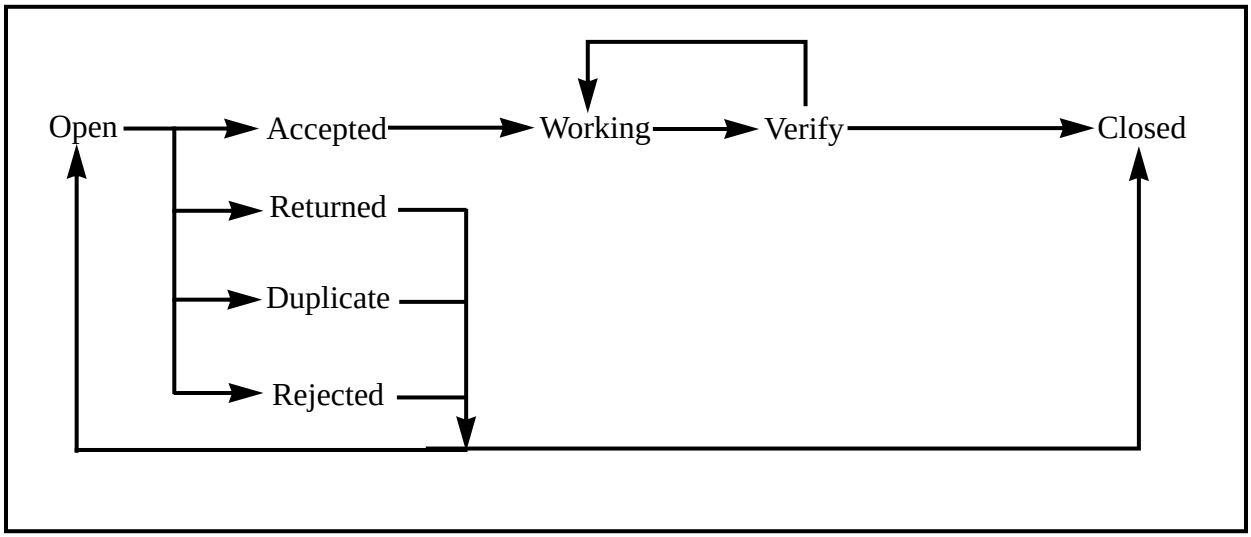
8. In this diagram, the Development box represents the development organization. Items within the box represent changes which originate within development and the activities in development to which those changes may be applied. For instance, Change Requests and Defects may originate from someone within the development organization. Items outside the box represent changes which originate from organizations outside of development which desire to change the system development activities within development. For instance, Problem Reports originate from users outside the development organization.

Copyright © 1997, Floyd W. Shackelford. All Rights Reserved

7.1. Defects

A Defect progresses through a series of states before it it finally closed. Graphically, it may be represented as shown in “Figure 3. Defect - State Transition Diagram” on page 16.

Figure 3. Defect - State Transition Diagram.



This procedure describes how a Defect progresses through its different states as depicted in the previous state diagram.

Table 6: Defect Procedure

Step	Roles	Activities	Deliverables
1.	Submitter	1. Discovers an error in a Configuration Managed Item 2. Fills out a Defect 3. Delivers the Defect to the Configuration Managed Item’s owner	Defect
2.	Owner	1. Reviews the Defect with the Submitter 2. Dispositions the Defect: • Return[a] • Reject[b] • Duplicate[c] • Accept[d]	Dispositioned Defect
3.	• Owner • Submitter	Agrees upon a fix	Fix identified

Copyright © 1997, Floyd W. Shackelford. All Rights Reserved

Table 6: Defect Procedure

Step	Roles	Activities	Deliverables
4.	Owner	1. Checks out the Configuration Managed Item 2. Applies the fix to the Configuration Managed Item	Fix applied
5.	Submitter	1. Verifies that the Configuration Managed Item has been fixed 2. Checks in the Configuration Managed Item 3. Closes the Defect	• Fix verified • Defect closed

a. The Configuration Managed Item Owner has determined that the Defect does not adequately describe the problem. If the Submitter agrees with this assessment, then the Submitter may either add additional information to the Defect and re-submit it or the Submitter may close the Defect.
b. The Configuration Managed Item Owner does not agree that the problem described by the Defect is an error in the Configuration Managed Item. If the Submitter agrees with this assessment, then the Submitter should close the Defect. If the Submitter disagrees with this assessment, then the Submitter should re-submit the Defect and escalate the issue to the appropriate level of management if necessary.
c. The Configuration Managed Item Owner has determined that the Defect is a duplicate of another Defect, a Problem Report, or is already incorporated in the work associated with an active Change Request. If the Submitter agrees with this assessment, then the Submitter should close the Defect. If the Submitter disagrees with this assessment, then the Submitter should re-submit the Defect and escalate the issue to the appropriate level of management if necessary.
d. The Configuration Managed Item Owner agrees that the problem described by the Defect is an error in the Configuration Managed Item and further agrees to fix the problem.

A Defect tracks certain information about the error. Here is a list of the fields which are tracked by the Defect.

Table 7: Defect Fields

Field	Description
Defect Identification	This is the identifier assigned to the Defect to uniquely distinguish it from all other Defect within development. Defect Identifiers are in the form: D*nnnnn* where: *nnnnn* is a unique sequential number
Current Status	This is the latest status

Copyright © 1997, Floyd W. Shackelford. All Rights Reserved

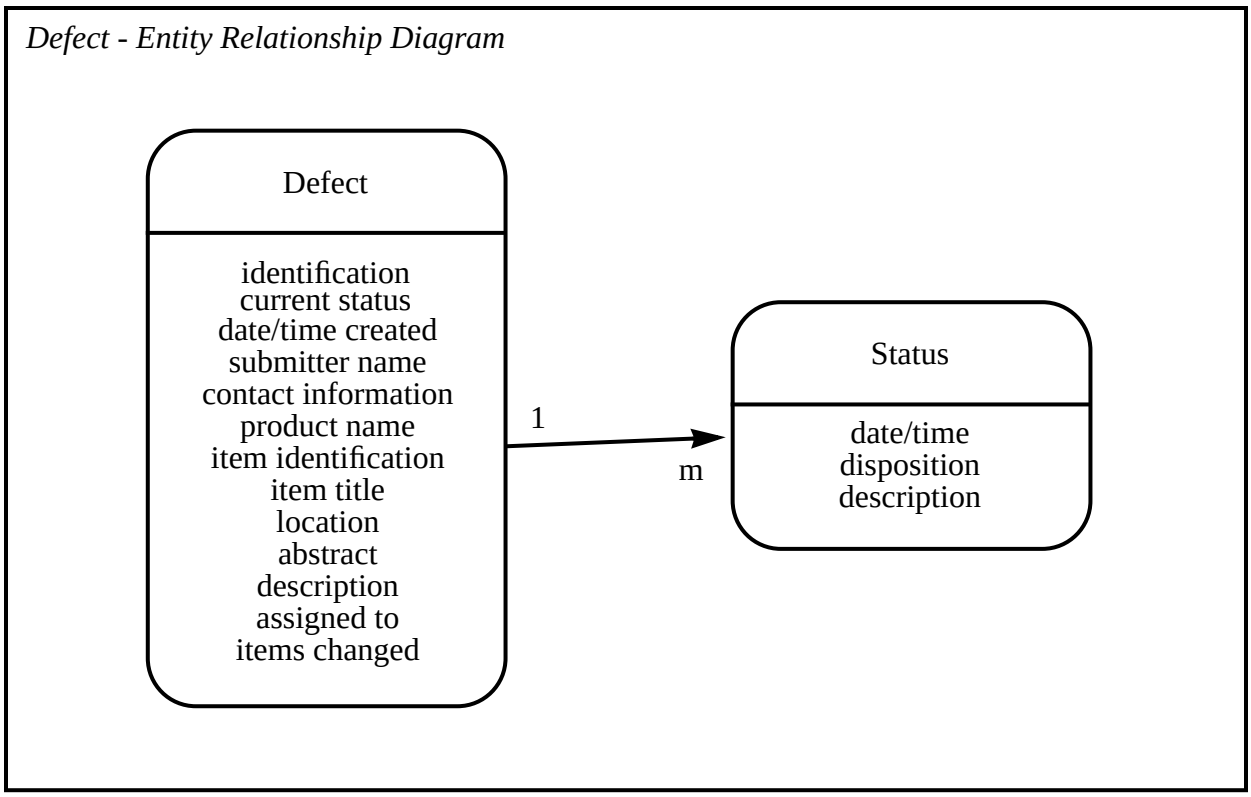
Table 7: Defect Fields

Field	Description
Date/Time created	This is the date and time which this Defect was first created
Submitter name	This is the full name of the person who submitted the Defect
Contact information	This is all the information necessary to make it easy for people to contact the Submitter regarding this Defect
Product name	The name of the Product in which this Defect was found
Item identification	This is the the identifier of the Configuration Managed Item in which this Defect was found
Item title	This is the title of the Configuration Managed Item
Location	This is where the Defect is located within the Configuration Managed Item[a]
Abstract	Give a brief description of the Defect
Detailed description	Give as much detail about the Defect as possible. This section can easily consist of many pages of material.
Assigned to	This is the full name of the Producer assigned to fix the Defect
Status:	Create a new status record every time the disposition changes
Status: date/time	The date and time of this status change
Status: disposition	The new disposition: • Return • Reject • Duplicate • Accept • Working • Verify • Close
Status: description	Justify the new disposition
Items changed	This is a list of all the Configuration Managed Items modified to fix this defect

a. For example: page number, line number, paragraph number, etc.

Copyright © 1997, Floyd W. Shackelford. All Rights Reserved

Conceptually, the Defect represents a relational organization. Here are the same fields in relational form:



7.2. Problem Reports

In this process, we use terminology not found in other parts of this document to specify certain roles particular to the Problem Report process.

User. The User is the person who is using the product and discovers the Problem.

Help Desk. The Help Desk is the primary point of contact for all Users. Users call the Help Desk to report all Problems.

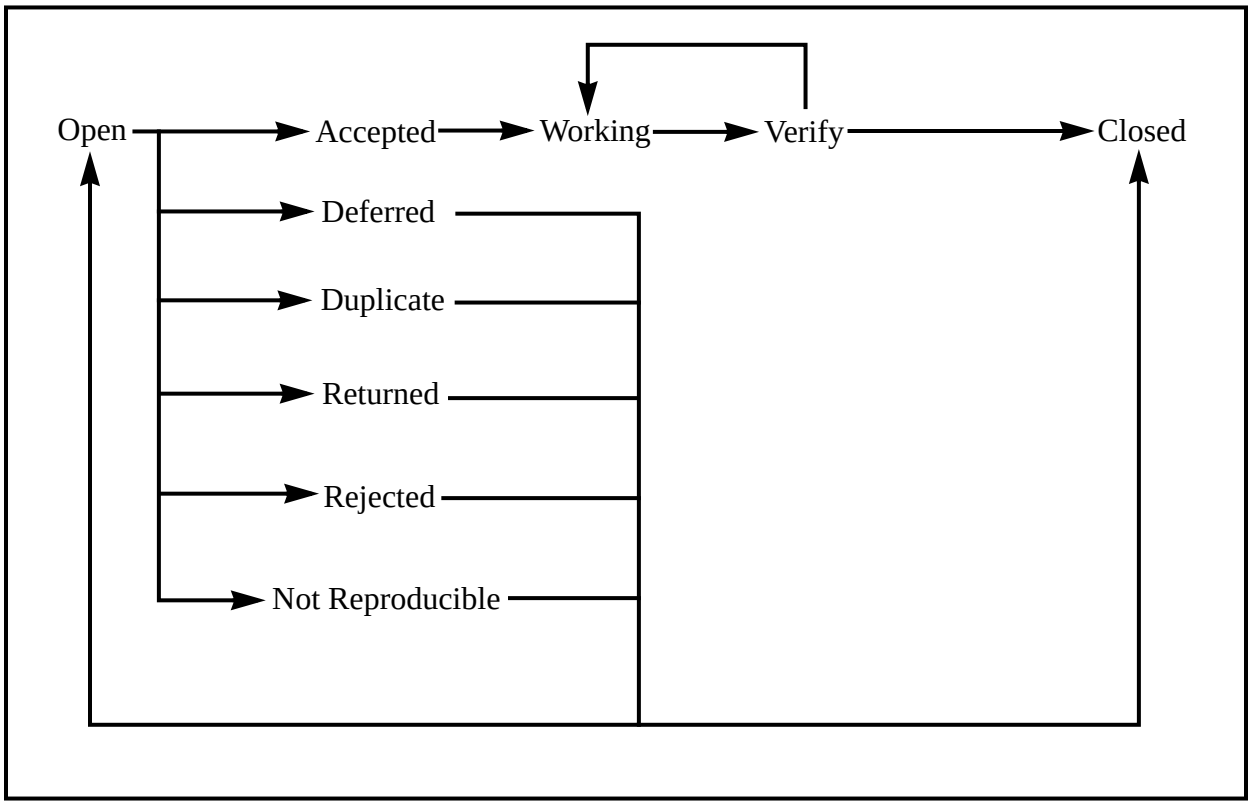
Product Support Specialist. A Product Support Specialist is someone who is knowledgeable in a particular Product and is able to service all Problem Reports which do not actually require a programming modification to fix.

Developer. A Developer is the person who is able to modify the Product's software.

A Problem Report progresses through a series of states before it it finally closed as shown in "Figure 4. Problem Report - State Transition Diagram" on page 20.

Copyright © 1997, Floyd W. Shackelford. All Rights Reserved

Figure 4. Problem Report - State Transition Diagram.



This procedure describes how a Problem Report progresses through its different states as depicted in the previous state diagram.

Table 8: Problem Report Procedure

Step	Roles	Activities	Deliverables
1.	User	1. Finds a problem in the product 2. Creates a Problem Report 3. Forwards the Problem Report to the Help Desk	
2.	Help Desk	1. Discusses the reported problem with the User and captures as much detail as possible 2. Routes the Problem Report to the appropriate Product Support Specialist	

Copyright © 1997, Floyd W. Shackelford. All Rights Reserved

Table 8: Problem Report Procedure

Step	Roles	Activities	Deliverables
3.	Product Support Specialist	1. Works with the user to work around the problem 2. Determines the severity of the Problem Report with the User 3. Adds additional details to the Problem Report 4. Forwards the Problem Report to the Problem Review Board	
4.	Problem Review Board	1. The Problem Review Board meets regularly and on an *ad hoc* basis as needed. 2. Reviews the Problem Report and determines the appropriate response: • Return[a] • Reject[b] • Defer[c] • Duplicate[d] • Accept[e] 3. Forwards the Accepted Problem Report to the appropriate Developer for action	

Copyright © 1997, Floyd W. Shackelford. All Rights Reserved

Table 8: Problem Report Procedure

Step	Roles	Activities	Deliverables
5.	Developer	1. Reviews the Problem Report and talks to the Product Support Specialist, the Help Desk, and the User as appropriate to gather additional information as required. 2. Creates a branch off of the main line of the Product's Repository[f] [g] 3. Recreates the problem[h] 4. Debugs the problem 5. Determines an estimation of when the fix can be ready for integration into the main line of the Product's Repository	
6.	Help Desk	Informs the user when the fix is expected to be released	
7.	Developer	1. Develops a solution to the problem[i] [j] 2. Causes the solution to be Inspected 3. Fixes Inspection Defects 4. Integrates the solution into the main line of the Product's Repository[k] 5. Forwards the Problem Report to the Build and Integration Team for inclusion in the next Product Release	

a. The Problem Report may be returned to the User or the Product Support Specialist for more information.
b. The Problem Report may not be a problem at all, or the Problem Report may be better classified as a functional enhancement. The Problem Report may be rejected with a suggestion that the user submit either a Change Request or a System Development Request. If the Submitter agrees with this assessment, then the Submitter should close the Problem Report. If the Submitter disagrees with this assessment, then the Submitter should re-submit the Problem Report and escalate the issue to the appropriate level of management if necessary.

Copyright © 1997, Floyd W. Shackelford. All Rights Reserved

c. The Problem Report may in fact be a true problem, but the Problem Review Board chooses to not fix the problem in the current release but instead fix the problem in some later release of the Product. If the Submitter agrees with this assessment, then the Submitter should close the Problem Report. If the Submitter disagrees with this assessment, then the Submitter should re-submit the Problem Report and escalate the issue to the appropriate level of management if necessary.
d. The Problem Report may be a duplicate of an existing Problem Report. In this case, the Problem Report should be returned to the Submitter, referencing the other Problem Report. If the Submitter agrees with this assessment, then the Submitter should close the Problem Report. If the Submitter disagrees with this assessment, then the Submitter should re-submit the Problem Report and escalate the issue to the appropriate level of management if necessary.
e. The Problem Report describes a problem with the product and the Problem Review Board chooses to fix the problem in the current release.
f. See “Creating a Branch” on page 12.
g. See “Checking out a Configuration Managed Item” on page 13.
h. If, after all reasonable attempts to recreate the problem, the problem cannot be recreated, then return the Problem Report to the User as “non reproducible”. If the Submitter agrees with this assessment, then the Submitter should close the Problem Report. If the Submitter disagrees with this assessment, then the Submitter should re-submit the Problem Report, demonstrate that the problem is reproducible, and escalate the issue to the appropriate level of management if necessary.
i. See “Building a Branch” on page 39.
j. See “Checking in a Configuration Managed Item” on page 13.
k. See “Merging a Branch” on page 13.

A Problem Report tracks certain information about the error. Here is a list of the fields which are tracked by the Problem Report:

Table 9: Problem Report Fields

Field	Description
Problem Report Identification	This is the identifier assigned to the Problem Report to uniquely distinguish it from all other Problem Report within development. Problem Report Identifiers are in the form: PR*nnnnn* where: *nnnnn* is a unique sequential number assigned by the Help Desk
Current Status	This is the latest status
Date/Time received by Help Desk	This is the date and time which this Problem Report was first reported to the Help Desk
Reported by	This is the full name of the person who reported the Problem to the Help Desk

Copyright © 1997, Floyd W. Shackelford. All Rights Reserved

Table 9: Problem Report Fields

Field	Description
Severity	1. • stops critical customer functions • no work around available • work around must be made available within 24 hours[a] 2. • impacts critical customer functions • work around available • fix strategy must be made available within 24 hours • fix must be made available in the next release 3. • customer can perform critical functions • no work around necessary • function does not work as documented • fix will be made available in the next release 4. minor correction[b]
Contact information	This is all the information necessary to make it easy for people to contact the Submitter regarding this Problem Report
Abstract	Give a brief description of the Problem
Description	Give as much detail about the Problem as possible. This section can easily consist of many pages of material.
Help Desk Attendant's name	This is the full name of the Help Desk Attendant who created the Problem Report
Steps to recreate the problem	These are the steps necessary to recreating the Problem
Date/Time forwarded to Product Specialist	This is the date and time at which the Problem Report was forwarded to a Product Specialist for resolution
Product Specialist name	This is the full name of the Product Specialist assigned to investigate the problem
Date/Time forwarded to PR Review Board	This is the date and time which this Problem Report was forwarded to the Problem Review Board
PR Review Board reviewers	This is a list of the names of the Problem Review Board members who reviewed this Problem Report

Copyright © 1997, Floyd W. Shackelford. All Rights Reserved

Table 9: Problem Report Fields

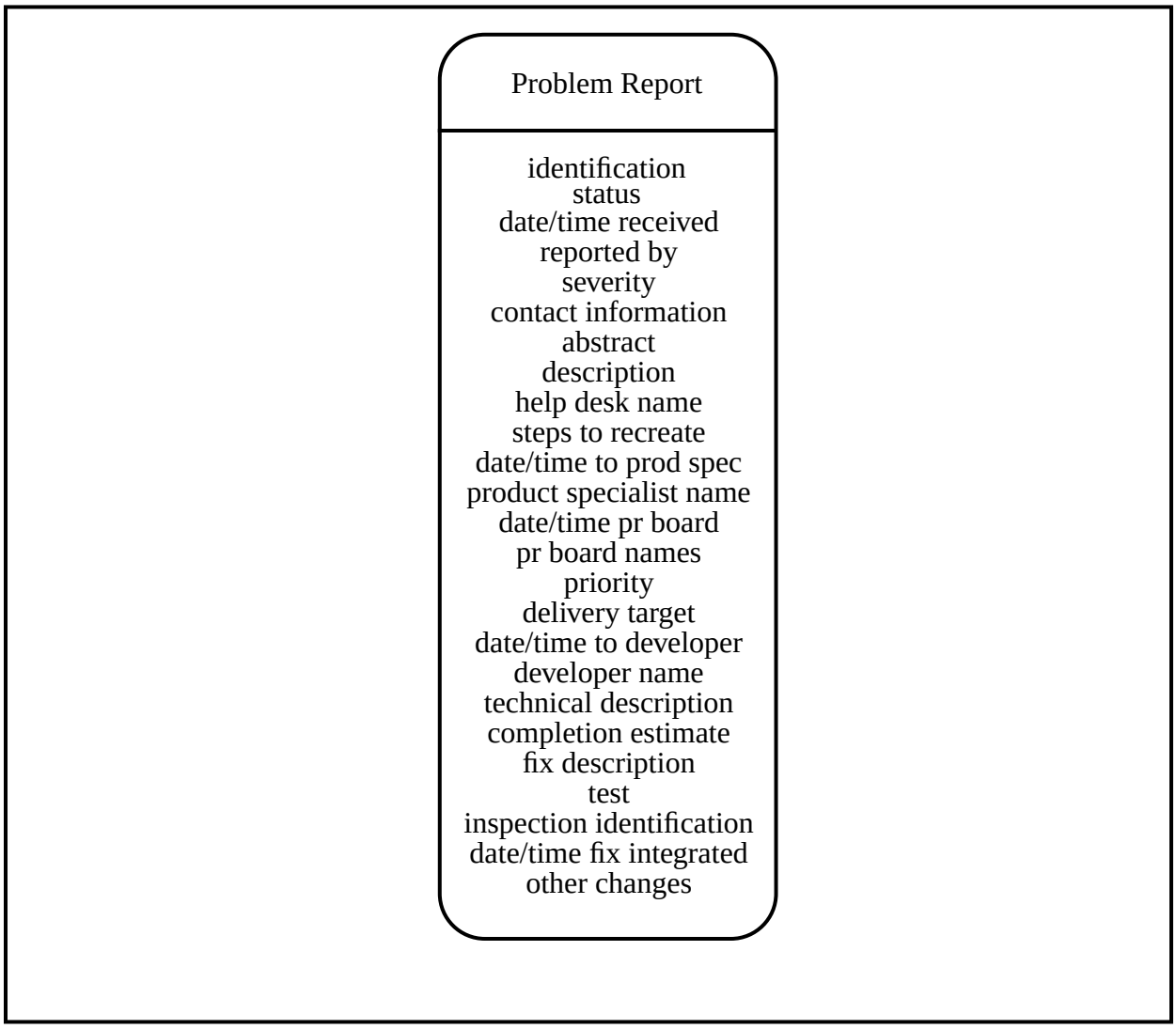
Field	Description
Priority	This is the priority assigned to the Problem Report by the Problem Review Board: 1. Fix and incorporate into a regularly scheduled build as soon as possible 2. Fix and incorporate into the next regularly scheduled release
Delivery target	When the Problem Review Board expects to have the fix delivered (e.g. date, release)
Date/Time forwarded to Developer	This is the date and time which this Problem Report was given to the Developer for fix
Developer name	This is the full name of the Developer
Technical description	This is a technical description of the problem
Completion estimate	This is an estimate of when the Developer thinks the fix can be ready for delivery
Description of fix	This is a technical description of the fix
Tests used to validate the fix	This is a description of the tests used by the Developer to verify that the fix: 1. fixed the problem 2. did not introduce additional errors
Inspection ID	Every fix must be inspected before it can be incorporated into the main line.
Date/Time fix integrated with main line	This is the date and time which this fix was incorporated into the main line
Resulting Change Requests and Defects	List all the Defects and Change Requests which were generated to implement a fix in follow-on releases.

a. At which time severity changes to 2.
b. For example: typographical errors, etc.

Conceptually, the Problem Report represents a relational organization. Here are the same fields in relational form

Copyright © 1997, Floyd W. Shackelford. All Rights Reserved

Figure 5. Problem Report - Entity Relationship Diagram.

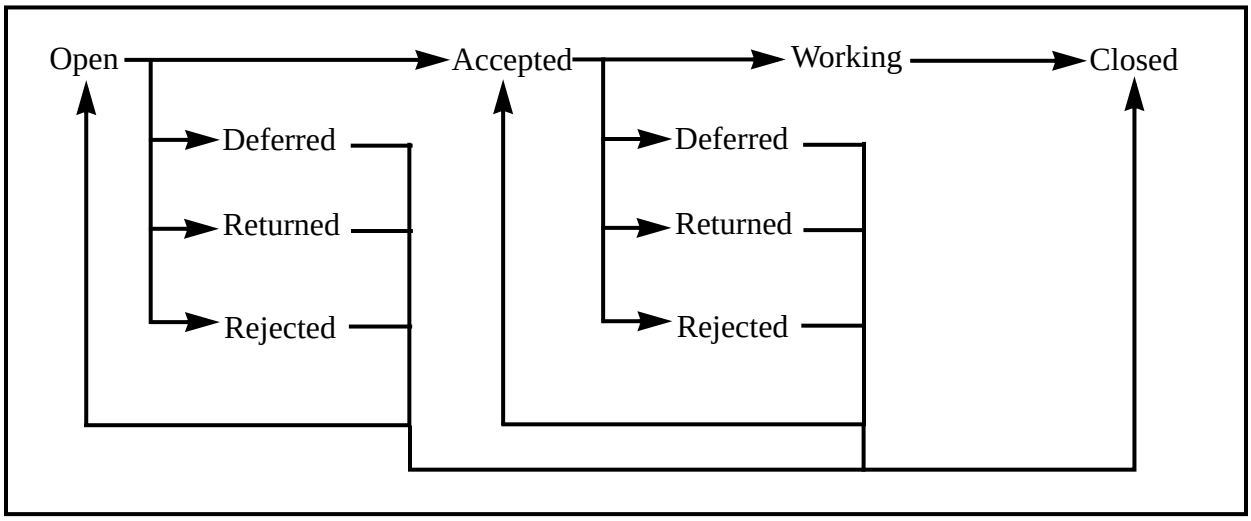


7.3. Change Requests

A Change Request progresses through a series of states before it is finally closed. Graphically, these state transitions may be diagrammed as shown in “Figure 6. Change Request - State Transition Diagram” on page 27.

Copyright © 1997, Floyd W. Shackelford. All Rights Reserved

Figure 6. Change Request - State Transition Diagram.



This procedure describes how a Change Request progresses through its different states as depicted in the previous state transition diagram.

Table 10: Change Request Procedure

Step	Roles	Activities	Deliverables
1	Submitter	1. Fills out a Change Request form 2. Submits the Change Request form to the Change Control Board	Change Request created
2	Change Control Board	1. Meets on a regular basis to review open Change Requests 2. Reviews the open Change Request and dispositions it: • Return[a] • Reject[b] • Defer[c] • Accept[d] 3. Appoints a Systems Analyst to process the Change Request	Initial disposition of Change Request registered

Copyright © 1997, Floyd W. Shackelford. All Rights Reserved

Table 10: Change Request Procedure

Step	Roles	Activities	Deliverables
3	Systems Analyst	1. Completes the requirements analysis, impact assessment, and dependencies sections of the Change Request 2. Presents the updated Change Request to the Change Control Board	Change Request technical and planning details completed
4	Change Control Board	Reviews the Accepted Change Request and dispositions it: • Return[e] • Reject[f] • Defer[g] • Working[h]	Change Request ready for inclusion in the Product
5	Product Team	Incorporates Change Request into project deliverables.	Requested change incorporated into the Product

a. The Change Control Board has determined that the Change Request has insufficient information to understand the change being requested. The Change Request is returned for more information.
b. The Change Control Board has determined that the Change Request is outside the scope of the Product targeted for the change. If the Submitter agrees with this assessment, then the Submitter should close the Change Request. If the Submitter disagrees with this assessment, then the Submitter should re-submit the Change Request and escalate the issue to the appropriate level of management if necessary.
c. The Change Control Board has determined that the Change Request is appropriate for the targeted Product and that the change should be included in the Product. However, the Change Control Board has also determined that the change should not be included in the release specified by the Change Request, but that the change should instead be deferred to a subsequent release. If the Submitter agrees with this assessment, then the Submitter should close the Change Request. If the Submitter disagrees with this assessment, then the Submitter should re-submit the Change Request and escalate the issue to the appropriate level of management if necessary.
d. The Change Control Board has determined that the Change Request should be included in the next release of the Product.
e. The Change Control Board has determined that the Change Request has insufficient information to evaluate the full extent of the change being requested. The Change Request is returned for more analysis.
f. The Change Control Board has determined that the Change Request is outside the scope of the Product targeted for the change. If the Submitter agrees with this assessment, then the Submitter should close the Change Request. If the Submitter disagrees with this assessment, then the Submitter should re-submit the Change Request and escalate the issue to the appropriate level of management if necessary.
g. The Change Control Board has determined that the Change Request is appropriate for the targeted Product and that the change should be included in the Product. However, the Change Control Board has also determined that the change should not be included in the release specified by the Change Request, but that the change should instead be deferred to a subsequent release. If the Submitter agrees with this assessment, then the Submitter should close the Change Request. If the Submitter disagrees with this assessment, then the Submitter should re-submit the Change Request and escalate the issue to the appropriate level of management if necessary.

Copyright © 1997, Floyd W. Shackelford. All Rights Reserved

h. The Change Control Board has determined that work may proceed with including the requested change in the next release of the Product.

A Change Request tracks certain information about the desired change. Here is a list of the fields which are tracked by the Change Request:

Table 11: Change Request Fields

Field	Description
Change Request Identification	This is the identifier assigned to the Change Request to uniquely distinguish it from all other Change Requests within development. Change Request Identifiers are in the form: CR*nnnnn* where: *nnnnn* is a unique sequential number assigned by the Configuration Management Team
Current Status	This is the latest status from either the Initial Review or the Detailed Review
Date/Time created	This is the date and time which this Change Request was first created
Submitter names	This is the full name of the person who submitted the Change Request
Contact information	This is all the information necessary to make it easy for people to contact the Submitter regarding this Change Request
Priority	1. critical functionality[a] 2. necessary functionality[b] 3. desirable functionality[c]
Products to be changed	List the names of the Products which will be changed by this Change Request
Type of change	• usability[d] • functionality[e] • performance[f] • platform[g] • integration[h]
Abstract	Give a brief description of the change
Description of the change	Give as much detail about the change as possible. This section can easily consist of many pages of material.

Copyright © 1997, Floyd W. Shackelford. All Rights Reserved

Table 11: Change Request Fields

Field	Description
Justification for the change	Discuss why this change should be made
CCB Initial Review:	Make an Initial Review entry for each time the Change Control Board conducts an Initial Review of this Change Request
CCB Initial Review: Date/Time submitted	The date and time the Change Request was submitted to the Change Control Board for an Initial Review
CCB Initial Review: Date/Time reviewed	The date and time the Change Control Board reviewed the Change Request
CCB Initial Review: Reviewed by names	The list of Change Control Board members who reviewed the Change Request
CCB Initial Review: Disposition	Record the results of the review in terms of a new disposition: • Return • Reject • Defer • Accept
CCB Initial Review: Details	Justify the new disposition
Systems Analyst name	This is the full name of the Systems Analyst assigned to research the Change Request
Requirements Analysis	This section contains the Requirements Analysis performed by the Systems Analyst
Impacts:	Make an Impact entry for each product which will need to be changed as a result of this Change Request
Impacts: Product	The name of the Product for which this Impact entry is created
Impacts: Description of changes	The technical changes which must be made to the Product to accommodate this Change Request
Impacts: Budget	The changes which must be made to the budget to accommodate this Change Request
Impacts: Schedule	The changes which must be made to the schedule to accommodate this Change Request
Impacts: Resources	The changes which must be made to the resource allocations to accommodate this Change Request

Copyright © 1997, Floyd W. Shackelford. All Rights Reserved

Table 11: Change Request Fields

Field	Description
Impacts: Dependencies	Any dependencies which which will be created, modified, or severed as a result of this Change Request
CCB Detailed Review:	Make a Detailed Review entry for each time the Change Control Board conducts a Detailed Review of this Change Request
CCB Detailed Review: Date/Time submitted	The date and time the Change Request was submitted to the Change Control Board for a Detailed Review
CCB Detailed Review: Date/Time reviewed	The date and time the Change Control Board reviewed the Change Request
CCB Detailed Review: Reviewed by names	The list of the Change Control Board members who reviewed the Change Request
CCB Detailed Review: Disposition	Record the results of the review in terms of a new disposition: • Return • Reject • Defer • Working
CCB Detailed Review: Details	Justify the new disposition
Target Release	Identify the release to which the Change Request is to be applied[i]
Implementation assigned to name	Identify the Project Manager who is responsible for implementing the Change Request[j]
Items Changed	This is a list of all the Configuration Managed Items changed to implement this Change Request

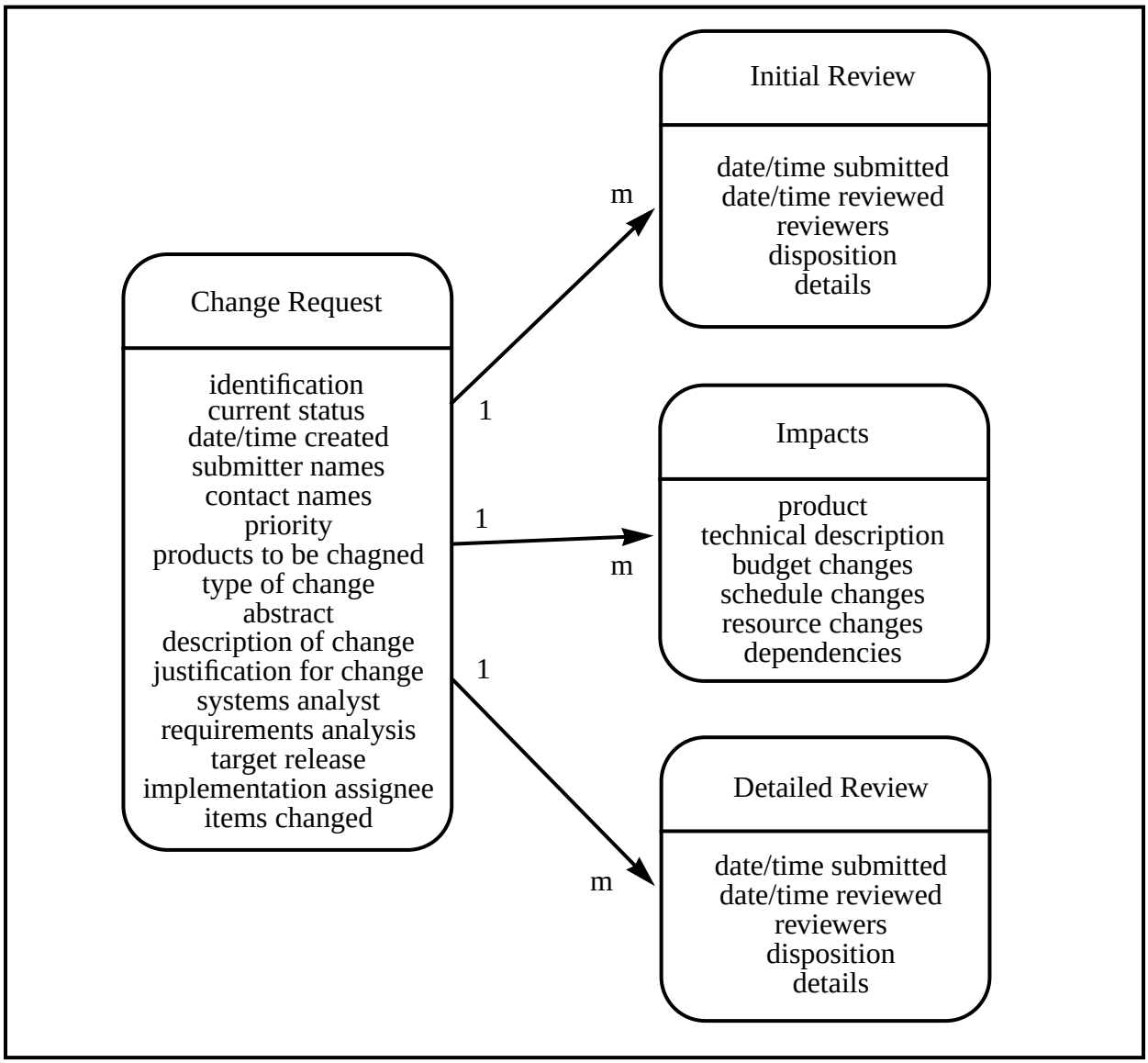
a. Critical Functionality is functionality which is necessary for the User to be able to conduct business. Lack of the functionality makes it extrememly difficult or impossible for the user to be able to perform critical business functions. Alternatively, projections show that this functionality has a tremendous return on investment or a significant impact on market share.
b. Necessary Functionality is functionality which is necessary for the User to be able to conduct business. Lack of the functionality means that the customer will have to continue performing the business function in a manual manner. Thus, the lack of this functionality means that the system's ability to enhance the User's productivity is greatly impaired. Alternatively, projections show that this functionality has a desirable return on investment.
c. Desirable Functionality is functionality which would make the User's job more efficient or eliminate or combine certain steps in the way the User performs the business function.
d. Usability means that this change is requested primarily to enhance the usability of the system.
e. Functionality means that this change is requested primarily to add or modify functionality.
f. Performance means that this change is requested primarily to improve the performance characteristics of the system

Copyright © 1997, Floyd W. Shackelford. All Rights Reserved

g. Platform means that this change is requested primarily to port the system to another platform, whether hardware or software or both.
h. Integration means that this change is requested primarily to integrate the system with another system.
i. If this Change Request impacts multiple Products, then list each Product and its corresponding release number.
j. If this Change Request impacts multiple Products, then list each Product and its corresponding Project Manager.

Conceptually, the Change Request represents a relational organization. Here are the same fields in relational form.

Figure 7. Change Request - Entity Relationship Diagram.

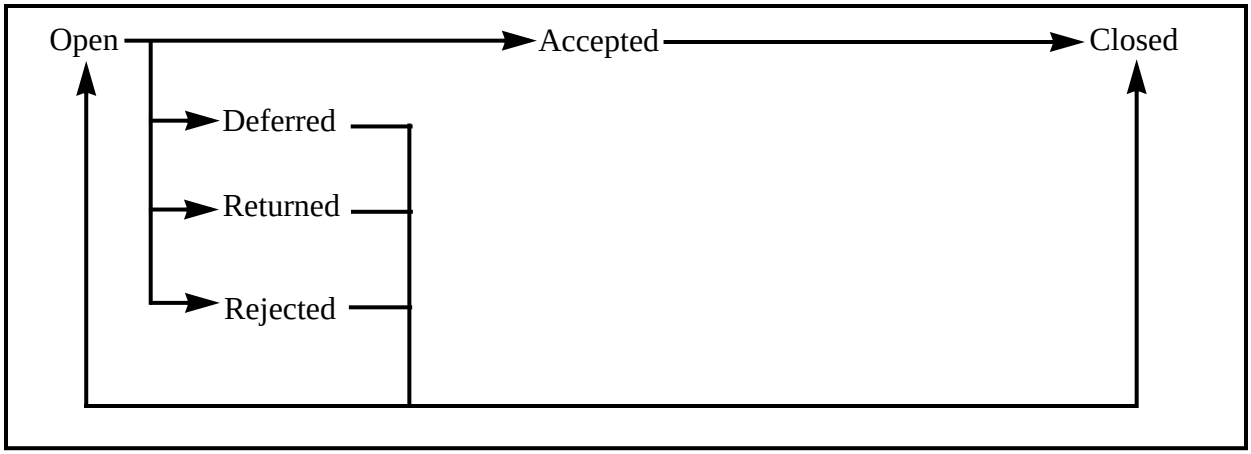


Copyright © 1997, Floyd W. Shackelford. All Rights Reserved

7.4. System Development Requests

A System Development Request progresses through a series of states before it is finally closed. Graphically, these state transitions may be diagrammed as shown in “Figure 8. System Development Request - State Transition Diagram” on page 33.

Figure 8. System Development Request - State Transition Diagram.



This procedure describes how a System Development Request progresses through its different states as depicted in the previous state transition diagram.

Table 12: System Development Request Procedure

Step	Roles	Activities	Deliverables
1.	Submitter	1. Determines that a new Product needs to be produced or that a *significant* change needs to be made to an existing Product 2. Completes the System Development Request form 3. Submits the System Development Request form to the Software Development Steering Committee	• Completed System Development Request submitted to the Software Development Steering Team • Software Development Life Cycle begins

Copyright © 1997, Floyd W. Shackelford. All Rights Reserved

Table 12: System Development Request Procedure

Step	Roles	Activities	Deliverables
2.	Software Development Steering Committee	1. Reviews the Software Development Request and dispositions it: • Return[a] • Rejected[b] • Deferred[c] • Accepted[d] 2. Assigns the Software Development Request to a Systems Analyst[e]	Define phase begins

a. The Software Development Steering Committee has determined that the System Development Request has insufficient information to understand the change being requested. The System Development Request is returned for more information.

b. The Software Development Steering Committee has determined that the System Development Request is beyond the business objectives of the development organization. If the Submitter agrees with this assessment, then the Submitter should close the System Development Request. If the Submitter disagrees with this assessment, then the Submitter should re-submit the System Development Request and escalate the issue to the appropriate level of management if necessary.

c. The Software Development Steering Committee has determined that the System Development Request is within the business objectives of the development organization. However, the Change Control Board has also determined that the Product should not be developed in the development business cycle specified by the System Development Request, but that the project should instead be deferred to a subsequent development business cycle. If the Submitter agrees with this assessment, then the Submitter should close the System Development Request. If the Submitter disagrees with this assessment, then the Submitter should re-submit the System Development Request and escalate the issue to the appropriate level of management if necessary.

d. The Software Development Steering Committee has determined that the requested Product should be included in the current development business cycle.

e. Continue with the Software Development Life Cycle process.

Copyright © 1997, Floyd W. Shackelford. All Rights Reserved

A System Development Request tracks certain information about the desired change. Here is a list of the fields which are tracked by the System Development Request:

Table 13: System Development Request Fields

Field	Description
System Development Request Identification	This is the identifier assigned to the System Development Request to uniquely distinguish it from all other System Development Requests within development. System Development Request Identifiers are in the form: SDR*nnnnn* where: *nnnnn* is a unique sequential number assigned by the Configuration Management Team
Current Status	This is the latest disposition assigned to this System Development Request
Date/Time created	This is the date and time which this System Development Request was first created
Submitter names	This is the full name of the person who submitted the System Development Request
Contact names	This is all the information necessary to make it easy for people to contact the Submitter regarding this System Development Request
Priority	1. critical functionality[a] 2. necessary functionality[b] 3. nice functionality[c]
Abstract	Give a brief description of the Product
Description	Give as much detail about the change as possible. This section can easily consist of many pages of material.
Justification	Discuss why this Product should be developed.

a. Critical Functionality is functionality which is necessary for the User to be able to conduct business. Lack of the functionality makes it extrememly difficult or impossible for the user to be able to perform critical business functions. Alternatively, projections show that this functionality has a tremendous return on investment or a significant impact on market share.

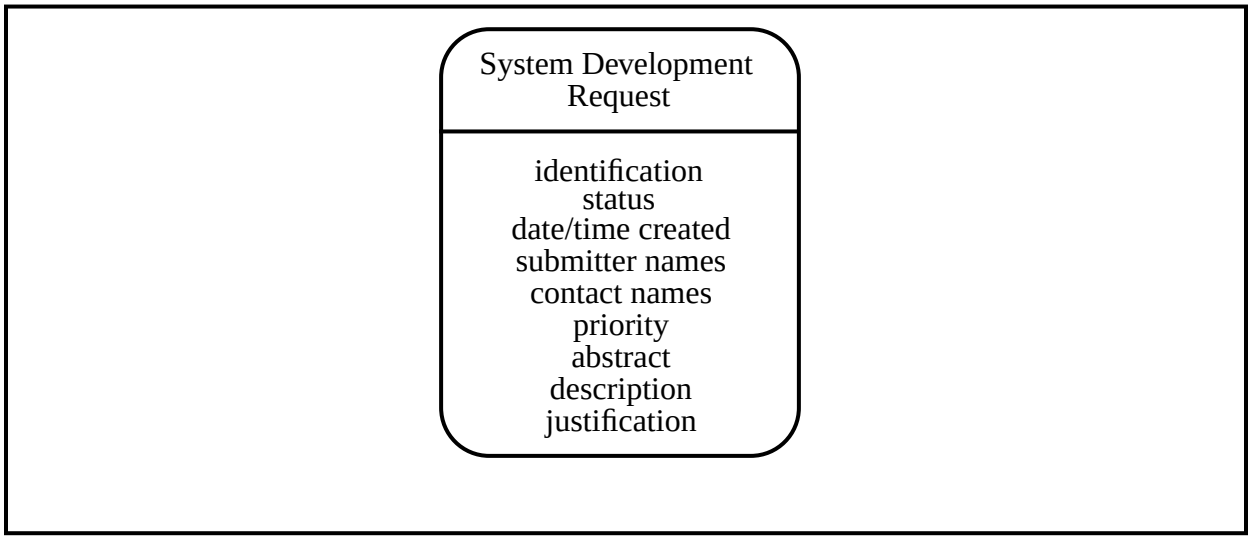
b. Necessary Functionality is functionality which is necessary for the User to be able to conduct business. Lack of the functionality means that the customer will have to continue performing the business function in a manual manner. Thus, the lack of this functionality means that the system's ability to enhance the User's productivity is greatly impaired. Alternatively, projections show that this functionality has a desirable return on investment.

c. Desirable Functionality is functionality which would make the User's job more efficient or eliminate or combine certain steps in the way the User performs the business function.

Copyright © 1997, Floyd W. Shackelford. All Rights Reserved

Conceptually, the System Development Request represents a relational organization. Here are the same fields in relational form

Figure 9. System Development Request - Entity Relationship Diagram.



Copyright © 1997, Floyd W. Shackelford. All Rights Reserved

8. Version Management

This section describes how we identify new versions of a Product.

Whenever a Configuration Managed Item is checked out, changed, and subsequently checked in, a new version number is automatically assigned to that new version of the individual Configuration Managed Item. We define a version of a Product as the set of all the Configuration Managed Items in the Product at particular version levels. Product versioning requires someone to decide which versions of its component Configuration Managed Items should be incorporated in the new version of the Product. Therefore, unlike version numbers for individual Configuration Managed Items, Product version numbers are manually generated and assigned by the Configuration Management Team.

Release Number. The Release Number is a sequential number which identifies significant versions of the Product which will be released to the Customer. Assigning Release Numbers is a policy decision which relies mostly upon subjective factors. The Release Number for a Product begins with 1 (one) for the first version of the Product to be released to the Customer and is sequentially incremented when it is decided that a sufficiently different version of the Product is to be developed.

Revision Number. The Revision Number is a sequential number which identifies versions of the Product which have been released for general use within the development organization. When a branch is committed to the main line, its component Configuration Managed Items are considered released to the development organization. The Revision Number for a Product begins with 1 (one) for the first time the Product is committed to the main line of a particular Repository and is sequentially incremented thereafter.

Version Number. The Version Number uniquely identifies the configuration of the Product at any single point in time. The Version Number is the value we use to label Configuration Managed Items in the Repository. A Version Number consists of a Release Number and a Revision Number and is usually constructed as follows:

v*n.m*

where:

- *n* is the Release Number
- *m* is the Revision Number

By way of example, suppose that we are working on Release 1 of Product XYZ. Suppose too that we have three Configuration Managed Items in the Product called ItemA, ItemB, and ItemC. Again, suppose that these Configuration Managed Items have the versions in the Product XYZ Release 1 Repository as shown in “Figure 10. Configuration Managed Item Versions.” on page 38.

Say we decide that Revision 1 of Release 1 of the System should consist of version 2.1 of ItemA, version 1.4 of ItemB, and version 1.1 of ItemC. Therefore, Product XYZ v1.1 consists of the selected Configuration Managed Items as shown in “Figure 11. System Version.” on page 38.

Copyright © 1997, Floyd W. Shackelford. All Rights Reserved

Figure 10. Configuration Managed Item Versions. *Each Configuration Managed Item has one or more versions within the repository.*

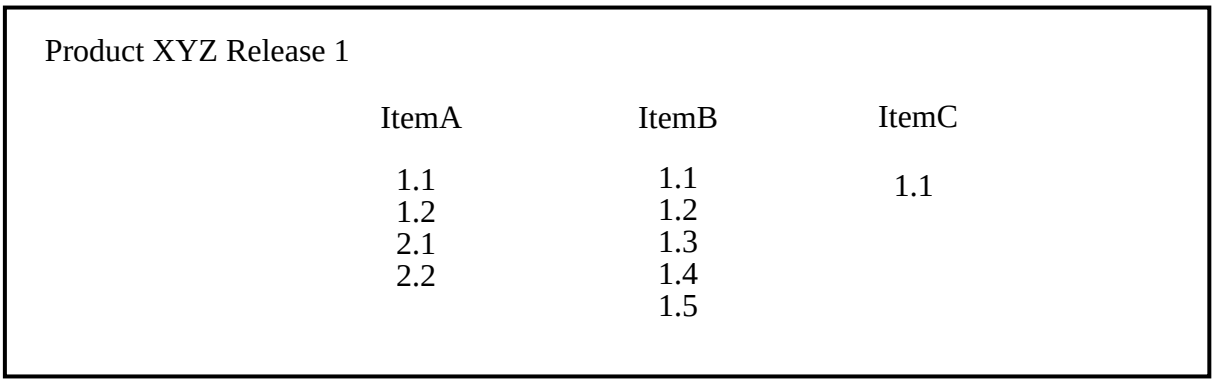
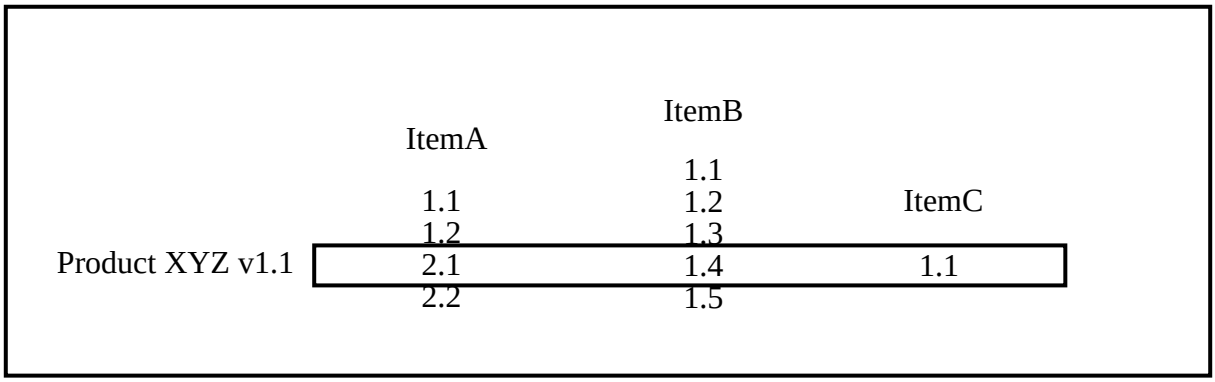


Figure 11. System Version. *The System consists of a particular version of each Configuration Managed Item.*



Copyright © 1997, Floyd W. Shackelford. All Rights Reserved

9. System Building

This section describes how to build the Product or a portion of the Product.

Developers will almost always work on a branch. However, the “official” version of the Product is always assembled and built by the Build Team from the Configuration Managed Items present on the Repository’s main line.

9.1. Building a Branch

This procedure is intentionally incomplete. This section must be customized for each Product.

Table 14: Building a Branch

Step	Activities
1.	

9.2. Building the Main Line

This procedure is intentionally incomplete. This section must be customized for each Product.

Table 15: Building the Main Line

Step	Activities
1.	

Copyright © 1997, Floyd W. Shackelford. All Rights Reserved

10.Release Management

This section describes how we manage the deployment of new releases of a Product.[9] We release the Product any time we deliver a new version of the Product to the Customer, or otherwise make the Product generally available. Typically, we do not release a new version of a Product to the Customer until we have something significant to deliver, like a fix for a serious Defect for instance.

Version Description Document. The Version Description Document (VDD) is a document which describes the content and configuration of the release. The VDD usually contains the following information:

- New features
- Defect fixes
- Software dependencies (e.g. operating system version)
- Hardware dependencies (e.g. minimum RAM requirements)
- Installation instructions

This procedure is primarily performed by the Configuration Management / Build and Integration Team, with assistance from the Project Team and Quality Assurance.

Table 16: Releasing the Product

Step	Activities
1. Build	1. Build the Product[a] 2. Complete the VDD
2. Validation	1. Run the appropriate regression and validation tests 2. Validate the VDD 3. Quality Assurance sign-off
3. Package	Package the run-time products into a deliverable format
4. Deliver	Deliver the run-time products to the Customer

a. See “Building the Main Line” on page 39.

9. Do not confuse the act of releasing the Product to the Customer with the Release Number of the Product. We do not necessarily increment the Release Number every time we release the Product to the Customer. However, when we do release the Product to the Customer, it *will* have a Version Number different from the previous release of the Product.

Copyright © 1997, Floyd W. Shackelford. All Rights Reserved

11.Quality Assurance

This section discusses the metrics to be captured by the Configuration Database, when audits of these metrics are to be conducted, and the analyses to be applied to those metrics. Audits and analyses are typically performed by the Quality Assurance Team.

Copyright © 1997, Floyd W. Shackelford. All Rights Reserved

12.Forms

This section contains sample form templates.

Copyright © 1997, Floyd W. Shackelford. All Rights Reserved

Defect

D_____ page 1 of 2

Date/Time created	
Submitter name	
Contact information	
Product name	
Item identification	
Item title	
Location	
Abstract	
Detailed description	
Assigned to	

Copyright © 1997, Floyd W. Shackelford. All Rights Reserved

D_____ page 2 of 2

Status Log		
date/time	disposition[a]	description

a. Valid dispositions are: Return, Reject, Duplicate, Accept, Working, Verify, Close

Copyright © 1997, Floyd W. Shackelford. All Rights Reserved

Change Request

CR_____ page 1 of 6

Date/Time created	
Submitter names	
Contact information	
Priority[a]	
Products to be changed	
Type of change[b]	
Abstract	
Description of the change	
Justification for the change	

a. Valid Priorities are: critical functionality, necessary functionality, desirable functionality
b. Valid Types are: Usability, Functionality, Performance, Platform, Integration

Copyright © 1997, Floyd W. Shackelford. All Rights Reserved

CR_____ page 2 of 6

Change Control Board - Initial Review				
Date/Time Submitted	Date/Time Reviewed	Reviewers	Disposition[a]	Description

a. Valid Dispositions are: Return, Reject, Defer, Accept

Copyright © 1997, Floyd W. Shackelford. All Rights Reserved

CR_____ page 3 of 6

Analysis	
Systems Analyst name	
Requirements Analysis	

Copyright © 1997, Floyd W. Shackelford. All Rights Reserved

CR_____ page 4 of 6

Impacts					
Product	Description of Changes	Budget Changes	Schedule Changes	Resource Changes	Dependency Changes

Copyright © 1997, Floyd W. Shackelford. All Rights Reserved

CR_____ page 5 of 6

Change Control Board - Detailed Review				
Date/Time Submitted	Date/Time Reviewed	Reviewers	Disposition[a]	Description

a. Valid Dispositions are: Return, Reject, Defer, Working

Copyright © 1997, Floyd W. Shackelford. All Rights Reserved

CR_____ page 6 of 6

Target Release	
Implementation assigned to name	
Items Changed	

Copyright © 1997, Floyd W. Shackelford. All Rights Reserved

Problem Report

PR_____ page 1 of 5

Date/Time received by Help Desk	
Reported by	
Severity[a]	
Contact information	
Abstract	
Description	

a. Valid Severities are: 1, 2, 3, 4.

Copyright © 1997, Floyd W. Shackelford. All Rights Reserved

PR_____ page 2 of 5

Help Desk Attendant's name	
Steps to recreate the problem	
Date/Time forwarded to Product Specialist	
Product Specialist name	
Date/Time forwarded to PR Review Board	
PR Review Board reviewers	
Priority[a]	
Delivery target	
Date/Time forwarded to Developer	

a. Valid Priorities are: 1, 2.

Copyright © 1997, Floyd W. Shackelford. All Rights Reserved

PR_____ page 3 of 5

Developer name	
Technical description	

Copyright © 1997, Floyd W. Shackelford. All Rights Reserved

PR_____ page 4of 5

Completion estimate	
Description of fix	

Copyright © 1997, Floyd W. Shackelford. All Rights Reserved

PR_____ page 5 of 5

Tests used to validate the fix	
Inspection ID	
Date/Time fix integrated with main line	
Resulting Change Requests and Defects	

Copyright © 1997, Floyd W. Shackelford. All Rights Reserved

System Development Request

SDR_____

Steering Team
Steering Team Representative
Date/Time Created
Submitter
Subject Matter Contacts
Priority (circle one): Critical Necessary Desirable
Urgency
Problem Abstract
Problem Description
Types and Numbers of Users
Expectations
Justification

Copyright © 1997, Floyd W. Shackelford. All Rights Reserved

System Development Request
Field Descriptions

Steering Team. This is the name of the Steering Team to which this System Development Request is to be submitted.

Steering Team Representative. This is the name of the Steering Team Representative who will present this System Development Request to the Steering Team for their consideration. Please include at a minimum: name, location, telephone number(s), and e-mail address.

Date/Time Created. This is the date and time at which the Submitter first filled out the System Development Request.

Submitter. This is the name and contact information of the person who is submitting the form. Please include at a minimum: name, location, telephone number(s), and e-mail address.

Subject Matter Contacts. List the names and contact information of the people who are the subject matter experts regarding this System Development Request. Please include at a minimum: name, location, telephone number(s), and e-mail address.

Priority. Circle the appropriate priority. Critical priority means that you cannot perform a necessary business function without Development support. Necessary priority means that you can perform a business function, but you cannot do so effectively without Development support. Desirable means that you can perform a necessary business function, but you can do so more efficiently with Development support.

Urgency. Express how urgent this request is in terms of a a particular date (e.g. “No later than 31 March 1997”) or a time frame (e.g. “3Q97”).

Problem Abstract. Describe the problem in one or two sentences.

Problem Description. Describe the problem in more detail.

Types and Numbers of Users. Describe the types and numbers of users which are affected by the problem.

Expectations. Describe your expectations of what a solution to this problem should provide.

Justification. Provide a quantitative description of why this System Development Request should be approved by the Steering Team in terms of: customer satisfaction, dollar expenditures, cycle time, business opportunities, etc..

Copyright © 1997, Floyd W. Shackelford. All Rights Reserved

13.References

[Cederqvist 1993] Per Cederqvist *et al.*, *Version Management with CVS for CVS 1.9* (Linkoping, Sweden: Sigmun Support AB, 1993) http://www.loria.fr/~molli/CVS/doc/CVS_toc.html

[Paulk *et al.* 1993] Mark C. Paulk, Bill Curtis, Mary Beth Chrissis, Charles V. Weber, *Key Practices of the Capability Maturity Model for Software, Version 1.1* (Pittsburgh: Software Engineering Institute, Carnegie Mellon University, 1993) L2.71-L2.83

[Sommerville 1996] Ian Sommerville, *Software Engineering*, Fifth Edition (Harlow, England: Addison-Wesley, 1996) 675-698

Copyright © 1997, Floyd W. Shackelford. All Rights Reserved