NPS-CAG-14-002

NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

Trusted Computing Exemplar: Life Cycle Management Plan

by

Paul C. Clark, Cynthia E. Irvine, and Thuy D. Nguyen

12 December 2014

Approved for public release; distribution is unlimited

Prepared for: United States Navy, OPNAV N2/N6

THIS PAGE INTENTIONALLY LEFT BLANK

NAVAL POSTGRADUATE SCHOOL
Monterey, California 93943-5000

Ronald A. Route
President

Douglas A. Hensler
Provost

The report entitled "Trusted Computing Exemplar: Life Cycle Management Plan" was prepared for United States Navy, OPNAV N2/N6 and funded in part by United States Navy, OPNAV N2/N6.

Further distribution of all or part of this report is authorized.

This report was prepared by:

_____

Paul C. Clark
Research Associate

_____

Cynthia E. Irvine
Distinguished Professor

_____

Thuy D. Nguyen
Research Associate

Reviewed by:

_____

Cynthia E. Irvine, Chair
Cyber Academic Group

Released by:

_____

Jeffrey D. Paduan
Dean of Research

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	**2. REPORT TYPE**	**3. DATES COVERED** *(From-To)*
12-12-2014	Technical	Nov 2013 to Nov 2014

4. TITLE AND SUBTITLE
Trusted Computing Exemplar: Lifecycle Management Plan

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Paul C. Clark, Cynthia E. Irvine, Timothy Levin, and Thuy D. Nguyen

5d. PROJECT NUMBER
W4C05

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) AND ADDRESS(ES)
Naval Postgraduate School
Monterey, CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER
NPS-CAG-14-002

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Rhonda Onianwa
OPNAV, N2N6 F13
rhonda.onianwa@navy.mil

LT David Rivera
OPNAV, N2/N6F1
david.j.rivera4@navy.mil

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for pubic release; distribution is unlimited

13. SUPPLEMENTARY NOTES
The view expressed in this report are those of the authors and do not reflect the official policy or position of the Department of Defense of the U.S. Government.

14. ABSTRACT

This document describes the Life Cycle Management Plan for the development of a high assurance secure product. A high assurance product is one for which its users have a high level of confidence that its security policies will be enforced continuously and correctly. Such products are constructed so that they can be analyzed for these characteristics. Lifecycle activities ensure that the product reflects the intent to ensure that the product is trustworthy and that vigorous efforts have been made to ensure the absence of unspecified functionality, whether accidental or intentional.

The overall purpose and guiding principle for this document is to provide a methodology that will result in the creation of a product that will have a high level of assurance.

15. SUBJECT TERMS
Machinery control systems, MCS, life cycle security, high assurance, system security, trustworthy systems

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			Cynthia E. Irvine
Unclassified	Unclassified	Unclassified	UU	35	**19b. TELEPHONE NUMBER** *(include area code)* (831) 656 2461

Standard Form 298 (Rev. 8-98)

Prescribed by ANSI Std. Z39.18

THIS PAGE INTENTIONALLY LEFT BLANK

CYBER ACADEMIC GROUP
NAVAL POSTGRADUATE SCHOOL

NPS-CAG-14-002

Trusted Computing Exemplar: Life Cycle Management Plan

Paul C. Clark
Cynthia E. Irvine
Thuy D. Nguyen

December 2014

ATTRIBUTION REQUEST

December 2014

The Cyber Academic Group (CAG) and the Center for Information Systems Security Studies and Research (CISR) at the Naval Postgraduate School (NPS) wish to facilitate and encourage the development of highly robust security systems.

To further this goal, the NPS CAG and NPS CISR ask that any derivative products, code, writings, and/or other derivative materials, include an attribution for NPS CAG and NPS CISR. This is to ensure that the public has a full opportunity to direct questions about the nature and functioning of the source materials to the original creators.

ACKNOWLEDGEMENT

The authors gratefully acknowledge the following organizations for providing support toward the development of this work: OPNAV N2/N6 F1.

The material presented here builds upon work supported in previous years by the Office of Naval Research.

A portion of the material presented here is based upon work supported by the National Science Foundation under Grant No. CNS-0430566 and CNS-0430598. This document does not necessarily reflect the views of the National Science Foundation.

Table of Contents

Table of Figures

Table of Tables

Preface

This Life Cycle Management Plan was developed with an explicit goal to serve as an example for those vendors who want to produce high-assurance products. Therefore, there is short commentary and explanations sprinkled throughout the document, when such additions may not normally be seen in such a plan.

1 Introduction

1.1 Purpose

This document describes the Life Cycle Management Plan for the development of a high assurance secure product. A high assurance product is one for which its users have a high level of confidence that its security policies will be enforced continuously and correctly. Such products are constructed so that they can be analyzed for these characteristics. Lifecycle activities ensure that the product reflects the intent to ensure that the product is trustworthy and that vigorous efforts have been made to ensure the absence of unspecified functionality, whether accidental or intentional.

The overall purpose and guiding principle for this document is to provide a methodology that will result in the creation of a product that will have a high level of assurance.

This document was originally written in support of the Trusted Computing Exemplar (TCX) project. The TCX was a research project with a goal to document how a high assurance product can be made and distributed in a way that is compliant with the highest level of assurance possible, as measured by the Common Criteria (CC). [1] This highest level is referred to as *evaluation assurance level 7* (EAL7). It is important to note that if an organization desires to produce a high assurance product it is not necessary to go through the official CC evaluation process to verify compliance with the CC; an organization may choose to only use the CC as an internal guide to reach an internal goal. This document was written with CC version 2.2 as its reference, which has been superseded by other versions; it will require additional effort to be compliant with the latest CC requirements to ensure that no newer requirements are missed or to tailor it to an alternative assurance standard.

This document is the overarching high-level document that guides a product throughout its life cycle, from its initial conception to its eventual retirement, by defining policy, process, and high-level procedures. This dependency is illustrated in Figure 1 by showing how all other documents are influenced by this document.

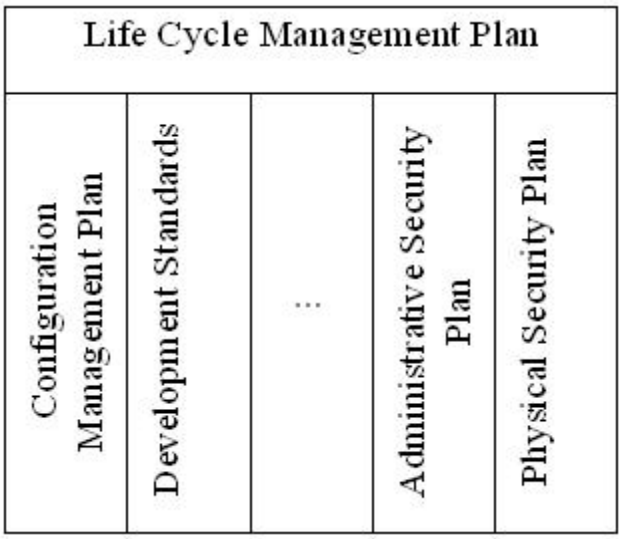

Figure 1 Influence of Life Cycle Management Plan on Other Documentation

Lower-level details, policies and requirements are spelled out in these and other documents. In addition to an overall influence on all documentation, the Life Cycle Management Plan influences all major activities, as illustrated in Figure 2.

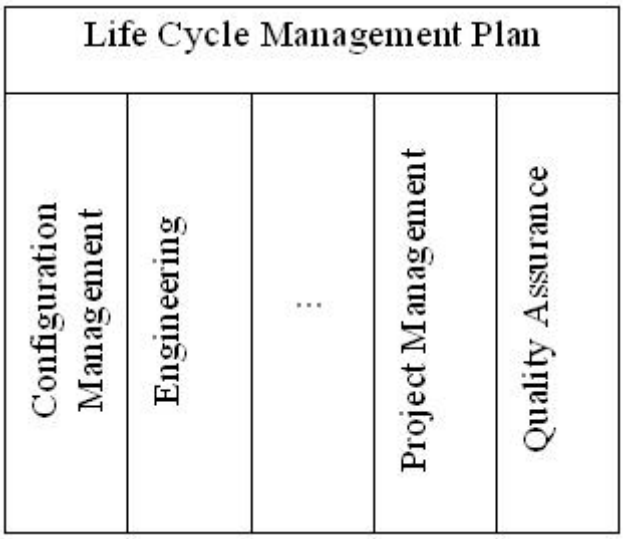

Figure 2 Influence of Life Cycle Management Plan on All Activities

The successful CC evaluation of a product requires management and staff to be familiar with the CC requirements. The purpose of this document is to state high-level requirements at logical points in the life cycle of a product, not to repeat all the detailed EAL7 requirements in a logical chronological order. The actual complete fulfillment of the EAL7 requirements will have to be undertaken by the assigned parties of a task. For example, this document states that an Architectural Description shall be provided, but the required contents of the Architectural Description are listed in the CC or a CC-mandated requirements document called the *protection profile* (PP).

1.2 The Role of the Project manager and the CCB

The Project Manager is the person who oversees the proper development of a product, using the Life Cycle Management Plan as guidance for building a product that meets the organization's standards of quality, timeliness of delivery, high assurance, etc. The Project Manager approves all efforts related to a product.

The Change Control Board (CCB) is a per-product committee that is responsible for enforcing organization-wide policies and standards for the design, implementation, maintenance, and retirement of the product it oversees. One of the first actions of a CCB is the approval of a Life Cycle Management Plan, (i.e., this document). The CCB is required to review and approve all related product submissions into the official Configuration Management (CM) system, e.g., documentation, source code, specifications, etc.

1.3 Formal Specifications

The EAL7 criteria specify that two design documents be described formally: the Functional Specification, and the High-Level Design. In addition, the Low-Level Design must be specified in a semiformal fashion. This document assumes that these three specifications will first be written in an informal style that is suitable for software engineers to understand. Another activity will then be engaged to represent them in the style required by the CC.

Common Criteria Definitions of Formality

- **Informal**
  "An informal specification is written as prose in natural language...(e.g., Dutch, English, French, German). An informal specification is not subject to any notational or special restrictions other than those required as ordinary conventions for that language (e.g., grammar and syntax)." [2]
- **Semiformal**
  "A semiformal specification is written in a restricted syntax language and is typically accompanied by supporting explanatory (informal) prose. The restricted syntax language may be a natural language with restricted sentence structure and keywords with special meaning, or it may be diagrammatic (e.g., data-flow diagrams, state transition diagrams, entity-relationship diagrams, data structure diagrams, and process or program structure diagrams). Whether based on diagrams or natural language, a set of conventions must be supplied to define the restrictions placed on the syntax." [3]
- **Formal**
  "A formal specification is written in a notation based upon well-established mathematical concepts, and is typically accompanied by supporting explanatory (informal) prose." [4]

2 Life Cycle Phases and Activities

Table 1 shows the phases of a product, and the activities that are performed in each phase. One of the purposes of this document is to specify what is done in each activity and when an activity can be considered complete. The activities in the following table are described in more detail in the remainder of this document.

Table 1 Life Cycle Phases and Activities

Phases	Activities
Conceive	Product Definition
	Requirements Definition
Design	Product Design
	Detailed Design
Build	Implementation
	Testing and Composition
	Packaging and Delivery
Modify	Maintenance
Retire	Retirement

The scope of this Life Cycle Management Plan does **not** include the following:

- **Project Management**
  This document does not set policy or procedure with respect to activities such as project estimation, resource allocation, training, etc.
- **User Installation and Operation**
  This document does not include procedures for a user to install the completed and packaged product.

3 Overview of the Spiral Life Cycle Model

A life cycle model is an abstraction for how an organization builds products. The model can by used as a management tool to ensure the right activities occur consistently in the right order. A methodology is then built around the model to show the processes used to complete each part of the building process. If followed, necessary activities are not only done, and done in the right order, they are done with completeness, and with evidence of compliance. By using the Spiral Life Cycle Model (hereafter referred to as just the "Spiral Model") in conjunction with this Life Cycle Management Plan, high assurance products are created.

Life Cycle Model

Large projects can be difficult to manage. A recognized way of managing the complexity of a large project, and to increase the likelihood of its quality, is to abstract the process into something easier to mentally grasp. The Life Cycle Model is such an abstraction. Therefore, when such models are described they typically have visually meaningful names, such as "Waterfall" and "Spiral".

The Spiral Model was selected for the TCX project because it is well known and understood, and considered one of the current "best practices" in engineering [5]. In addition, the

Spiral Model works well within an academic environment where students may enter a project at any point and refine some aspect of a design or implementation. Not all of the life cycle activities are represented in the model, which is shown in Figure 3.

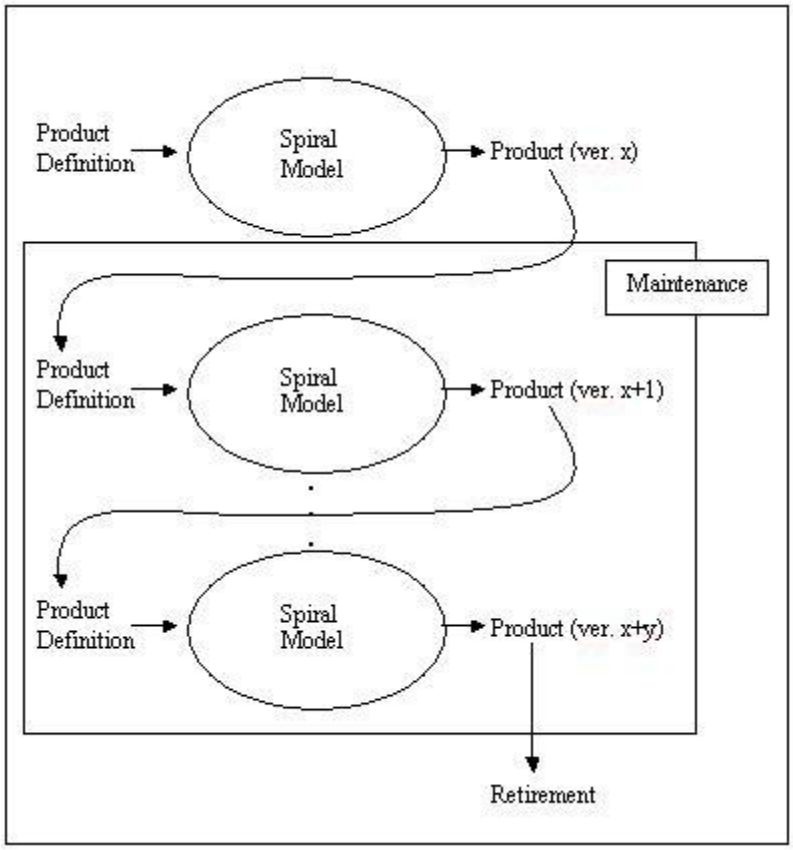

Figure 3 Life Cycle Context of the Spiral Model

The life cycle activities introduced in the previous section appear to be harshly serial, such that an activity cannot be started before the previous one is completed, which sounds like the traditional Waterfall model. The difference between the Spiral Model and the Waterfall Model is that the former allows the cycle to be completed without a deliverable product, with the understanding that iterating through the activities some number of times will eventually lead to a deliverable product. In other words, the Spiral Model allows a prototype to be developed with a compressed schedule in each activity, to allow the intermediate results to be examined before deciding whether to continue with the project. It is essentially a risk-reduction model, allowing potential design and implementation problems to be discovered early in the project.

The model can be viewed from a high level as shown in Figure 4, with the spiral starting at the inside and gradually moving outward.

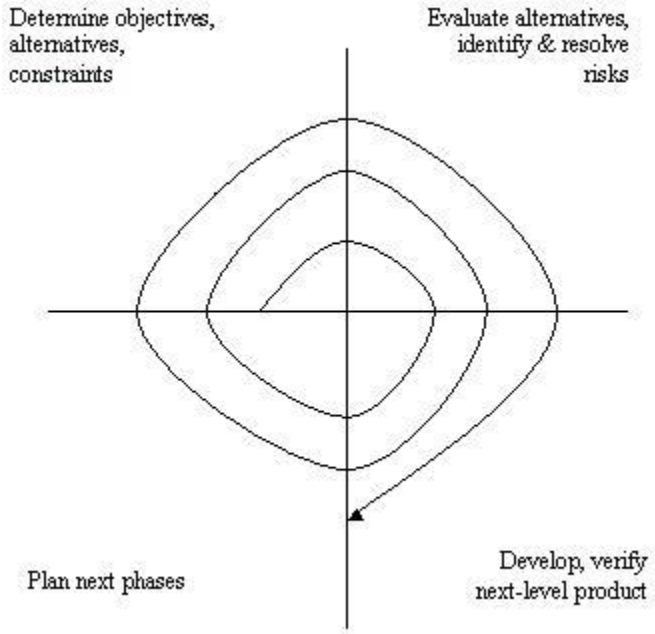

Figure 4 Simple View of the Spiral Model

The spiral intersects a plane with four quadrants, as shown above. This model can now be filled in to show the activities that are performed in each quadrant, as shown in Figure 5.

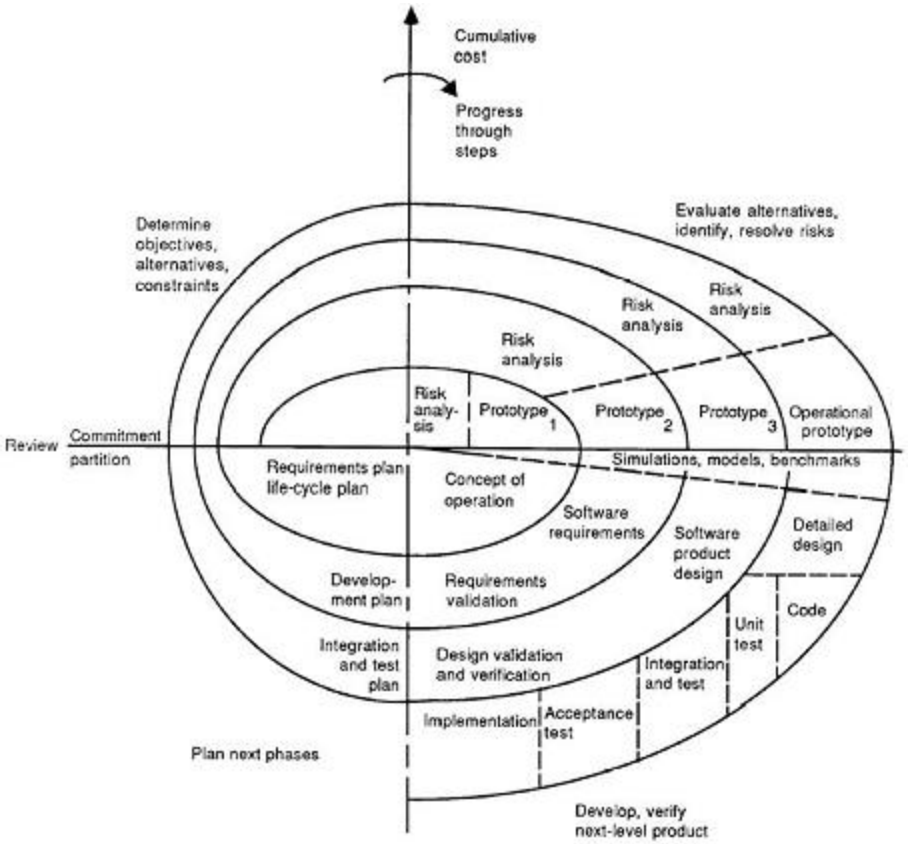

Figure 5 Detailed View of the Spiral Model [6]

The Spiral Model is very flexible and, as the outermost spiral shows, it is possible to "collapse" it into the traditional waterfall model if only one iteration produces a finished product. Such a situation may develop if the project is well understood and/or small enough to have little risk. The sections that follow describe the phases and activities in more detail.

4 Conceive Phase

4.1 Product Definition

Inputs: A product idea, Evaluation Assurance Level 7 (EAL7).

Outputs: Product Definition, Product Identifier, Life Cycle Management Plan, Configuration Management Plan, Configuration Management Procedures, Change Control Board (CCB), Configuration Management (CM) System, Configuration Items (CI) List, Personnel Security Plan, Physical Security Plan, Development Standards.

The development of a product idea occurs over time. Whether the idea is formed by engineering staff or conceptualized by upper management is not important. However, at some point management must be involved to approve a new product idea and define its scope, resulting in a high-level description of what the potential product is going to be and what it is going to do. This description becomes the Product Definition. The targeted Evaluation Assurance Level (EAL) must be identified from the outset of the project because it affects the management of the product from the very beginning. This plan targets EAL7.

After the product idea is well defined and management commits resources to the project, a Configuration Management (CM) system must be established if one does not already exist. A unique project identifier is then defined and registered with Configuration Management, and a Change Control Board (CCB) is formed.

At a minimum, the following documentation must be produced and subsequently approved by the CCB before this phase can be considered finished:

- Life Cycle Management Plan
- Configuration Management Plan
- Configuration Management Procedures
- Configuration Items (CI) List
- Personnel Security Plan
- Physical Security Plan
- Development Standards

These documents are placed under CM, either as one "documentation" Configuration Item (CI) or as individual CI's, as seen fit by the Project Manager. Other documentation will be required as the project progresses. All of this documentation falls under the CM umbrella, and therefore each document must be assigned to an appropriate CI and approved by the CCB.

The Project Manager must decide how to proceed through the Spiral Model. The answers to the following questions will help form the direction the project takes: Is the problem small and well defined to try to get a product completed in one iteration of the model? Is some level of prototyping required before any major design decisions can be made? What activities will be skipped (if any)? After all the outputs of this activity have been finished, the Product Definition activity is exited and the Requirements Definition activity is entered. The Product Definition activity may not be considered complete because it may be revisited if the model starts another iteration.

4.2 Requirements Definition

Inputs: Product Definition.

Outputs: Requirements Definition, Acceptance Plan, Acceptance Tests, (maybe a) Protection Profile (PP), Security Target (ST).

This Requirements Definition activity separates itself from the Product Definition activity by clearly stating the minimal requirements that the product must meet to be acceptable. Such requirements may include performance standards, interface requirements, functional capabilities, security attributes, etc. A product cannot be designed without sufficient detail about what it is supposed to do. If there is some doubt about some aspects of the product, some amount of prototyping, simulation, modeling (or some other tool) can help to solidify the requirements. It may take one or two iterations of the spiral before the product requirements can be confidently completed. Among other things, the Requirements Definition articulates the TOE Security Policy (TSP).

A major decision that must be made when considering the requirements is the targeted Protection Profile (PP), which will determine many of the product requirements, and how it will be developed. The number of evaluated PP's is fairly small, and the targeted EAL may further reduce the selection pool. It is possible that no suitable PP exists for the defined product and EAL, but that does not stop the product from continuing forward, though it may add additional time to the length of the evaluation process.

After the PP has been selected, then the ST shall be written, which describes how the product shall meet the security requirements specified in the PP. The ST shall be written even if there is no suitable PP for the defined product. It should be emphasized that the PP and ST only describe the security requirements of the product, and therefore only represent a subset of the overall Requirements Definition. A complete Requirements Definition shall also contain all the non-security requirements of the product.

With the Requirements Definition completed, an Acceptance Plan can be started. The purpose of the Acceptance Plan is to provide the strategy for testing a product before it is considered ready for delivery. The Acceptance Plan uses the Requirements Definition as its input. The Acceptance Tests are the tests, procedures, check-lists, etc., that implement the Acceptance Plan.

5 Design Phase

5.1 Product Design

Inputs: Requirements Definition.

Outputs: Formal TSP Model, Functional Specification, Formal Functional Specification, Architectural Design, Product Test Plan, Product Tests.

The informal TOE Security Policy (TSP), as contained in the Requirements Definition, must be translated into a formal representation known as the Formal TSP Model and supported by a formal proof.

Using the Requirements Definition as input, the capabilities and external interfaces of the product are designed and specified in a document referred to as the Functional Specification. The Functional Specification must show that the design will satisfy the requirements set forth in the Requirements Definition. The Functional Specification is also expressed “in a formal style”. This Formal Functional Specification must include rationale that all TOE Security Functions (TSFs) are represented. The Functional Specification will include all functions (security and non-security-related), but the Formal Functional Specification will only include the security-related functions. This relationship is true for all specifications that require some level of formal work. A correspondence must then be proven formally between the Formal TOE Security Policy Model and the Formal Functional Specification.

After the Functional Specification has been completed, an engineering team divides the product into a manageable number of subsystems, which are documented in the Architectural Design. Each subsystem must be described in sufficient detail concerning its function and how it interacts with the other subsystems. Breaking the system into many subsystems may make it easier to manage the product's complexity, but it increases the overhead of design because each subsystem must be separately designed, reviewed, approved, and managed. Therefore, a balance must be struck between subsystem granularity and product administration, which may be dependent on the skills of the managers and staff involved. It is also possible to take a subsystem, identify it as a separate product, and divide it into more subsystems. In addition to the software and hardware subsystems, the required documentation must also be identified, such as user manuals, programming manuals, system administration manuals, etc. The Architectural Design shall identify those CI's that shall be part of the evaluation and those CI's that shall be excluded from evaluation. Support tools, for example, might not be included if the PP does not specify their evaluation. The Project Manager shall determine to what degree the non-evaluated CI's shall adhere to the rigorous standards described in this document.

After the Architectural Design is approved (which may take a couple of iterations through the spiral model), the designated subsystems are identified as Configuration Items (CI's). A change request is then submitted through the CCB to add these new CI's to the list of items managed under CM.

After the Functional Specification is completed, a Product Test Plan can be started. The Product Test Plan is the strategy for testing the completed product for compliance with the designed external product interfaces, as documented in the Functional Specification. The Product Tests are the tests, procedures, check-lists, etc., that implement the Product Test Plan.

5.2 Detailed Design

Each subsystem must be designed in greater detail. For each subsystem specified in the Architectural Design, a High-level Design is produced, followed by a Low-level Design.

5.2.1 High-Level Design

Inputs: Functional Specification, Architectural Design.

Outputs: High-Level Design (per subsystem), Formal High-Level Design (per subsystem), Subsystem Test Plan (per subsystem), Subsystem Tests (per subsystem).

The High-level Design describes the function and interface of the associated subsystem in detail, including all interfaces that are visible outside of the subsystem.

The High-level Design must be translated into a formal representation, known as the Formal High-Level Design. The correspondence between the Formal High-Level Design and the Formal Functional Specification must be proven formally.

After the High-level Design has been completed, the Subsystem Test Plan can be started. This test plan is the strategy for testing the external interfaces of the completed subsystem, as documented in the High-Level Design. The Subsystem tests are the tests, procedures, check-lists, etc., that implement the Subsystem Test Plan.

5.2.2 Low-Level Design

Inputs: High-Level Design.

Outputs: Low-Level Design (per subsystem), Semiformal Low-Level Design (per subsystem), Internal Description, Unit Test Plan (per subsystem), Unit Tests (per subsystem), Covert Channel Analysis Plan, Covert Channel Analysis Tests.

The Low-level Design is the lowest level of design prior to implementation activities. The specification provides a detailed design of the internals of the subsystem by breaking it down into modules. The modules are also grouped into layers with the upper layers dependent on lower layers, and lower layers **in**dependent of upper layers. In other words, there must be a loop-free design among the modules. Each module is then designed in detail. The Low-level Design is also expressed semiformally. The correspondence between the Semiformal Low-Level Design and the Formal High-Level Design must be demonstrated semiformally.

The Internal Description is developed in cooperation with the Low-Level Design of each subsystem. It provides the overall module layering design for the TOE Security Functions

(TSF) across all subsystems, and explains how the internal design of the TSF meets the modularity, layering and minimization requirements of the CC.

After the Low-level Design has been completed, a Unit Test Plan can be developed and finalized. The Unit Test Plan is the strategy for testing each module. The Unit Tests implement the Unit Test Plan. The CC also requires "Implementation Tests" that use the source code as the specification for writing another layer of tests. If the Low-Level design is not heavily detailed, then a separate Implementation Test Plan and Implementation Tests shall be developed in the Build Phase. However, if the Low-Level design is sufficiently detailed, then it is acceptable to review the implementation in the Build Phase and add only those tests (to the Unit Test Plan and Unit Tests) that are deemed necessary to satisfy Implementation Test requirements, based on a metric to be determined.

Covert Channel Analysis

Covert Channel Analysis is an analysis of the design and implementation of a product to find implicit, unintended avenues of information flow that are contrary to the enforced policies, and to "estimate their capacity". At EAL7, the search for covert channels must be "systematic" [1] The search for existing covert channels is done through a variety of ways, such as an analysis of the functional interface for interaction between the function's effects on metadata and exceptions returned. If a covert channel is found, then the PP, ST or Project Manager may impose additional requirements.

Lastly, with the Low-level Design in place, the Covert Channel Analysis Plan can be written. This is the strategy for conducting the systematic Covert Channel Analysis. The Covert Channel Analysis Tests implement this plan.

6 Build Phase

6.1 Implementation

Inputs: Low-Level Design, Schematic (for Hardware), Unit Tests.

Outputs: Source code (for Software), Electronic components (for Hardware), Flaw Tracking and Remediation System, Flaw Tracking and Remediation Plan, Flaw Tracking and Remediation Procedures, (conditional) Implementation Test Plan (per subsystem), (conditional) Implementation Tests (per subsystem), Unit Test Results, Administrator Guidance, User Guidance.

If it has not been established yet, the first thing that must be done in this activity is to provide a system and procedures for reporting, receiving and tracking problem reports

(e.g., software bugs) related to the product. This documentation is referred to as the Flaw Tracking and Remediation Plan, and the Flaw Tracking and Remediation Procedures.

The product is then built according to the specifications given in the Low-Level Design, adhering to the organization's development standards. The process from Product Definition to Code is shown in Figure 6, highlighting that there are separate High-Level Designs and Low-Level Designs for each subsystem.

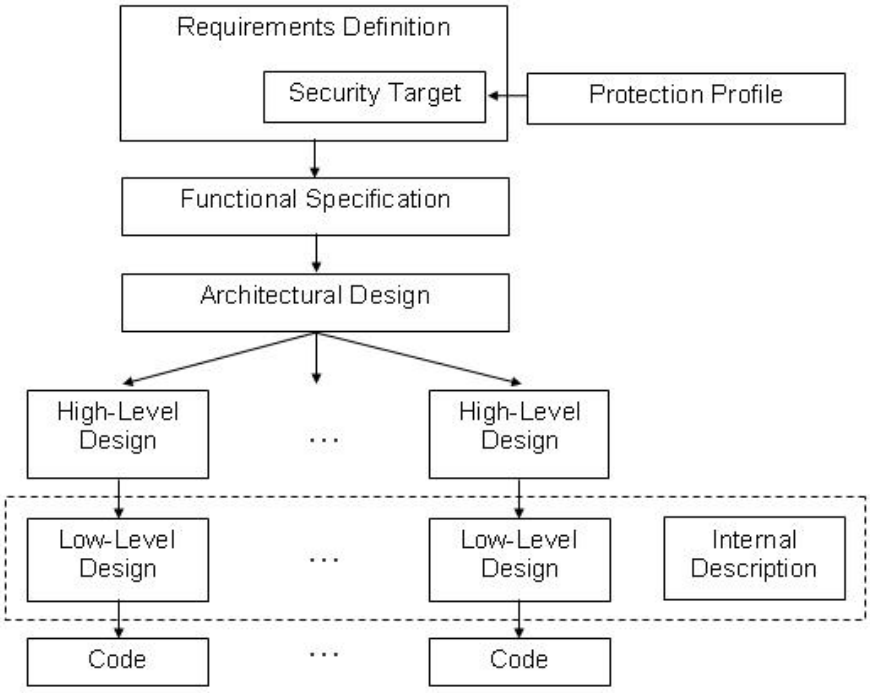

Figure 6 From Design to Code

Modules are implemented from "the bottom up", meaning that the independent modules are implemented first. After each module is implemented, it must undergo unit testing. These bottom-layer modules then form the foundation for implementing modules in the next layer up. These modules then undergo unit testing before continuing to the next layer. And so it continues until all the modules have been implemented and unit tested.

A completed hardware component must also undergo unit testing to show that it performs as specified.

As described in Section 5.2.2, a decision is made by the Program Manager (or delegated to the CCB), concerning whether the detail of the Low-Level Design is sufficient to obviate the need for separate Implementation Test Plans and Implementation Tests. If the design documents are sufficiently detailed, then the implementation is reviewed to consider which tests, if any, need to be added to the Unit Test Plans and Unit Tests in this activity. If the design documents are **not** sufficiently detailed, then separate Implementation Test Plans and Implementation Tests need to be developed in this activity, using the implementation as the specification to be tested.

The following documentation must be written before this activity is considered complete, though it could have been started in an earlier activity:

- Administrator Guidance (for the customer)
- User Guidance (for the customer)

6.2 Testing and Composition

Inputs: Implemented Subsystems, Subsystem Tests, Product Tests, Acceptance Tests, Flaw Tracking and Remediation System, Covert Channel Analysis Tests, Semiformal Low-Level Design, User Guidance, Administrator Guidance.

Outputs: Validated Product, Covert Channel Analysis Report, Representation Correspondence, Implementation Correspondence, Vulnerability Analysis, Flaw Reports, Subsystem and Product Test Results, Testing Analysis, Guidance Documentation Analysis.

As described in Section 6.1, a lot of testing is performed when code is being produced, i.e., unit testing. This section describes the testing process that must occur beyond the per-module testing. As Figure 7 shows, testing occurs at every stage to ensure that the pieces being composed are functioning according to specification **before** they are used.

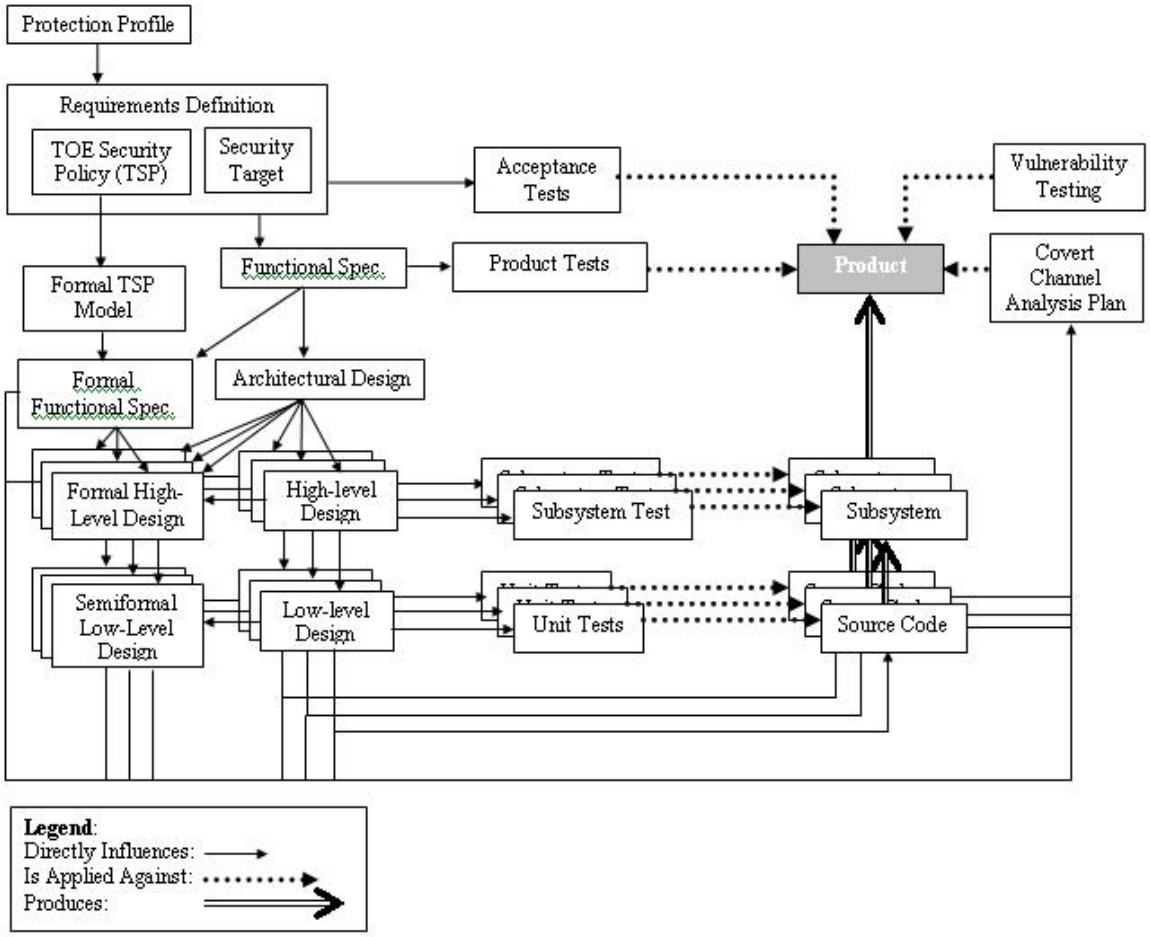

Figure 7 Relationship of Specifications, Components and Testing

After all the modules of a subsystem have been completely implemented and unit tested, the subsystem must be tested according to the Subsystem Test Plan. A subsystem can be baselined after the following have been done:

- It has passed all its Subsystem tests.
- It has undergone appropriate review.
- It has been approved by the CCB.

After all the subsystems have passed their tests, they are composed into a working version of the product. The Product Tests are applied against this composition.

The testing that has occurred up to this point is functional in nature, viz., it is ensuring that the components perform as specified. The last testing step is to validate that the finished product meets the requirements specified in the Product Definition by performing the Acceptance Tests.

In addition to the testing performed as described above, the source code is shown to be a true implementation of the security policy by showing a correspondence between the adjacent pairs of specifications. The resulting effort is recorded in a document known as the Representation Correspondence. This correspondence effort is shown between the informal specifications down to the source code, and between the formal specifications down to the source code. In addition to this pair-wise correspondence, the TSF must be shown to have a "direct correspondence" with the code, the result of which is known as the Implementation Correspondence. A less rigorous requirement for these correspondence efforts may be specified for a particular EAL or by the target PP.

Subsystem Testing

It is possible for a subsystem to be completely implemented before the subsystems it depends on are implemented. In this situation, a subsystem can be tested, as long as the subsystems it depends on are emulated in some kind of "test harness" with sufficient expected behavior of the unfinished subsystems. It is then possible to have a subsystem implemented, tested and baselined before these supporting subsystems are implemented. However, there must be a balance struck between the time and effort to implement such a test harness, and the time it will take to wait for the supporting subsystems to be completed, and the risk that the test harness will not be a true representation of the eventual behavior of the baselined supporting subsystems, which would necessitate a repetition of the subsystem tests.

The Covert Channel Analysis Tests must be performed on the product to identify and measure all covert channels. This effort concludes with a written document known as the Covert Channel Analysis Report.

Another activity that must be performed on the product is a vulnerability analysis. This analysis takes the flaws found during testing (and other means), and ensures that the flaws cannot be used to violate the enforced security policies in some way. The outcome of this activity is recorded in a document known as the Vulnerability Analysis.

An analysis must be performed on the Administrator and User Guidance documents and documented in the Guidance Documentation Analysis. This analysis must be performed by someone other than the author(s) of the documents. “The objective is to ensure that misleading, unreasonable and conflicting guidance is absent from the guidance documentation...” [7]

In conclusion, after all tests have been performed, an analysis must be made to show that the testing included sufficient depth and breadth. The outcome of this activity is recorded in a document known as the Testing Analysis.

6.3 Packaging and Delivery

Inputs: Product Definition, Requirements Definition, Functional Specification.

Outputs: Integration Procedures, Delivery Procedures, Installation Procedures, Completed Product.

Baselined Product

A product is considered baselined when all its Configuration Items (CIs), as defined in its Configuration Items List, have been baselined.

The Integration activity determines how to securely package and deliver the baselined product to the user. Some aspects of how this will be done must be considered early in the design process in order to comply with CC requirements, such as Delivery and Operation (ADO). This planning is documented in the Integration Procedures and Delivery Procedures. A product cannot be delivered to a customer until it has been baselined and approved for release by the CCB.

Another document, known as the Installation Procedures, must be written for the end-user to provide the procedures for a secure installation of the product.

Integration vs. Delivery vs. Installation

The CC requires Integration procedures [8], Delivery procedures [9], and Installation procedures [10]. These procedures have some overlap, as shown in Figure 8.

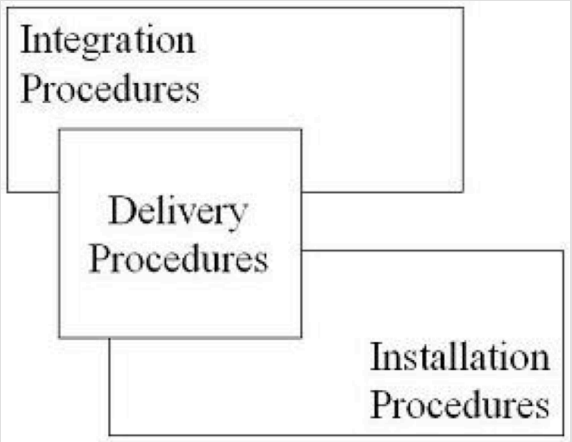

Figure 8 Overlap of Delivery Procedures

Integration procedures describe how the individual baselined items are properly obtained from the CM system during the manufacturing process, and how the product is built and packaged for a customer. The Delivery procedures describe how the product is built so that it maintains its integrity, and how the customer can verify that the product received is what was sent. The installation procedures provide the steps the customer must take to securely install and configure the product. It is acceptable to split the Delivery procedures into its manufacturing and customer parts, and merge those parts into the appropriate Integration and Installation procedures, resulting in only two physical documents.

7 Modify Phase / Maintenance Activity

Inputs: Completed Product, Flaw Tracking and Remediation System.

Outputs: Revised Completed Product, Assurance Maintenance Plan, TOE Component Categorization Report, Evidence of Assurance Maintenance, Security Impact Analysis.

This phase is officially entered immediately after a product has been completed and baselined, in order to ensure that the product maintains its level of assurance between the time of its initial completion and the release of the next version. It is therefore advisable to start writing the Assurance Maintenance Plan before the completion of a product so the plan can be completed and baselined soon afterwards.

At some point a product may need additional functionality and/or have a need to fix identified problems. Either effort requires approval from management to expend resources. Abstractly, such work can be viewed as another iteration of the Spiral Model. With respect to this Life Cycle Management Plan, the effort begins with the Product Definition activity. However, because documentation already exists for the product, it is a matter of updating the documentation to reflect new and/or modified functionality.

Assurance Maintenance Plan

The CC requires an Assurance Maintenance (AM) Plan to describe "...the plans and procedures a developer must implement in order to ensure that the assurance that was established in the certified TOE is maintained as changes are made to the TOE or its environment." [11] It is technically not a required document for the initial evaluated product, but evaluators want to review it with the initial evaluation. This is to avoid a situation where a subsequent evaluation is denied because the handling of the baselined product, or the handling of the development environment, tainted the confidence in the product. Because of the kind of information required in an AM Plan, it cannot be completely finished until all the CI's of a product release have been baselined.

All proposed changes to baselined items must first be reviewed and approved by the Project Manager before effort is expended. All proposed changes must clearly state the reason for the change, the affected CI's, and the proposed product version to receive the changes. As bugs are fixed, the bug-tracking system must be updated to reflect what was modified to correct the behavior, when it was changed, who changed it, and the expected version of the product to receive the change. Consideration should be given to notifying existing customers, especially if there is a change to the product's external interface.

The following documentation must also be provided for the re-evaluation of the product:

- **TOE Component Categorization Report**
  Each component of the TOE is categorized "according to its relevance to security". The intent is that it will provide useful input for the Security Impact Analysis.
- **Security Impact Analysis**
  This document provides a review of the changes made to the evaluated TOE (as well as the development process) and discusses the impact of those changes on the security and assurance of the new release.
- **Evidence of Assurance Maintenance**

For the most part, this document provides proof that the Assurance Maintenance Plan was followed during the maintenance activity.

8 Retire Phase / Retirement Activity

Inputs: Completed Product.

Outputs: Retirement Announcement.

After it has been fielded, management can decide that resources should no longer be expended on the support of a product (or version of a product). The minimal steps taken when such a decision is made are:

- **Set a date**
  Set a date for the termination of product support. This date should be far enough in the future that users of the product have time to consider their options.
- **Notify users**
  Send a Retirement Announcement to all known internal and external users of the product, or other means to communicate the termination of the product to the appropriate users.

9 Other Policies

If violations of the policies set forth in this document are found (e.g., unit testing was not performed on a baselined module), then an audit must take place to determine how it was allowed to happen, and whether the problem has a larger scope (e.g., no unit testing was done on all modules submitted by a particular person or group). In military terminology, a "stand down" must take place, overseen by the CCB, until the process can be put back into place. A review must then take place to consider how the problem can be avoided in the future.

All reviews, approvals and audits must have a "paper trail" to provide evidence that the required steps were performed, and to provide the necessary accountability for who approved and performed the work.

References

[1] *Common Criteria for Information Technology Security Evaluation*, version 2.2, 2004.
[2] *Common Criteria for Information Technology Security Evaluation*, version 2.2, 2004, part 3, pg. 93.
[3] Ibid.
[4] Ibid.
[5] J. Nyman. (2004, Sep. 28). Product Development Process. [Online]. Available: http://www.globaltester.com/sp4/pdp.htm
[6] B. W. Boehm, "A spiral model of software development and Enhancement," *IEEE Computer,* vol. 21, issue 5, pg. 61, May 1988.
[7] Ibid., 2, pg. 173.
[8] Ibid., pg. 79.
[9] Ibid., pg. 87.
[10] Ibid., pg. 88.
[11] Ibid., pg. 194.
[12] Ibid., pg. 142.

Bibliography

This is a list of readings that were not explicitly referenced, but which provided direction and background.

A Guide to Procurement of Trusted Systems, NCSC-TG-024, 1992.

C. E. Irvine, et al., "Overview of a High Assurance Architecture for Distributed Multilevel Security," in *Proc. of the 2004 IEEE Workshop on Inform. Assurance and Security*, West Point, NY, Jun. 2004, pp. 38-45.

"Final Evaluation Report: Gemini Computers Inc., Gemini Trusted Network Processor," National Computer Security Center, Ft. Meade, MD, Rep. No. 34-94 (Lib. No. S-241,887), Jun. 1995.

IEEE Recommended Practice for Software Requirements Specifications, IEEE Standard 830, 1988.

IEEE Standards for Developing Software Life Cycle Processes, IEEE Standard 1074, 1997.

Methodology concepts. Global Tester. [Online]. Available: http://www.globaltester.com/sp4/concepts.html. Accessed Sep. 28, 2004.

Methodology and testing concepts. Global Tester. [Online]. Available: http://www.globaltester.com/sp4/standard.html. Accessed Sep. 28, 2004.

Methodology vs. life cycle. Global Tester. [Online]. Available: http://www.globalter.com/sp4/cycle.html. Accessed Sep. 28, 2004.

Nature of life cycles. Global Tester. [Online]. Available: http://www.globaltester.com/sp4/lifecycles.html. Accessed Sep. 28, 2004.

Nature of processes. Global Tester. [Online]. http://www.globaltester.com/sp4/process.html. Available: Accessed Sep. 28, 2004.

Nomenclature roadmap. Global Tester. [Online]. Available: http://www.globaltester.com/sp4/roadmap.html. Accessed Sep. 28, 2004.

Product development process. Global Tester. [Online]. Available: http://www.globaltester.com/sp4/pdp.html. Accessed Sep. 28, 2004.

Project lifecycle models: how they differ and when to use them. Business eSolutions. [Online]. Available: http://www.business-esolutions.com/islm.htm. Accessed Sep 28, 2004.

R. C. Linger, et al., C. A. Sledge, "Life cycle models for survivable systems," Carnegie Mellon Software Eng. Inst., Pittsburgh, PA, Rep. CMU/SEI-20020TR-026, Oct. 2002.

Software engineering lecture slides. University of California, Berkeley, Computer Science Department. [Online] Available: http://www.bmrc.berkeley.edu/courseware/cs169/spring01/lectures/. Accessed Sep. 28, 2004.

Appendix A – Time Lines

Figure 9 is a representation of the dependencies on the various deliverables described in this document. The vertical lines show the relative starting times for tasks in relation to other tasks.

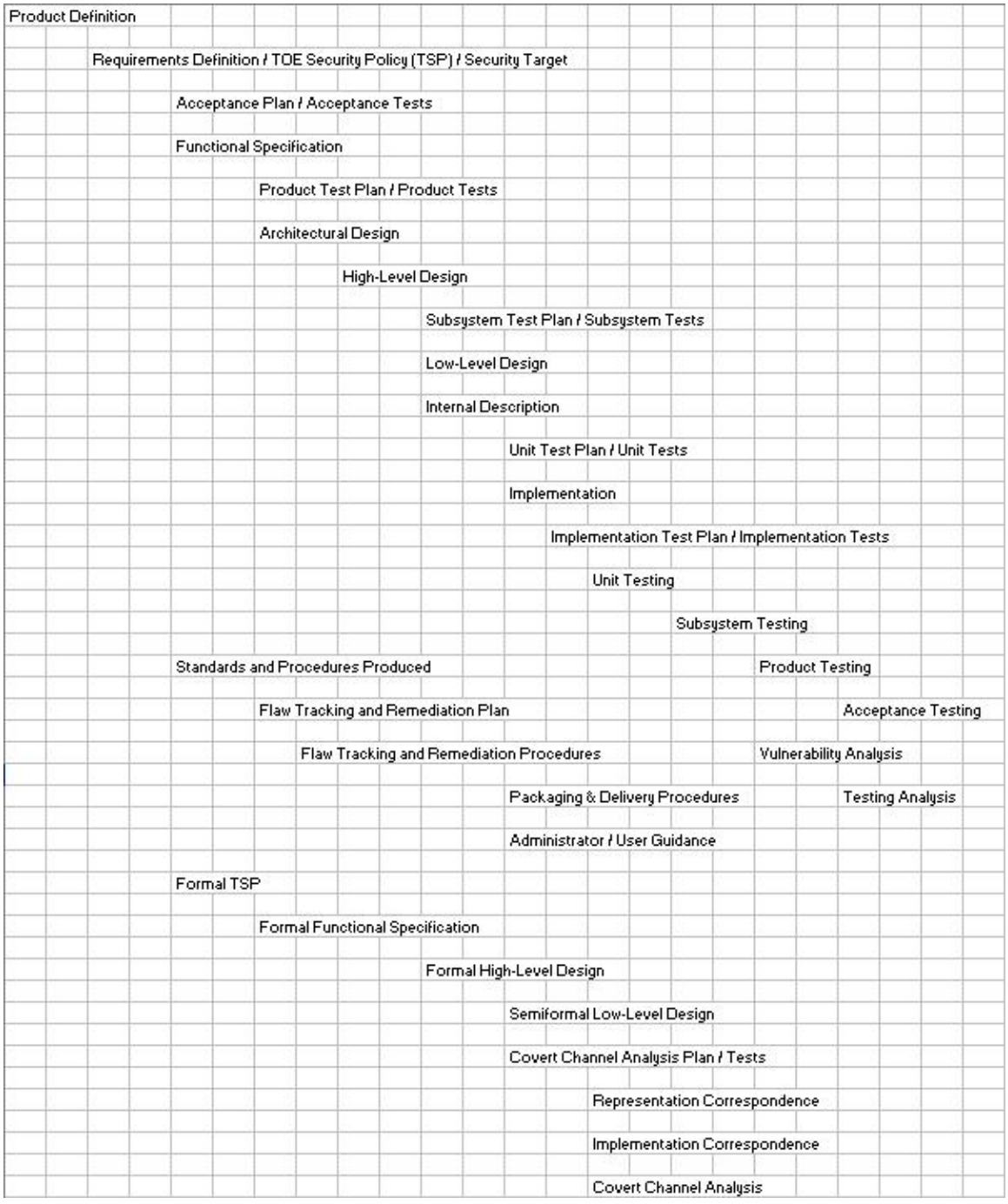

Figure 9 Time Line Dependencies

Appendix B – Overview of Documentation Generated

This appendix lists all the documentation generated by following the process described in this document. It is possible that in reality one physical document contains all the information required in more than one document listed below. The two tables provide the same information, with Table 2 sorted by Activity, and Table 3 sorted by document title.

Table 2 Documentation Required (Sorted by Time)

Document	Activity that Generates the Document
Product Definition	Product Definition
Configuration Management Plan	Product Definition
Configuration Management Procedures	Product Definition
Configuration Items List	Product Definition
Personnel Security Plan	Product Definition
Physical Security Plan	Product Definition
Development Standards	Product Definition
Project Plan	Product Definition
Requirements Definition	Requirements Definition
Acceptance Plan / Acceptance Tests	Requirements Definition
Protection Profile (may not be selected nor produced)	Requirements Definition
Security Target	Requirements Definition
Formal TSP Model	Product Design
Functional Specification	Product Design
Formal Functional Specification	Product Design
Product Test Plan / Product Tests	Product Design
Architectural Design	Product Design
High-Level Design	Detailed Design
Formal High-Level Design	Detailed Design
Subsystem Test Plan / Subsystem Tests	Detailed Design
Low-Level Design	Detailed Design
Formal Low-Level Design	Detailed Design
Internal Description	Detailed Design
Unit Test Plan / Unit Tests	Detailed Design
Covert Channel Analysis Plan	Detailed Design
Source Code	Implementation
Implementation Test Plan / Implementation Tests (Conditional)	Implementation
Flaw Tracking and Remediation Plan	Implementation
Flaw Tracking and Remediation Procedures	Implementation
Unit Test Results	Implementation
Administrator Guidance	Implementation
User Guidance	Implementation
Covert Channel Analysis	Testing and Composition
Representation Correspondence	Testing and Composition

Document	Activity that Generates the Document
Implementation Correspondence	Testing and Composition
Vulnerability Analysis	Testing and Composition
Subsystem and Product Test Results	Testing and Composition
Guidance Documentation Analysis	Testing and Composition
Testing Analysis	Testing and Composition
Delivery Procedures	Packaging and Delivery
Installation Procedures	Packaging and Delivery
Integration Procedures	Packaging and Delivery
Assurance Maintenance Plan	Maintenance
TOE Component Categorization Report	Maintenance
Evidence of Assurance Maintenance	Maintenance
Security Impact Analysis	Maintenance
Retirement Announcement	Retirement

Table 3 Documentation Required (Sorted by Document Title)

Document	Activity that Generates the Document
Acceptance Plan / Acceptance Tests	Requirements Definition
Administrator Guidance	Implementation
Architectural Design	Product Design
Assurance Maintenance Plan	Maintenance
Configuration Items List	Product Definition
Configuration Management Plan	Product Definition
Configuration Management Procedures	Product Definition
Covert Channel Analysis	Testing and Composition
Covert Channel Analysis Plan	Detailed Design
Delivery Procedures	Integration
Development Standards	Product Definition
Evidence of Assurance Maintenance	Maintenance
Flaw Tracking and Remediation Plan	Implementation
Flaw Tracking and Remediation Procedures	Implementation
Formal High-Level Design	Detailed Design
Formal Low-Level Design	Detailed Design
Formal Functional Specification	Product Design
Formal TSP Model	Product Design
Functional Specification	Product Design
Guidance Documentation Analysis	Testing and Composition
High-Level Design	Detailed Design
Implementation Correspondence	Testing and Composition
Implementation Test Plan / Implementation Tests (Conditional)	Implementation

Document	Activity that Generates the Document
Installation Procedures	Packaging and Delivery
Integration Procedures	Implementation
Internal Description	Detailed Design
Low-Level Design	Detailed Design
Personnel Security Plan	Product Definition
Physical Security Plan	Product Definition
Product Definition	Product Definition
Product Test Plan / Product Tests	Product Design
Project Plan	Product Definition
Protection Profile (may not be selected nor produced)	Requirements Definition
Representation Correspondence	Testing and Composition
Requirements Definition	Requirements Definition
Retirement Announcement	Retirement
Security Impact Analysis	Maintenance
Security Target	Requirements Definition
Source Code	Implementation
Subsystem and Product Test Results	Testing and Composition
Subsystem Test Plan / Subsystem Tests	Detailed Design
Testing Analysis	Testing and Composition
TOE Component Categorization Report	Maintenance
Unit Test Plan / Unit Tests	Detailed Design
Unit Test Results	Implementation
User Guidance	Implementation
Vulnerability Analysis	Testing and Composition

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center 2
   Ft. Belvoir, Virginia

2. Dudley Knox Library, Code 013 2
   Naval Postgraduate School
   Monterey, California 93943

3. Research Sponsored Programs Office, Code 41 1
   Naval Postgraduate School
   Monterey, California 93943

4. Paul C. Clark 1
   Naval Postgraduate School
   Monterey, California 93943

5. Dr. Cynthia E. Irvine 1
   Naval Postgraduate School
   Monterey, California 93943

6. Thuy D. Nguyen 1
   Naval Postgraduate School
   Monterey, California 93943