DOCUMENTING THE SOFTWARE DEVELOPMENT PROCESS

June S. Hopkins and Jean M. Jernow

Litton Data Systems
8000 Woodley Ave.
Van Nuys, CA 91406

ABSTRACT

The Software Engineering Process Group (SEPG) at the Data Systems Division of Litton Systems, Inc., was given the task of documenting the software development process used within the division. This paper describes how the SEPG at Litton accomplished this task. It discusses the sources we used for guidance and describes the resulting documentation for defining the software development process and the methods and tools that support the process.

After reviewing the existing software process documentation at Litton, the SEPG concluded that three separate documents were required: a revised set of Software Policies and Procedures (PPGs), a Software Engineering Handbook, and a Software Management Handbook. The SEPG established working groups to develop these documents. The working group responsible for the Software Engineering Handbook decided to develop it as a user manual for the software development process. Following Weiss' guidelines for developing a usable user manual, the working group developed storyboards for sections of the manual. A model initially developed at IBM and refined by SEI and others was used to describe the software development process as a series of work tasks, each of which has entry criteria, exit criteria, objectives, and steps to perform.

Several authors developed the storyboards and the corresponding modules of the handbook. The handbook was partitioned into short modules, each of which has a topic sentence and a figure (where applicable). The result is a modular Software Engineering Handbook that is easy to read and maintain.

The use of working groups and the development of the Software Engineering Handbook as a user manual proved to be efficient and effective methods for generating high quality software process documentation.

1.0 BACKGROUND

Litton Data Systems develops software for the Department of Defense (DoD), which is concerned with the cost of developing and maintaining software. In 1985 the DoD established the Software Engineering Institute (SEI) at Carnegie Mellon University to study the problems of software development. The SEI established a set of criteria for use by the DoD in assessing the software development maturity level of prospective contractors. In response, Litton established the Software Engineering Process Group (SEPG) to evaluate its current software maturity level and develop a plan for improving it following the SEI guidelines. The SEPG was given the following charter:

- Maintain awareness of the state-of-the-art in software engineering and associated technology
- Assess the applicability of new technology to the software development process at Litton
- Transition such technology into practice as soon as practical

One person was assigned full-time to head the SEPG with part time support of personnel from the other software development organizations at Litton.

According to the SEI, an important criterion for assessment of an organization's software maturity level is the degree to which its software development process is defined (and, therefore, documented). The SEI's reasoning is that an organization cannot improve its software development process until it has been defined. The SEPG's concern was to develop a mechanism for defining the process so that it could be communicated easily to the software engineers. The SEPG investigated various methods that had been defined in the literature [1,2,3,4].

In addition, the Director of the Software Engineering Laboratory at Litton wanted a common software engineering handbook for his 150 software engineers. He wanted to be able to move software engineers to new

© 1990 ACM 0-89791-414-7/90/1000-0125 1.50

projects with minimal retraining and to provide a readable set of documents to use in training new employees in the Litton software development process.

2.0 REVIEW OF EXISTING DOCUMENTATION

Before attempting to define and document the software development process, the SEPG examined the existing documentation, which was a set of Software Policies and Procedures (PPGs) that had been in place for many years. The PPGs consisted of policies, procedures, and guidelines for both managers and software engineers. Much of the information in them was out of date. Since only senior engineers had copies of the PPGs, many software engineers were unaware of their existence.

The SEPG concluded that as well as being out-of-date, the PPGs lacked information describing how they fit together to define a coherent software development process. Some information contained in them was oriented toward software managers; other information was directed to software engineers. The SEPG concluded that several documents were required to define the software development process. After several weeks of study, the SEPG selected the following set of documents (as shown in Figure 1):

- Software Policies and Procedures (PPGs) - Contains the policies, standards, and formal procedures for the implementation, management, and control of the software development process.

- Software Engineering Handbook (SEH) - Presents Litton's software development methodology. The SEH references the PPGs for specific information and provides the "glue" that relates the specific software development tasks to the entire process. The audience is both software engineers and software managers.

- Software Management Handbook (SMH) - Describes how to perform the software project management functions currently in practice at Litton. It contains guidance on cost estimation, project scheduling, software development metrics, etc. The audience is software managers only.

In addition, the SEPG decided that in order for these documents to be used, all software engineers should have copies of the PPGs and the SEH in their offices.

3.0 UPDATING THE PPGs

To aid in the task of updating the PPGs and to provide a sense of ownership in them by the software organizations, the SEPG established a PPG Update Working Group consisting of members from several software development organizations. The working group concept is based upon the SEPG guidelines from the SEI [9].

The working group's charter was to:

- Decide what information should stay in the PPGs and what information belongs in other documents (SEH or SMH).

- Update the information remaining in the PPGs. Add new PPGs as required.

- Review and comment on the proposed additions/updates.

- Distribute the updated PPGs to the software organizations for their review and comments.

The working group met once a week for approximately an hour. Prior to each meeting, the chairperson of the working group prepared and issued an agenda of the items to be discussed. After each meeting, the chairperson documented the meeting results and individual work assignments in an informal report that was sent to the working group members.

For consistent terminology throughout the PPGs and the handbooks, the working group decided to use the terminology in the IEEE standard on terminology [7] and to follow the verb tense usage and format shown in Figure 2.

The PPGs were divided into the following categories:

000-099 Introduction to the PPGs, purpose, procedure

100-199 Software management procedures

200-299 Documentation format and content policies

300-399 Configuration management procedures

400-499 Software engineering practices and standards

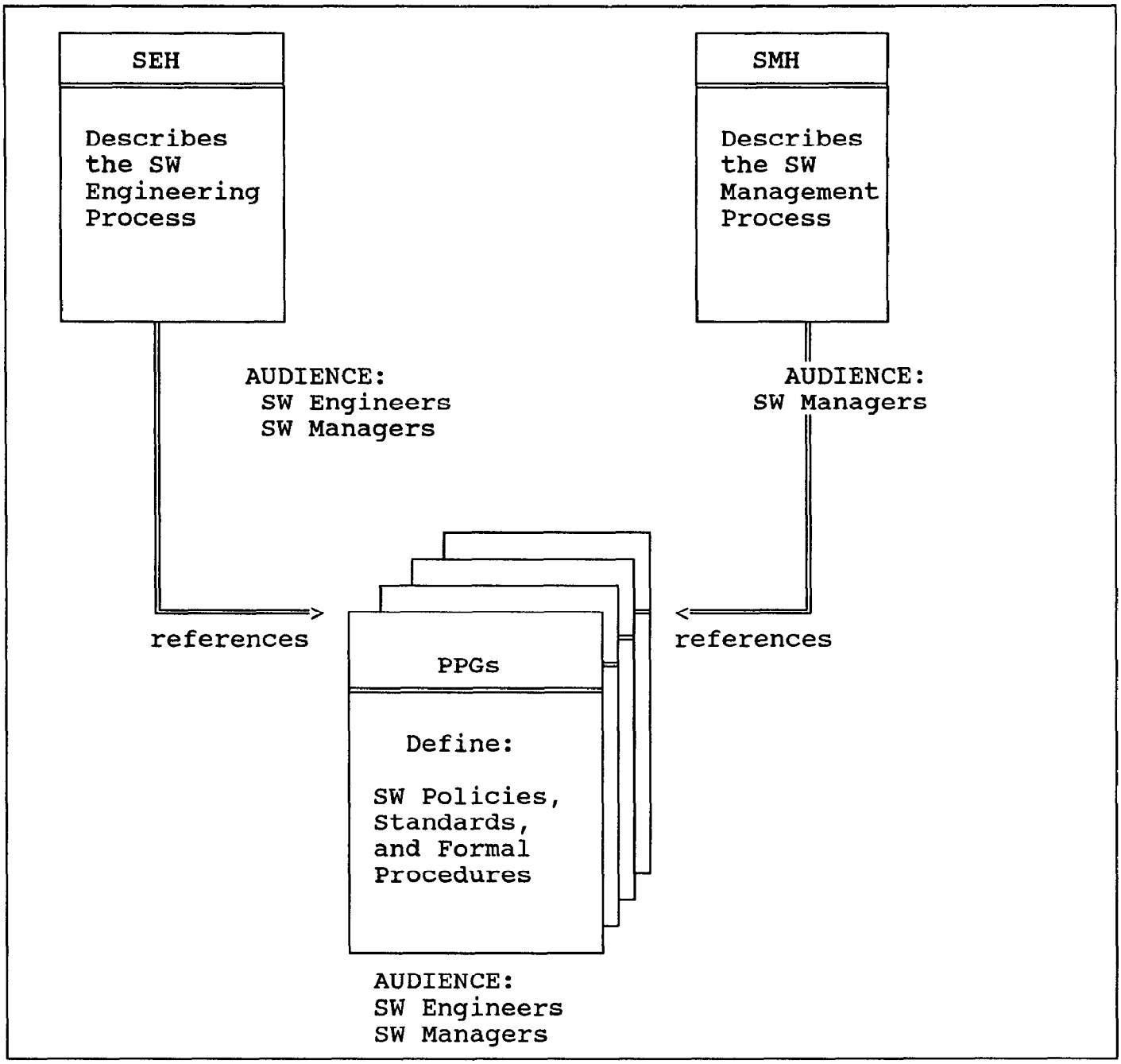


Figure 1. The Litton Software Development Process Documents

500-599 Review procedures and responsibilities

600-699 Software test policies

700-799 Computer and communication resource utilization policies

The working group developed a new PPG that defined Litton's newly established inspection process. This process was based upon the guidelines established by Gilb [6] and Fagan [5]. Since inspection can be used to uncover defects in both documents and software, the working group tested the new inspection process by using it to validate key PPGs. In this manner we verified our inspection process and inspected the PPGs at the same time. We found it effective to use relatively junior as well as senior and management personnel to inspect the PPGs. It gave them an introduction to this new inspection process and the PPGs and provided them with a sense of ownership in the PPGs.

After the PPGs had been inspected, draft copies of them were sent to the software managers for their review and

1. PURPOSE

This paragraph defines the purpose and subject matter of the PPG.

2. REFERENCES

This paragraph lists the documents that are referenced by the PPG or that affect or are affected by the PPG. The references may include Division Standard Practices (DSPs), Engineering Procedures (EPs), other PPGs, and military or DoD documents.

3. POLICY

This paragraph states the Software/Firmware Engineering Laboratory policy regarding the subject matter. Actual policy statements are written using the term **shall**. All other statements (definitions, clarifications, statements of fact or intent, etc.) are written in the present tense.

4. PROCEDURE

This paragraph describes the procedure(s) to be followed to implement the specified policy. The statements in this paragraph are written in the present tense to the extent possible.

Figure 2. PPG Format and Contents

comments. Through this review process the working group obtained concurrence on the PPGs from all software organizations. After this review and subsequent update, the PPGs were distributed to all software engineers with instructions for their usage and a request to provide the SEPG with comments and criticism.

4.0 DEVELOPING THE SEH

Once the PPGs were distributed, the SEPG established a new working group to develop the SEH. Since the SEH was a new handbook, the task to generate it was more difficult than revising the PPGs. The working group spent many hours deciding how to organize this handbook. We asked ourselves, "What should it contain? Should it be oriented toward Ada? What about projects that don't use Ada?"

The working group organized the SEH according to the software development phases defined in the DoD's standard for software development (DOD-STD-2167A [8]). The phases provide a series of steps for performing the software development process. Within the handbook each phase is described in terms of its entry criteria (what do I need to start this phase?), the steps to be performed during the phase, and its exit criteria (how do I know when I have finished this phase?).

Since most new projects at Litton will be mandated to develop new software in Ada, we decided that the handbook should describe the Ada software development methodology, which differs in some ways from the methodologies used for other languages. We decided to use appendices to contain the guidelines for (1) tailoring the methodology for non-Ada projects and (2) using the methodology for modifying existing software.

The working group used Weiss' method [1] for developing a usable user manual as the basis for developing the SEH. The working group developed an outline for the SEH and had it approved by the software organizations and the Director of Software. This process required several iterations to obtain an approved outline. Figure 3 contains a portion of the SEH outline. The working group tried to use meaningful titles for the chapter and section headings.

Once the SEH outline was approved, the working group members were tasked to develop storyboards for the SEH modules. The storyboards contained the following information:

- Module title
- Summary statement for the module
- Objective of the module
- Exhibit (table or figure) that supports the module
- Caption for the exhibit
- Points to emphasize in the module

1. Introduction
1.1 Purpose of the Software Engineering Handbook
1.2 Definition of Terms
1.3 Who Maintains This Handbook?
1.4 Format of the Handbook

2. Software Engineering and the Litton Methodology
2.1 What Is Software Engineering?
2.2 An Introduction to Basic Software Engineering Concepts
2.3 Benefits of a Standard Software Development Methodology
2.4 How to Use the Litton Methodology on Your Project
2.5 Where to Get More Information

3. The Software Development Life-Cycle
3.1 An Overview of the Life-Cycle Phases
3.2 Life-Cycle Phases Performed in the Software Laboratory
3.3 Life-Cycle Concepts: Waterfall vs. Spiral

4. Software Documentation
4.1 What Documentation Is Required?
4.2 Use Document Templates to Save Time
4.3 Trace Requirements Throughout the Software

5. Software Requirements Analysis
5.1 An Overview
5.2 What Is Needed to Start?
5.3 Steps To Perform During Requirements Analysis
5.4 Using Object-Oriented Requirements Analysis
5.5 Guidelines for Software Requirements Analysis
5.6 When Are You Through?

.
.

Figure 3. A Portion of the SEH Outline

The summary statement and objective were particularly important. If these two items could not be written, the working group reexamined the purpose of the module in the SEH to determine whether it should be retained. Figure 4 provides an example of an SEH storyboard.

Each software development phase was defined by a Fagan [5] diagram as shown in Figure 6. The steps within each phase were illustrated using step diagrams similar to those used at the Vitro Corporation [4] to define their software development process. Figure 7 contains an example of a step diagram.

Once all storyboards were written, they were reviewed by the software managers. After this review, the storyboards were transformed by their authors into text modules. To support the maintainability and usability of the SEH, the working group limited the size of the text modules to one or two pages. Each module had a topic sentence and an associated illustration. The topic sentence was usually taken from the summary sentence from the storyboard. The illustration was usually a refined version of the exhibit from the storyboard. The authors found that it was an easy task to translate a storyboard into a one or two page module. Figure 5 contains the SEH module that was written from the storyboard in Figure 4.

Some humor was included in the SEH to make it a more readable document. For example, it is a well known fact that most software engineers hate to develop software documentation. To acknowledge this fact, the following statement was included in the module on software documentation:

> Documentation is the castor oil of programming. Managers know it must be good because the programmers hate it so much.

Since each project at Litton may have unique software development requirements imposed by the contracting agency for the project, the SEH includes instructions on how to tailor the methodology in the SEH for a particular contract. For each new software development project the software manager will tailor the methodology for the project and describe this tailoring in the project's Software Development Plan. Once the methodology is tailored for a project, software engineers will use the SEH as a cookbook to perform the phases of the software development process (e.g., detailed design) and to understand how these phases relate to the entire software development methodology.

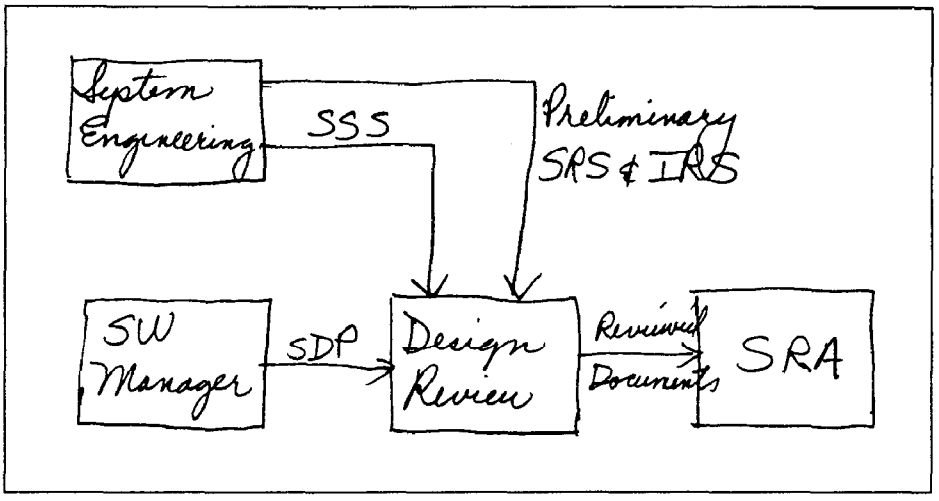
Document: SE Handbook
Author: J. Hopkins
Date: November 8, 1989

5.3 What Is Needed to Start Requirements Analysis?

Summary: Before starting software requirements analysis, the system requirements must be clearly stated. The entry criteria (see below) must be met before this activity can be started.

Objective: Convince the reader of the need for clear and consistent system requirements prior to the start of software requirements analysis.

Exhibit:



Caption: The Inputs to Software Requirements Analysis Must Be Reviewed

Points to Emphasize:

1. There is an SSS that describes the system requirements.
2. There is a SDP for the project.
3. There are preliminary interface requirements for the system.
4. All inputs have been reviewed (see PPG 400).
5. SW requirements analysis can start when a portion of an SSS has been reviewed.

Figure 4. Example of an SEH Storyboard

5. Software Requirements Analysis
5.3 What Is Needed to Start?

The specified entry criteria must be met before software requirements analysis can be started.

We need clear and consistent system requirements prior to the start of software requirements analysis. To ensure this, the following entry criteria must be satisfied:

1. There is an SSS or equivalent document that describes the system requirements.

2. There is an SDP for the project that specifies the responsibilities of Software Development and Systems Engineering. The director of the Software/Firmware Engineering Laboratory has approved the SDP.

3. There are preliminary interface requirements for the system.

Software requirements analysis can start when a portion of an SSS (or its equivalent) becomes available.

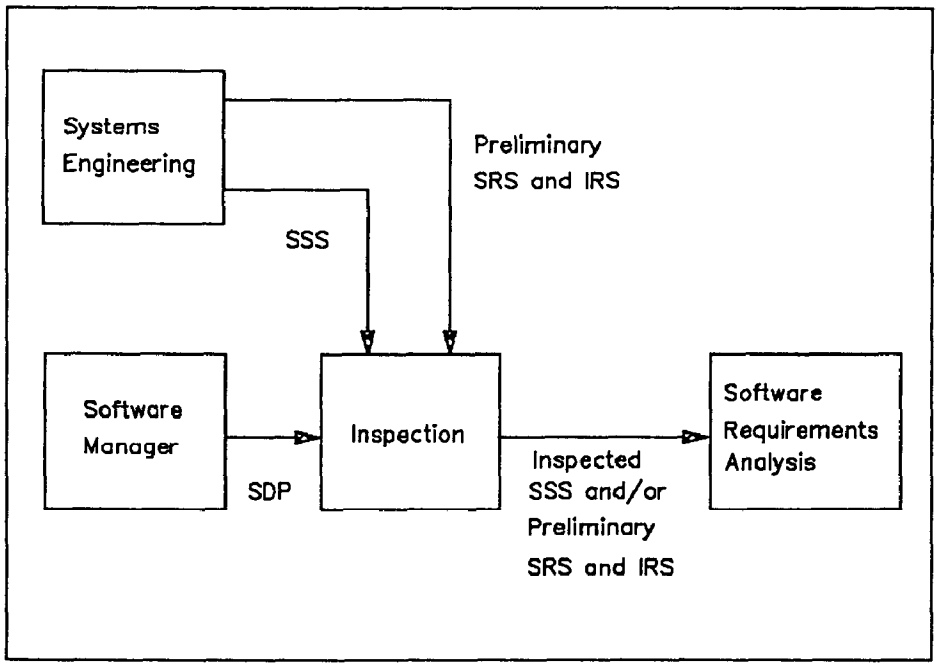


Figure 5-3. The Inputs to Software Requirements Analysis Must Be Reviewed

Figure 5. Resulting SEH Module

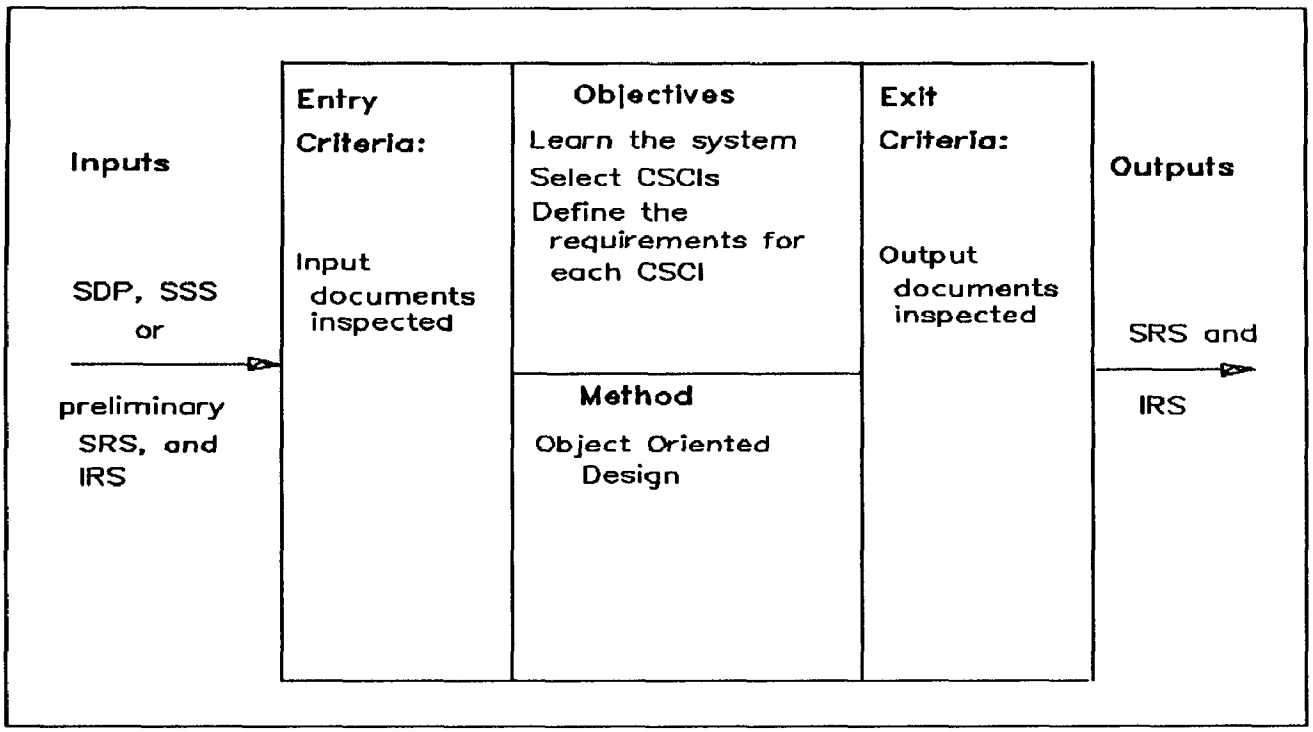


Figure 6. Example of a Fagan Diagram

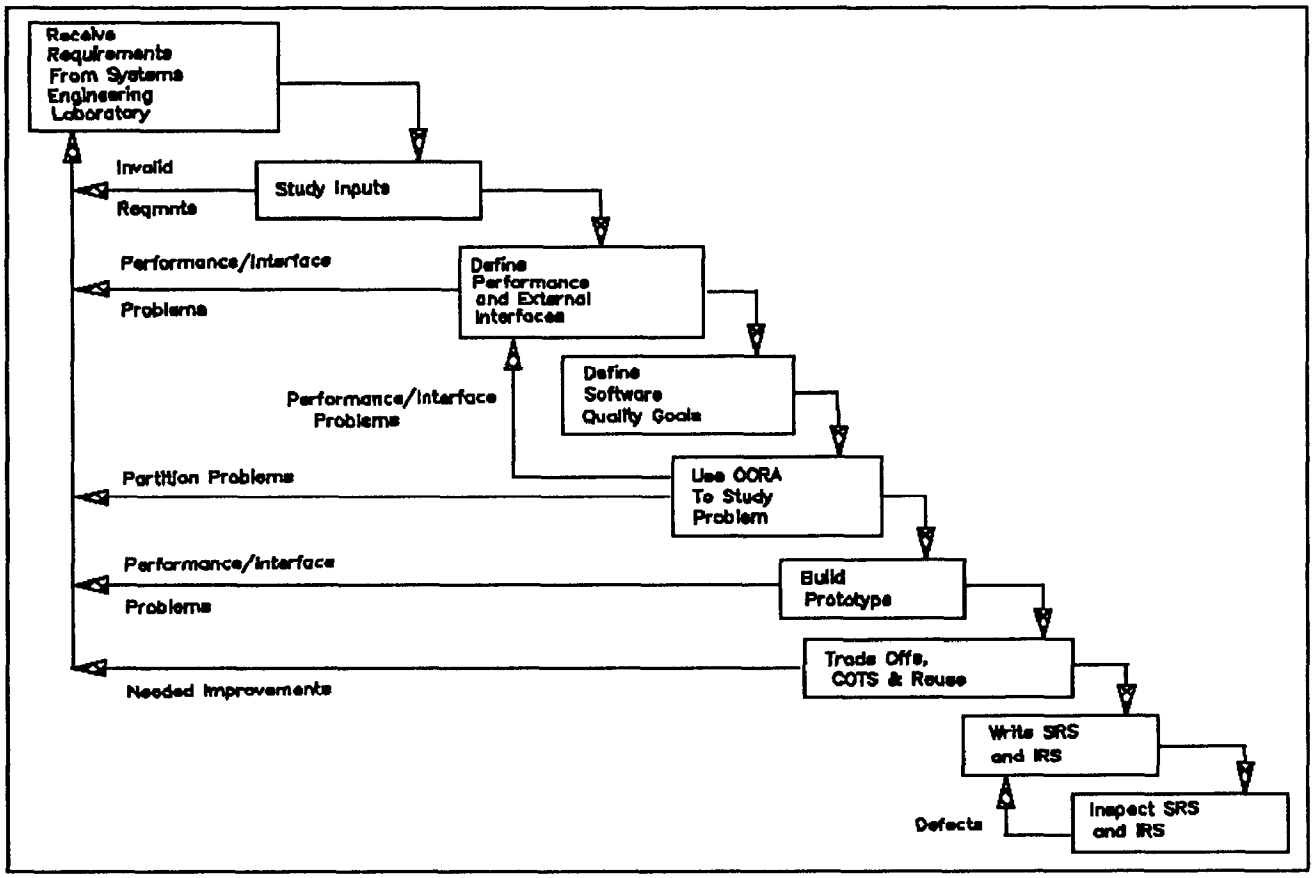


Figure 7. Example of a Step diagram

5.0 CURRENT STATUS

Besides developing the software process documents, the SEPG is responsible for maintaining them. Each document provides instructions for notifying the SEPG of errors or omissions. At least once a year the SEPG reviews all PPGs to ensure that they are clear, up-to-date, consistent with current practices, and consistent with the SEH and SMH. Similar reviews are planned for the SEH and SMH after they have been released and used.

All SEH modules and appendices have been written and are undergoing internal review and inspection prior to their release to the software managers for review. The outline of the Software Management Handbook was developed at the time of the review of the existing documentation. Once the SEH has been completed, the SEPG will establish a new working group to develop the Software Management Handbook.

6.0 CONCLUSIONS

Litton found that working groups were an effective means for developing software process documentation. Working groups (and inspections as well) provided a means for empowering software engineers by giving them a voice in defining and documenting the software development process. In addition they expanded the capabilities of the SEPG.

The working group concept worked well with Weiss' methodology for developing a usable user manual since it encouraged the use of multiple authors to prepare the storyboards and resulting text modules. In addition, the use of storyboards provided an effective method for determining the layout and content of the manual before writing the actual text. Because this method has been used for many years at Litton to write proposals, it was easy to adapt it to the writing of the SEH.

REFERENCES

[1] Humphrey, Characterizing the Software Process: A Maturity Framework, CMU/SEI-87-TR-11.

[2] Radice, et al, A Programming Process Architecture, IBM Systems Journal, Vol. 24, No. 2, 1985.

[3] Edmond Weiss, *How to Write a Usable User Manual*, ISI Press, 1985

[4] Baynes & Herman, Vitro Software Engineering Program, *Vitro Technical Journal*, Volume 6, Number 1, Fall 1987

[5] Michael Fagan, presentation given at the Fifth Annual National Joint Conference on Software Quality and Productivity, February 1989.

[6] Gilb, *Principles of Software Engineering Management*, Addison-Wesley, 1988.

[7] IEEE Standard Glossary of Software Engineering Terminology, ANSI/IEEE Std 729-1983

[8] Defense System Software Development, DOD-STD-2167A, 29 February 1988.

[9] Fowler & Rifkin, Software Engineering Process Group Guide, SEI-89-MR-1, June 21, 1989, (Draft).