IMPLEMENTATION OF A SOFTWARE PROJECT OFFICE AT HONEYWELL AIR TRANSPORT SYSTEMS

by Michael A. Ross

Abstract. This paper justifies, defines and describes an organization-level software project management process concept called a Software Project Office (SPO). The paper begins by describing the context of software development and software project management within Honeywell Air Transport Systems. It then provides the SPO justification, definition, and description. The SPO justification is based on the work of Tom DeMarco in *Controlling Software Projects*. The SPO definition (specification of requirements) is presented within sample policies for the Software Engineering Institute (SEI) Capability Maturity Model (CMM) Key Process Areas (KPAs) of Software Project Planning and Software Project Tracking and Oversight. The SPO description focuses on the Honeywell Air Transport Systems implementation of this concept.

INTRODUCTION

Purpose

This paper justifies, defines, and describes an organization-level software project management process concept called a Software Project Office (SPO) and describes the implementation of this concept at Honeywell Air Transport Systems, Phoenix, Arizona.

Scope

While the scope of actual experience upon which this paper is based is limited to Honeywell Air Transport Systems and Dutch PTT Telecom, the concept is applicable to any organization committed to implementing the SEI CMM Level 2 KPAs for Software Project Planning and Software Project Tracking and Oversight.

Background

In 1989 Honeywell Air Transport Systems (ATS) embarked on a program of Continuous Quality Improvement (CQIP) in order to perpetuate their viability as a world leader in the development and production of commercial avionics. One result of this program was the establishment of overall cost, cycle time, and quality objectives.

Concurrent with the formation of CQIP, an outside evaluation of Honeywell ATS software development was performed. This evaluation was done within the framework of the SEI CMM where, on a scale of five levels where five represents most mature, Honeywell ATS was evaluated at Level 1, “The Initial Process.” [Humphrey, 1989]

Following the evaluation, Honeywell ATS redesigned its organization to meet the challenges posed by the above-mentioned issues. Part of this reorganization included the formation of an Integrated Product Development (IPD) directorate with subordinate Processes and Tools departments. The Processes department eventually evolved into the Systems and Software Engineering Process Group (SSEPG).

Activity within the SSEPG is currently centered around achieving SEI CMM Level 2. This effort and the lessons learned from the Boeing 777 Aircraft Information Management System program (1990 through 1995) have together illuminated a critical need for a systematic software project management process.

HONEYWELL ATS DEVELOPMENT PROCESS

Process Overview

The Honeywell ATS product development process (see Figure 1) provides the context within which a successful Honeywell ATS software project management process must operate. Note that each of the development process's software-related constituent processes are assigned to one of four life cycle categories: Feasibility Study, Functional Design, Main Build, and Operations & Maintenance.

Feasibility Study (Software)

The objectives of the activities within this life cycle category are to develop a complete and technically feasible set of requirements for the system and to formulate the top level approach and plan for their implementation. Typical products resulting from these activities include: (1) a system specification, statement of need, or list of capabilities that the user or the marketing organization expects from the system, or a marketing specification; (2) a feasibility study report or an assessment of whether the need can be met with the available technology in a timely and economic manner; and (3) a set of plans documenting the project management approach to development, quality assurance, configuration management, verification, etc. [Putnam-Myers, 1992]

Functional Design (Software)

The objectives of the activities within this life cycle category are to complete the system-level design by choosing the appropriate constituent technologies (hardware, software, etc.) and to allocate each system-level requirement to the appropriate technology. Software requirements are defined. The software top-level architecture is defined. Typical products include the following: (1) specifications defining the interfaces between the system-level components; (2) software requirements specifications describing the inputs, processing (logic and/or algorithms), and outputs; and (3) updated project plans [Putnam-Myers, 1992].

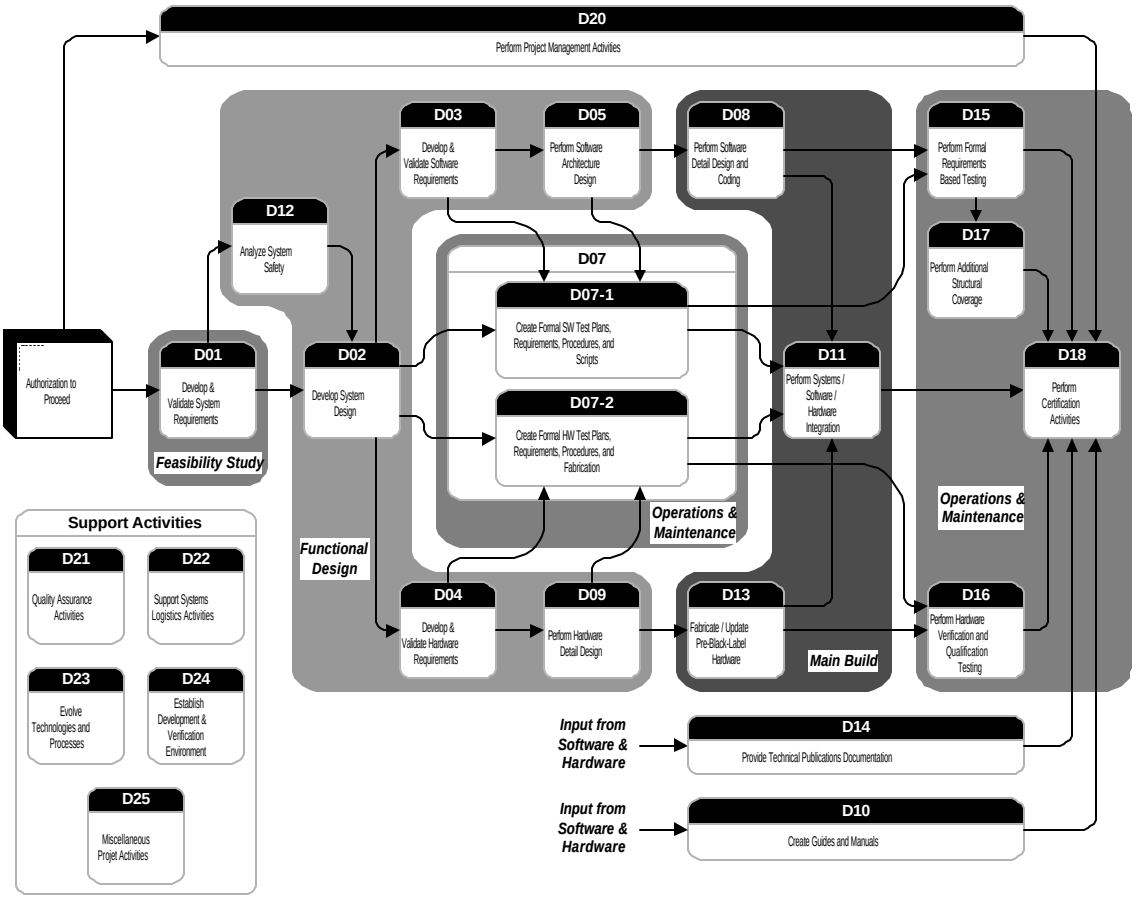

Figure 1 Honeywell ATS Product Development Process [Senechal, 1995]

Main Build (Software)

The objectives of the activities within this life cycle category are to implement the software requirements as defined by the products of the Functional Design category. Once the software requirements have been baselined, the activities in the Software Main Build category implement these requirements through detailed software design, coding, and integration. Typical products include: (1) design documentation; (2) verification (review and inspection) documentation; and (3) a full-functionality software product for which 95% of the total defects have been found and fixed [Putnam-Myers, 1992].

Operation & Maintenance (Software)

The objective of the activities within this life cycle category is to provide the customer base with product support once the system is operational. The principle activities within this category include: (1) increased reliability and assurance testing; (2) correction of latent defects; (3) new enhancements; (4) modification and tuning of the system; and (5) operation support. Typical products include: (1) testing documentation; (2) user manuals and other operating documentation; (2) maintenance manuals; and (4) other certification-related documentation [Putnam-Myers 1992].

SOFTWARE PROJECT OFFICE JUSTIFICATION

One of the keys to advancing from SEI CMM Level 1, "The Initial Process" to Level 2, "The Repeatable Process" is software project management [Humphrey, 1989]. Two of the five Key Process Areas (KPAs) associated with SEI CMM Level 2 are directly related to software project management. The first KPA is ***Software Project Planning***. The second KPA is ***Software Project Tracking & Oversight***; also referred to in this paper as "control". Additionally, a third KPA associated with Level 2, ***Software Subcontract Management***, draws from an organization's planning and control approach.

The Software Project Office concept provides ***dedicated***, ***centralized***, and ***independent*** execution of the ***planning*** and ***control*** aspects of an organization's software project management process. Comprehensive project management includes much more than planning and control (e.g., recruiting, organizing, training, equipping, and terminating). Therefore, it is important to note that the Software Project Office is ***not*** intended as a replacement for traditional software project management but rather as an enhancement to two traditional software project management activities.

Need for Dedication

Planning and control expertise is experience driven. Skills must be developed over time and with much practice. Line engineers are primarily concerned with getting products "out-the-door" and have neither time nor opportunity to practice these skills [DeMarco, 1982]. Because planning and control activities are the primary concern of the Software Project Office analysts, they ***do*** have time and opportunity to practice.

Need for Centralization

A centralized approach to planning and control is good for the following reasons:

- While the primary goal of planning and control is to support individual projects, it must also provide aggregate information to organization-level management in order to support the strategic decision-making process. A centralized approach can support this goal more efficiently and consistently than can a distributed approach.
- Economies related to tools procurement, tools development, tools and methods training, technology consulting, and external interfaces with industry and academia can be realized with a centralized approach.
- A centralized approach will better support the creation and maintenance of an organization-wide historical data repository.

Need for Independence

An independent approach to planning and control (one that minimizes adverse political influence from product-line organizations) is good for the following reasons:

- It is a conflict of interest to have the same people responsible for planning and control also be responsible for the work itself. While the planning and control processes must rely heavily on product-line personnel to ***collect*** data, the ***analysis*** of that data is best left to personnel not involved in the project being measured. The dispassionate judgment required to analyze the data and to make reasonable projections is compromised by ego involvement in performance when the same people do both [DeMarco, 1982].
- The value of planning and control outputs will suffer if, due to reporting relationships, they can be influenced by people with a stake in the outcome [DeMarco, 1982].
- The product-line development and Software Project Office processes have different goals and, therefore, should have different and independent evaluation criteria. Developers should be evaluated based on the level of success of the project. Software Project Office analysts should be evaluated based on how quickly their projections converge with what actually happens on projects and should have no stake in the success or failure of the project. It should be possible for Software Project Office analysts to be successful even though the project turns out to be a failure if that failure has been predicted in a timely fashion [DeMarco, 1982]. Likewise, it should also be possible for Software Project Office analysts to receive low marks on a successful project if that success has not been predicted in a timely fashion.

SOFTWARE PROJECT OFFICE DEFINITION

The following subordinate paragraphs contain sample organizational policies for software project planning and for software project tracking and oversight. They are included here as a vehicle for defining the Software Project Office and its role in supporting the management of software projects.

SW Project Planning Policy

Purpose

The purpose of Software Project Planning is to establish achievable plans for performing and managing software development [Paulk 1 et. al., 1993].

Software Project Planning involves developing estimates for the work to be performed, establishing the necessary commitments, and defining the plan to perform the work [Paulk 2 et. al., 1993].

Software Project Planning begins with a statement of the work to be performed and the goals and constraints that define and bound the software project (those established by the practices of Requirements Management). The software planning process includes steps to estimate the management measures (size, time, cost/effort/staffing, reliability, productivity, and manpower buildup), identify and describe the activities to be performed,

identify and assess risks and opportunities, and negotiate commitments. Iterating through these steps may be necessary to establish a baseline plan. This plan is referred to as the Software Development Plan (SDP) [Paulk 2 et. al., 1993].

The SDP documents the commitments made to the software project's customer and provides the basis for managing the software project's activities [Paulk 2 et. al., 1993].

Scope

This policy applies to all {*company name*} software development projects unless excluded by the Vice President of Engineering.

Requirements

Requirement 1

The Program Manager shall, for each software project within the program, ***identify a Software Project Manager***. The Software Project Office Manager shall, for each software project within the program, ***select from within the Software Project Office, a Software Project Analyst***.

Requirement 2

The Software Project Manager and the Software Project Analyst shall, based on a Statement of Work, the program's constraints, and those system requirements that have been allocated to software (software requirements), ***provide baseline estimates/assumptions*** for the following ***management measures***:

- **Lowest Size**, **Highest Size**, and **Expected Size** (Effective Source Lines of Code, Normalized Module Units, Function Points, or other size unit)
- **Start Date**, **Finish Date**, and **Duration** (months) of the four macro-level development categories:
    1. *Feasibility Study* (System Requirements Definition and Project Planning)
    2. *Functional Design* (System Design, Software Requirements Definition, and Software Architecture Design)
    3. *Software Main Build* (Software Detail Design, Coding, and Integration)
    4. *Maintenance* (Exposure, Problem Fixing, Assurance Testing, and Certification Support)
- **Cost** (cumulative $ over time), **Effort** (cumulative person-months over time), and **Staffing** (headcount over time) for each of the four macro-level development categories listed above
- **Reliability** (Mean Time to Defect over time) and **Cumulative Defects Over Time**

- **Productivity** (the constant of proportionality that relates size, time, and effort)
- **Manpower Buildup Rate** (the constant of proportionality that relates time and effort independent of size)

The Software Project Manager shall, for each management measure, determine the green-to-yellow and the yellow-to-red **control bounds** including a description of the **corrective actions** to be taken at each status transition.

Requirement 3

The Software Project Manager shall ***negotiate and document commitments (milestones)*** established with the Program Manager and with other organizations (internal and external) associated with the program. These commitments shall be consistent with the estimates developed in *Requirement 2* above. The following information shall be provided for each commitment:

- a **unique identifier**;
- a **description** of the commitment made and to whom;
- a list of the activities upon which this commitment is **dependent** (see *Requirement 4*); and
- an agreed-to **date** on which this commitment will be satisfied.

The Software Project Manager shall, for each commitment date, determine the green-to-yellow and the yellow-to-red **control bounds** including a description of the **corrective actions** to be taken at each status transition.

Requirement 4

The Software Project Manager shall, for each of the four macro-level development categories listed in *Requirement 2* above, ***develop a list of all the activities*** that must be completed in order for the associated category to be considered complete. The following information shall be provided for each activity:

- a **unique identifier**;
- a **description** of the activity that includes methods, tools, required skills and training, and procedures (sequence of events);
- a list of the **items consumed** by this activity;
- a list of the **items produced** by this activity;
- a specification of the **method for earning value**;
- a specification of the **completion criteria**;
- an indication of **priority** relative to the priorities of other activities;

- a list of the activities upon which this activity is **dependent** and the nature of each dependency (start-start, start-finish, finish-start, finish-finish with overlaps and gaps specified);
- an estimate of the activity's **duration**;
- an estimate of the **effort** required to complete the activity; and
- an estimate of the activity's **cost**.

Requirement 5

The Software Project Manager shall ***develop a list of risks and opportunities***. The following information shall be provided for each risk/opportunity:

- a **unique identifier**;
- a **description of the issue**;
- an estimate of the **cost/benefit of occurrence**;
- an estimate of the **probability of occurrence**;
- a description of the **key measure(s)** that will be used to monitor the risk/opportunity along with the green-to-yellow and the yellow-to-red **control bounds** including a description of the **corrective actions** to be taken at each status transition; and
- a description of any potential **risk mitigation** or **opportunity enhancement** strategies, the estimated **cost** of those strategies, the estimated **impact** on the risk/opportunity probability of occurrence, and the **circumstances** under which they will be employed.

Requirement 6

The Software Project Manager shall ***create a Software Development Plan (SDP)*** that includes all the information developed in *Requirements 2, 3, 4, and 5* above. The SDP shall also include the frequency with which Software Project Tracking and Oversight will be performed.

Requirement 7

The Program Manager shall conduct a ***review of the SDP***. Representatives of all affected organizations shall be present.

Requirement 8

The SDP shall be ***managed and controlled*** throughout the life cycle. Managed and controlled implies that the version of the SDP in use at a given time (past or present) is known (i.e., version control), and changes are incorporated in a controlled manner (i.e., change control).

SW Project Tracking and Oversight Policy

Purpose

The purpose of Software Project Tracking and Oversight is to provide adequate visibility into actual progress so that management can take effective actions when the software project's performance deviates significantly from the software plans [Paulk 1 et. al., 1993].

Software Project Tracking and Oversight involves tracking and reviewing the software accomplishments and results against documented estimates, commitments, and plans, and adjusting these plans based on the actual accomplishments and results [Paulk 2 et. al., 1993].

A documented plan for the software project (i.e., the Software Development Plan (SDP), established by the Software Project Planning process) is used as the basis for tracking progress, communicating status, and revising plans. Software management measures, activities, risks/opportunities, and commitments are periodically tracked and compared to their corresponding planned values. When it is determined that the software project's plans are not being met, corrective actions are taken. This may include revising the SDP to reflect the actual accomplishments and replanning the remaining work or taking actions to improve performance [Paulk 2 et. al., 1993].

Scope

This policy applies to all {*company name*} software development projects unless excluded by the Vice President of Engineering.

Requirements

Requirement 1

The Software Project Analyst shall use an approved version of the SDP as the basis for tracking the software project.

Requirement 2

The Software Project Analyst shall periodically (as prescribed in the SDP) ***track*** (as a function of elapsed calendar time) and compare the ***management measures*** to their corresponding baseline estimates/assumptions and control bounds contained in the SDP. The result of each comparison shall be a status determination of green, yellow, or red. The Software Project Manager shall, as a function of status, initiate the appropriate corrective action as stated in the SDP.

Requirement 3

The Software Project Analyst shall periodically (as prescribed in the SDP) ***assess*** each ***commitment (milestone)*** contained in the SDP and determine an earliest possible completion date. These dates shall be compared to their corresponding agreed-to dates and control bounds in the SDP. The result of each comparison shall be a status

determination of green, yellow, or red. The Software Project Manager shall, as a function of status, initiate the appropriate corrective action as stated in the SDP.

Requirement 4

The Software Project Manager shall periodically (as prescribed in the SDP) ***assess*** the ***activities*** within each of the four macro-level development categories (*Feasibility Study*, *Functional Design*, *Software Main Build*, and *Maintenance*) to determine the amount of value earned for each category. Earned value for an activity shall be computed according to the activity's method for earning value contained in the SDP.

The Software Project Analyst shall periodically (as prescribed in the SDP) ***track*** (as a function of elapsed calendar time) and compare the ***earned value*** for each ***macro-level development category*** to its corresponding baseline estimated value and control bounds contained in the SDP. Note that value may be expressed as dollars, person-hours, or a percentage of the total value. The result of each comparison shall be a status determination of green, yellow, or red. The Software Project Manager shall, as a function of status, initiate the appropriate corrective action as stated in the SDP.

Requirement 5

The Software Project Analyst shall periodically (as prescribed by the SDP) ***track*** (as a function of elapsed calendar time) and compare the ***key measures*** associated with each ***risk and opportunity*** to their baseline estimates/assumptions and control bounds contained in the SDP. The result of each comparison shall be a status determination of green, yellow, or red. The Software Project Manager shall, as a function of status, initiate the appropriate action as stated in the SDP.

Requirement 6

The Software Project Analyst shall periodically (as prescribed in the SDP) ***create a Software Development Status Report*** that includes all the information gathered in *Requirements 2, 3, 4, and 5* above. Each Software Development Status Report shall include the version of the SDP to which this report applies.

Requirement 7

The Program Manager shall conduct ***reviews of each Software Development Status Report***. Representatives of all affected organizations shall be present.

Requirement 8

Every Software Development Status Report shall be ***archived***.

Software Project Office Description

The Software Project Office at Honeywell ATS (see Figure 2) is a ***process*** (methods, tools, training, and activity flow). It provides an ***archive*** (history repository) for the organization's past performance. It provides history-based ***estimates*** of product ***size***. It

provides a viable project ***plan*** based on these estimates and past performance. It provides project ***control*** (tracking, forecasting, and correcting) by comparing measured actual results to the project plan, predicting the project outcome based on to-date trends, triggering corrective action when actuals deviate significantly from the plan, and ***archiving*** the final results in the organization's history repository.

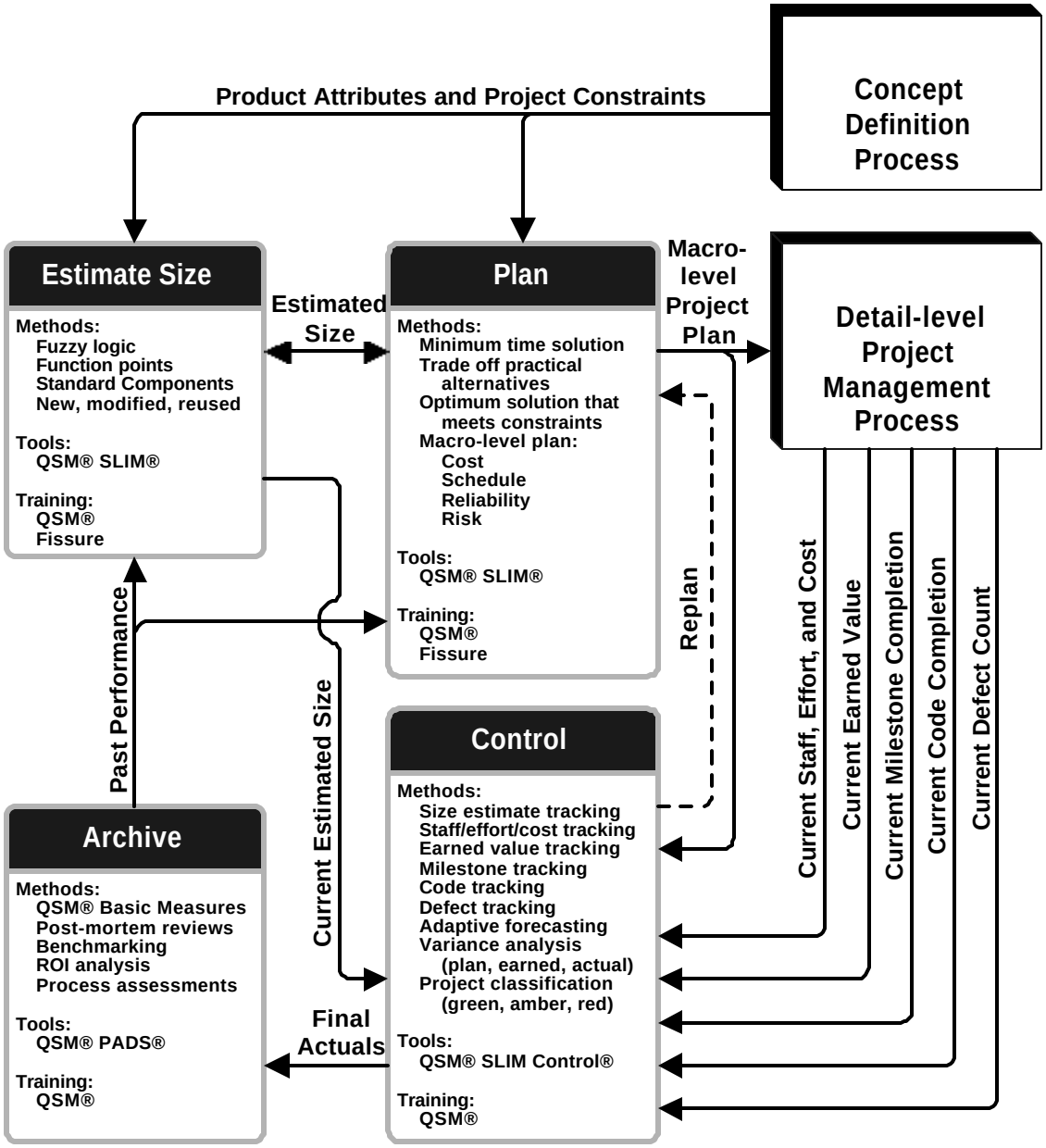

Figure 2 Software Project Office Process [Greene, 1990]

References

[DeMarco, 1982]

DeMarco, T., *Controlling Software Projects: Management, Measurement, and Estimation*. New York, NY: Yourdon Press, 1982.

[Greene, 1990]

Greene, J., "Management Measures for Excellence: The Software Control Office." UK: QSM Ltd., 1990.

[Humphrey, 1989]

Humphrey, W., *Managing the Software Process*. Reading, MA: Addison-Wesley Publishing Co., 1989.

[Putnam-Myers, 1992]

Putnam, L. & Myers, W., *Measures for Excellence: Reliable Software On Time, Within Budget*. Englewood Cliffs, NJ: Yourdon Press, 1992.

[Paulk 1 et. al., 1993]

Paulk, M.C., Curtis, B., Chrissis, M.B., Weber, C.V., *Capability Maturity Model for Software, Version 1.1*. Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1993.

[Paulk 2 et. al., 1993]

Paulk, M.C., Weber, C.V., Garcia, S.M., Chrissis, M.B., Bush, M., *Key Practices of the Capability Maturity Model, Version 1.1*. Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 1993.

[Senechal, 1995]

Senechal, B., "Lessons Learned from the Boeing 777 Program," *Proceedings of QSM Users Conference 1995*. McLean, VA: QSM Inc., 1995.