A Beginner's Look at Process Improvement Documentation

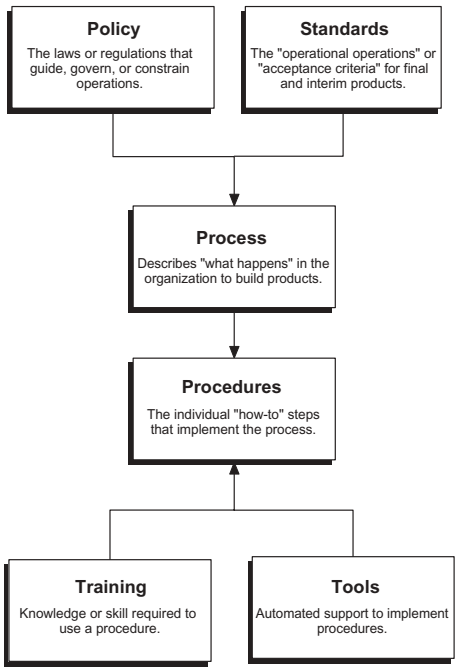
Ronald A. Starbuck
MetaVista Consulting Group

While working to get our software processes in place for process improvement, we ran across the documentation underpinning used by the Software Engineering Institute (SEI) as a guideline to implementing the Capability Maturity Model® for Software (SW-CMM®) [1] and its successor the Capability Maturity Model® Integration for Systems Engineering and Software Engineering (CMMI®-SW/SE). This underpinning is based on a very powerful yet simple structural concept useful to those just beginning the process improvement journey. By following its outline, my organization found the rewards for its improvement effort resulted in well designed and executable software processes. These are the types of processes that provide repeatability for the implementation of solid process improvement. This article looks into the fabric of this underpinning to provide a beginner's reference to understand how documentation, when done right, models process improvement for your organization.

Good processes are not just an accident! They are based on the internal associations and relationships defined and linked together by a documentation framework that can be read, interpreted, and executed by people. The framework that anchors the Capability Maturity Model® for Software (SW-CMM®) and the Capability Maturity Model® Integration (CMMI®) is derived from the structural concepts outlined in the Software Engineering Institute's (SEI) Software Process Framework (SPF) [2]. This model gives organizations the foundation they need to recognize and document the different positions and tasks used in the SW-CMM [3], and establishes the architecture needed for doing process definition.

The SPF, as we discovered, embodies what many in software development consider the foremost in *best practices* for implementing software process improvement. The architecture used in it for process definition is built on an operational framework of *process information types* comprised of policy, standards, processes, and procedures. We will explore what constitutes this process definition and the operational framework more closely in the following sections.

Figure 1: *Operational Framework*



Defining Architecture

The process definition model used in the SPF is comprised of two very important and interrelated structural concepts. These have been around mature software development organizations for a long time and considered by many to be world-class software practices. They are the following:

- First, there is an ordering hierarchy for process information types, which defines and constrains the process activities to be performed. This is identified as the *operational framework*.
- Second, there is a standard format used for consistent construction of the process information based on a defined set of process attributes, known as *process definition criteria*.

Let us look at each of these architectural concepts in more detail.

Operational Framework

The operational framework forms the relationships and dependencies between what is to be done, by whom, and how to do it. It is a very powerful yet simple hierarchical documentation concept. The relations of this concept are shown in Figure 1. They are comprised of the different process information types, which each describes a distinct set of information about the overall process. Collectively, they provide a complete operational description of how the process is done and implemented.

At the top level of the operational framework shown in Figure 1 are the controls and discipline constructs. These prescribe the organizational policies that govern operations and establish the acceptance criteria for the developed products.

The next level of abstraction is the *what-happens-and-how* constructs. They provide detail into what processes are used to build the product and the corresponding procedures that describe how to do it.

Following these at the lowest level of abstraction are the *support constructs of tools and training*. They enable the process and procedures into action in the organization through training and tools. These support information types are not process definition constructs, but the means by which processes can be put into operation by the organization.

The following are the definitions of the process information types:

- **Policy:** The laws or regulations that govern, guide, or constrain operations. They dictate the organizational approach that governs operations. Usually these laws enforce using the organization processes by identifying the required or acceptable processes or approved ways of doing work.
- **Standards:** The operational definitions or acceptance criteria for developed products. These are the operational definitions of organizational work products constraining organizational processes by setting acceptance criteria on the output of the processes.
- **Processes:** Provide the context of *what happens* over time to build products conforming to standards in accordance with the organizational policies. Processes are constrained by the organizational policies and standards; they must specify ways to develop products that conform to organizational standards in accordance with its policies. Processes are implemented by specific procedures.

- **Procedures:** Describe the *how-to* or *step-by-step* instructions that implement the process. There are many procedures to processes. They focus on how to perform a certain task identified in a process.

The following are support types the organization uses to import processes into action:

- **Training:** The knowledge/skills required to use a procedure. Training is used to support the use of processes and procedures in the organization.
- **Tools, Automated:** Provide the support needed to implement procedures, policies, standards, processes, and training needed to build software products. Tools like training are used to support the use of processes and procedures in an organization.

Using the SPF format to construct process information types removes any confusion about where the information is found for processes or where it is supposed to go, and eliminates the bad practice of mixing process information types (i.e., policy and procedure) within the same document for an organization. This is a problem that the author's organization had to clean up in its implementation effort, and unfortunately exists in many immature process documentation implementations that you should try to avoid.

Process Definition Criteria

The criteria used for process definition supports the operational framework by defining a standard format to construct process descriptions. These descriptions are comprised of the attributes that define a set of definition elements that tie together complete process descriptions. This process information is essential for doing processes and procedures. The process attributes used by the SPF establish a solid format for consistent definition of process descriptions collectively satisfying the *what*, *who*, *when*, and *how* of processes. This concept ensures that processes and procedures are completely defined and that they convey all of the information needed to enact them by people. The basic process elements that establish the criterion are shown in the flowchart in Figure 2 and described in the following process elements descriptions:

- **Roles:** What has to be done by whom to perform the activities required for the process?
- **Entry Criteria:** The conditions that must be in place to start doing the process, i.e., software requirements must be approved before starting a design and coding process.
- **Input:** Description of work products used by process, such as allocated requirements or a developer-sanctioned standard.
- **Activities:** Description of actions that are done by the process to transform the input data into a product.
- **Outputs:** Description of work products (i.e., code, documentation, lists, etc.) that result from performing the process.
- **Exit Criteria:** How we know we are done doing the process, e.g., the responsible party approves the baseline submitted by configuration management.

In addition to the basic process elements, there are several other pieces of information useful to include in process descriptions. These aid in transforming the data into work products produced from the process. They are the following:

- **Reviews and Audits:** List of reviews and audits performed during the process. What type of audits, i.e., Physical Configuration Audit, or reviews, i.e., Preliminary Design Review, are required for the process?
- **Work Products Managed and Controlled:** List of work products (code, documentation, etc.) to be managed and controlled.
- **Measurements:** Description of process measurements, i.e., lines of code.
- **Documented Procedures:** List of activities to be completed according to a documented procedure.
- **Training:** List of training for the process, formal or on-the-job training.
- **Tools:** List of tools to support the process, i.e., automated configuration management tool or spreadsheet used during the process.

Successful Considerations for Documentation

The most important consideration we found to the success of process documentation depends on the level of detail provided to the process elements. The usability of process documentation comes down to how well these process elements are described. This will determine if they will be easily understood or are too detailed for people to use. No matter how good the architecture of the SPF, the bottom line comes down to how effectively people are able to interpret, translate, and execute the documented processes.

Process information is only as good as the process elements are described. If there is too much detail, or conversely not enough, people will have difficulty comprehending and following the processes. To ensure the success of your process documentation, the writing goal is simplicity. Process elements must be simple and clear enough to use – and not too detailed to be useable [4]. An example of this style of writing is shown in Figure 3 (see page 20). This illustration example is a configuration management process.

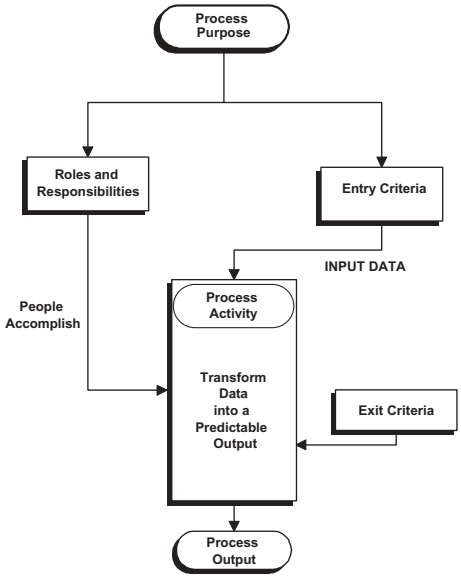


Figure 2: *Process Definition Criteria*

In this very simplistic process for configuration management, all of the process elements shown in Figure 2 were embodied. It provides readers clear, plain, and unconfused statements that inform them of the *what*, *who*, *when*, and *how* for process execution. The language used is distinct and uncomplicated. Additionally, it requires only one page to describe the process, further underscoring that being clear and uncomplicated does not require a lot of narrative pages for descriptions.

The goal is not to do one-page processes and procedures, but to determine what is appropriate and important for simple and clear process descriptions. This consideration for documenting should be given the highest priority by your organization for writing process documentation.

Factors Effecting What Is Documented

Lastly, there are a number of factors that influence how organizations finally document their processes. These factors establish the need, organizational tone for doing processes, and the funds available for who does the work. How they come together is different for every organization along with the *bottom line as* to what end the processes are documented. Organizations start out with good inten-

Configuration Management Baselining Process

Purpose of Process: To Establish Software Baselines

Roles and Responsibilities:

Configuration Manager:

- Identify what is to be baselined.

Quality Assurance:

- Verify what has been baselined is correct and current.

Entry Criteria: Life-cycle configuration milestone has been reached according to Project/System Configuration Management Plan.

Input Data: Design Progress Review, Preliminary Design Review, Test Readiness Review, or Post Deployment Support.

Process Activity: Configuration manager does following procedures:

Baseline procedure to create baseline identification in database, configuration log, and version release records.

Verification procedure to coordinate with quality assurance to ensure correct configuration has been baselined.

Status Accounting procedure to ensure project and others baseline visibility of evolving configuration is under control.

Exit Criteria: Release or identification of the following:

- Baseline Identification
- Entry into configuration log
- Database Identification completed

Process Output: Baseline established or updated

Figure 3: *Configuration Management Baselining Process*

tions for doing processes, but each of these factors has an impact on the outcome of how the documents are finally done and must be considered in your process development. The following factors are the ones the author's organization found to be important to documentation:

- **Size of the Organization:** Larger organizations have the need for more communications due to their size and the number of people involved in processes and projects. They are generally provided with a budget that supports doing processes. Conversely smaller organizations do not have the same communications problems. Because of a smaller size and fewer employees, everyone is easily informed to be able do his or her part of the job. Generally, there is little or no budget for doing processes.
- **Organizational Culture:** This is how the organization thinks and has evolved over time. Its founders have deeply seeded what this is over time by their convictions and philosophies. Organizational culture changes and evolves very slowly over time.
- **Available Budget:** This is the bottom line – funds have to be there for the organization to do anything.
- **Degree of Support for Process Definition:** This is the resource provided to do the processes in the organization. It indicates whether an organization is dedicated to full-time positions, or those extra duties in addition to a full-time job. The outcome will be different based on who is doing the process documentation implementation.

Conclusions

The author's organization found software process improvement is predicated on documentation that is defined by standard definition criteria and modeled by a structural framework. This framework forms the relationships and dependencies of operation among what is to be done, who will do it, and how it will be done.

The software process model used for the SW-CMM and the CMMI is the SPF. It is enabled by the process information types of policy, standards, processes, and procedures that are defined by definition elements that enable construction of consistent process descriptions. These concepts work together to achieve processes that are interpretable, translatable, and executable by people who do them.

The level of detail provided to the process definition elements (criteria) is the key to the success of the process documentation. Documentation is only as good as the information provided for doing processes. There are a number of factors that influence how organizations finally document their processes. These various factors establish the need, organizational tone for doing processes, and the funds available for doing the work. How they come together is different for every organization and the *bottom lines* as to what end the processes are documented.

These are the concepts the author's organization found when doing process implementation. Hopefully, they will provide beginners with the foundation they need to understand how to model process documentation that results in successful software process improvement.◆

References

1. Paulk, Mark, et al. Capability Maturity Model® for Software, Ver. 1.1. TR CMU/SEI-93-TR-24. Pittsburgh, PA: Software Engineering Institute, 1993.
2. Software Engineering Institute. SEI Software Process Framework (SPF). CMU/SEI-94-HB-1. Pittsburgh, PA: Software Engineering Institute, Sept. 1994.
3. Caputo, Kim. CMM Implementation Guide. Addison-Wesley, 1998: 70.
4. Software Engineering Institute. Questions and Answers on the CMM. Issue No. 2. Pittsburgh, PA: Software Engineering Institute, Aug. 1994: 16.

About the Author

Ron Starbuck is a consultant with MetaVista Consulting Group. He has more than 24 years experience in all aspects of configuration management, and also in software quality assurance, process engineering, and programming. Starbuck has implemented and maintained software configuration management for large- and small-scale software efforts in both the Department of Defense and the private sector. While at the Sacramento Army Depot, he received the U.S. Army's Civil Service Achievement Medal for implementing and managing configuration management. Starbuck authored the Configuration Management Section for the Army's Test Program Set Procedures Manual. He was responsible for putting together the architecture for software configuration management in conjunction with the business rules for developing pre-processors used by the Output Technology Systems. He is cofounder of the El Dorado Hills Chapter of the Software Process Improvement Network and believes in furthering professional software development in greater Sacramento, Calif.

MetaVista Consulting Group
Phone: (916) 933-2398
E-mail: ronstarbuc@email.msn.com