THE PROCESS GROUP

POST

Vol. 9, No. 2

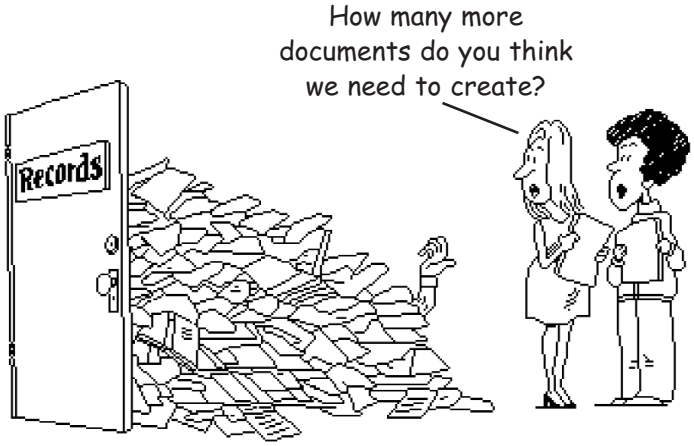
HELPING YOU IMPROVE YOUR SOFTWARE ENGINEERING PROCESS

October 2002

PRACTICAL PROJECT AND PROCESS DOCUMENTATION

By Neil Potter and Mary Sakry

INTRODUCTION



Documentation! We hate to do it and we are upset when it seems like irrelevant paperwork to please some external party. When we don't have any, we can feel left in the dark because of a lack of direction and communication among colleagues, and mentally taxed because we have to remember everything that has ever been discussed. If you are experiencing either too much or too little documentation, there is an effective middle ground; one that allows you to be productive in your projects, but not taxed with irrelevancy.

This article is a brief look at some strategies you can use to make documentation work for you. The examples contain a mix of project and process documentation scenarios.

(Continued on pages 2-6)

Shoestring Process Improvement

– don't stop because times are tough

By Mary Sakry

It's natural for anyone cutting expenses to ask, "Is this expenditure on improvement necessary?" In the short term, sometimes we do need to stop spending and let things settle down a little. But extended short-term thinking can hurt us for months and years to come.

If you have begun some improvements (such as better estimation, peer reviews or risk management), sometimes you will experience reduced productivity or increased expense (while you practice the new skill), with the anticipation of significant future savings. Let's take software inspection (peer reviews) for example. If you have started to inspect your code and documents, the defects you find will save you money and reduce rework — all contributing to a healthier bottom line. But often the first common reaction during difficult times is to cut out inspections to "save time".

Learning how to get better requirements would be another example. Getting good requirements takes time, but the information gained can tell you which features to focus on and which features to eliminate.

Continued on page 3

THE PURPOSE OF DOCUMENTATION

(Continued from page 1)

In the 15 years we have been observing companies and helping them improve, a common cause of irrelevant or overwhelming stacks of paper is a lack of purpose or objective. When we ask the average team member why he or she has so much documentation, a common answer is, "Because my organization requires me to fill out the templates." If we ask about the usefulness of such documentation, a typical reply is, "I guess it will provide a trail of what has happened so that management can study my project later." But almost no one ever goes back and ploughs through "the stack." With purposes as unclear as these, it is not surprising that people "fill out the templates."

Evidence is free when good practices are followed.

What is the purpose of documentation? Here are two examples for project and process documentation:

- **Project documentation:** a method of concisely capturing and sharing critical project concepts, plans and information as they are developed, so that impacted parties can share this information, make informed decisions, and keep the project moving forward without having to revisit old discussions.
- **Process documentation:** a method of capturing and sharing engineering and management practices so that an organization can remember, reuse and refine its skills and not have to re-invent lessons learned and best practices for each new project.

Note that we did not include in our definition "a form of evidence to please managers or auditors." If one defines the information that should be captured to manage a project effectively, this *natural* document should provide ample evidence that certain practices are occurring. For example, if we plan a project *correctly* and capture the details so that they can be communicated to others, the natural document that results (the plan) should be ample evidence that planning took place. Evidence is free when good practices are followed.

Before you develop any document, ask yourself:

- **What is the purpose of this document, why does this information need capturing and who is going to use the document when it is complete? If there is no "user," maybe it is "useless."**
- **If this document were not created, what would the risk be to the project's success?**
- **Is the information we capture critical or are we just "Filling out the template," because "We believe that we have to?"**

If you can't provide good answers to these questions, stop until you can. If you have sound responses, continue reading the following strategies to make your documents more effective.

Strategies for Making Documentation Practical

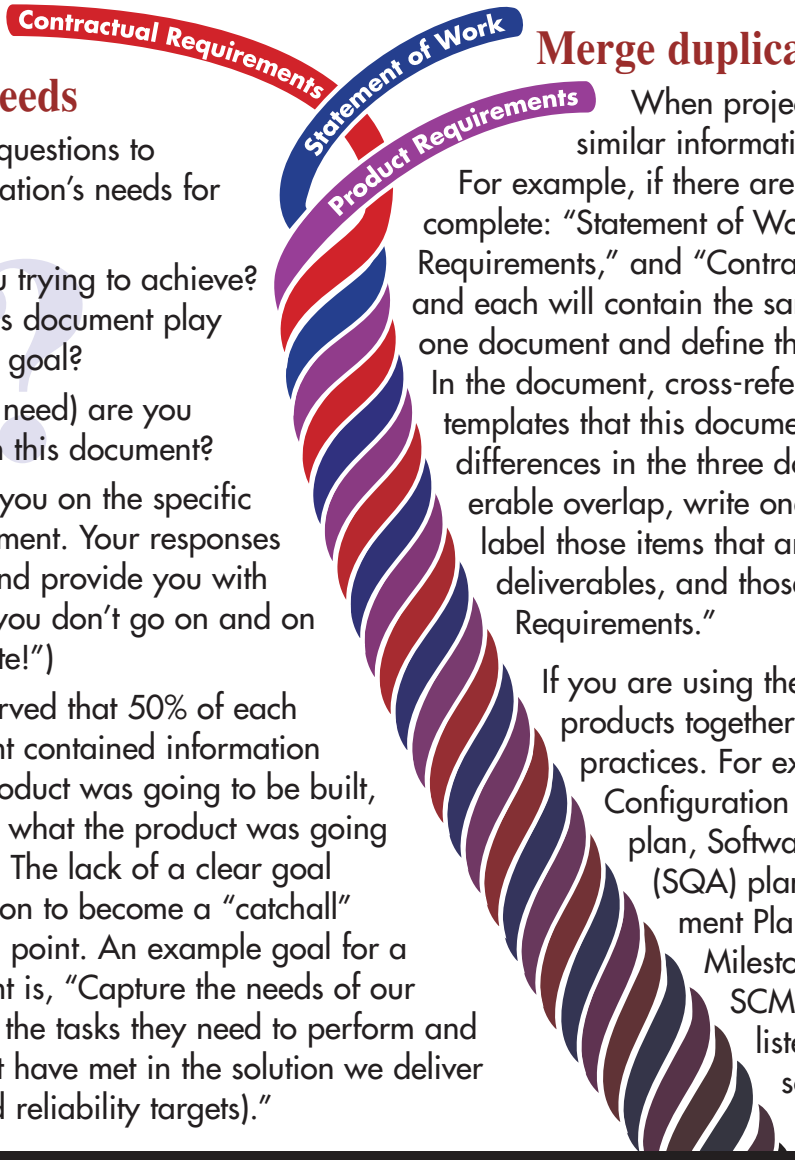
Focusing on the organization's needs

Answer the following questions to determine the organization's needs for each document:

- What goal are you trying to achieve? What role does this document play with respect to this goal?
- What problem (or need) are you trying to solve with this document?

These questions focus you on the specific purpose of each document. Your responses scope the document and provide you with an end point (so that you don't go on and on "Filling out the template!")

In one group we observed that 50% of each requirements document contained information describing how the product was going to be built, instead of focusing on what the product was going to do for the end user. The lack of a clear goal allowed the specification to become a "catchall" document with no end point. An example goal for a requirements document is, "Capture the needs of our customers by defining the tasks they need to perform and expectations they must have met in the solution we deliver (i.e., performance and reliability targets)."



Merge duplicate work products

When project documents contain similar information, merge them together. For example, if there are three documents to complete: "Statement of Work," "Product Requirements," and "Contractual Requirements," and each will contain the same information, write one document and define the information one time. In the document, cross-reference the other two templates that this document satisfies. If there are differences in the three documents, but considerable overlap, write one set of requirements, label those items that are "Statement of Work" deliverables, and those that are "Contractual Requirements."

If you are using the SEI CMM[1], merge work products together to implement specific practices. For example, a Software Configuration Management (SCM) plan, Software Quality Assurance (SQA) plan and Software Development Plan (SDP) can be merged. Milestones and activities for SCM and SQA might be listed on the master schedule in the SDP.

Shoestring Process Improvement *(Continued from page 1)*

When adversity hits, you can and should examine everything you are doing. Is there a cheaper, yet reasonably effective way to do it? Perhaps you could use fewer reviewers in the review meeting, or test the software differently. Can you do it with fewer resources? Perhaps you could have two people working on that design or get by with one less machine. Can you reduce the frequency? For example, meet once a week instead of every day. Can you combine some activities? For example, combine a periodic status review with a milestone review. Can this improvement wait a few weeks? (If it requires significant training or deployment, maybe next quarter would be a wiser time frame.)

Perhaps you can focus on only fixing the most pressing problems. Maybe you can streamline meetings by using a strong moderator and halve the time you usually take. You could also eliminate altogether the cause of some of the meetings.

Good process improvement balances the current business needs with ensuring long-term company viability. When times are tough, you need to re-evaluate how to spend your time and what needs to be improved. The last thing you want to do is cut out the very things that are going to allow your company to survive.

[1] *Software Engineering Institute Capability Maturity Model*

Remove redundancy in templates

Closely examine sections that are redundant in your templates. The template might have looked sound when first created, but during use you might find that some of the sections contain the same information. Each use of the template is an opportunity to put the template "on a diet" and delete redundant sections. For example, the requirements template in Figure 1a can be slimmed down to the template in Figure 1b when we realize that everything said in sections 1 and 4 have already been said in sections 2 and 3.

Requirements Spec.

1. Product Objectives
2. Business Requirements
3. Product Advantages
4. Value Proposition

Figure 1a

Revised Requirements Spec.

1. Business Requirements
2. Product Advantages

Figure 1b

Figures 1a and 1b. Redundant template sections are removed.

Process documents also suffer from a lack of purpose clarity. For example, suppose you are using the SEI CMM and have been chartered to develop a process for creating project schedules (Software Project Planning activity 12). You might be tempted to build the world's greatest and most comprehensive schedule creation process, with all known "bells and whistles." In the document one could regurgitate the CMM text, include references to 15 books on the subject, and refer to "Critical Chain Analysis," (whatever that is!). The appendix could include three pages of cross-references to other models and standards.

Alternatively, ask the first question and you might decide that the goal is to determine which product features can be complete by the established delivery deadlines given the available resources. This process describes how to develop a schedule to help achieve that goal.

The second question would bring out the problem(s) you want to solve. An example is to prevent your project from chronically over-committing, causing financial loss to the company. Now write a small process to accomplish these two items. An example is shown in Figure 2.

Schedule Creation Process to Scope a Project and Avoid Financial Loss Due to Over-commitment

1. Determine project tasks.
2. Determine project task dependencies.
   a. For each pair of tasks (A+B), must task A complete before task B starts, or can both tasks execute in parallel?
   b. Draw dependency between tasks.
3. Add effort estimates for each task (uninterrupted time).
4. Add resources to each task (people, equipment, resource assumptions).
5. Add resource availability, i.e.,
   a. Planned percentage each resource will be allocated.
   b. The dates each resource is available.
6. Overlay desired project completion deadline. If the deadline is impossible:
   a. What features fit within the deadline? Is this a satisfactory list?
   b. What options are available to achieve deadline (e.g., make/buy tradeoffs, adding resources to the Critical Path, simplifying features, subcontracting work out, reusing existing code).
   c. What features should be demoted for later release?
7. Present data, schedule options and risks to management and the customer. Agree on a schedule that has acceptable customer satisfaction and acceptable risk of failure.

Figure 2. A schedule creation process.

When do you stop defining this process? When your goal has been achieved (e.g., scoping the project) and your problem solved (e.g., avoiding over-commitment). Refine the document further when it no longer meets the need.

Don't have separate audit checklists that repeat the original process. Use the original process as the checklist

If you have a process assurance function that audits projects for process compliance, use the process descriptions that the projects use; don't write a separate audit checklist. Write processes (for example, estimation, schedule creation and change control) in a style that can be used for both project and audit purposes. It might be necessary to provide auditors with some additional guidance in conducting the audit and reporting the results, but it is unnecessary to duplicate the same process information in a different format.

Consider one representation

Write processes using one representation. For example, if you are creating a process for risk management, it would be redundant to have one file of presentation slides, the same process formatted using a word processor, a version in html for browsing, and the same information again using a flow diagramming tool.

Instead, determine how the process document will be used (e.g., online use by developers during project execution, or in a classroom setting with 100 people being trained). Then consider one representation that can suit all needs. For example, a presentation slide format can be printed for reading, e-mailed for sharing, presented for teaching and uploaded for browsing.

Always consider one page (small) for each process or sub-process

There are approximately 60 lines on a page and 10 words per line. That is quite a lot of information. So consider keeping process documentation to one or two pages (at least at the beginning).

How do you keep processes to one or two pages? By limiting how much detail you allow yourself to write. Unless you plan on writing forever, you have to put some limit on the document, so start with one page. When you are tempted to add more explanation and detail, refine what you have defined, don't necessarily add more sections.

A "Documented Procedure" can be the instructions embedded in a work product template

Organizations using process improvement frameworks, such as the SEI CMM and ISO9001, might be tempted to write procedures "because the framework states that they are needed." Procedure creation is often followed by creating a template to assist the procedure user (for example, a template for an SCM, SQA or project plan). Creating both a procedure and template can lead to redundancy.

An alternative approach is to imbed the instructions for completing a template in the template itself. The procedure and the template are the same document. For example, the practice in the CMM, "Create an SCM project plan according to a documented procedure," can be implemented by developing a lightweight template with imbedded instructions for use (see Figure 3).

SCM Plan Template	Instructions
Step 1: List Configuration Items - x, y, z	~~~~~~~~~~~~~~~~~~~ ~~~~~~~~~~~~~~~~~~~~~ ~~~~~~~~~~~~
Step 2: Establish File Naming Conventions - File-x<n>.doc	~~~~~~~~~~~~~~~~~~~ ~~~~~~~~~~~~~~~~~~~~~ ~~~~~~~~~~~~
Step 3. Establish Baseline File Structure ~~~~~~~~~~~~~~~~	~~~~~~~~~~~~~~~~~~~ ~~~~~~~~~~~~~~~~~~~~~ ~~~~~~~~~~~~

Figure 3. Procedure for creating an SCM plan combined with an SCM plan template.

Knowing when you are in trouble

An organization is in "document trouble" when project team members create documents that have little use or value. This can occur when either the team members are unclear on a document's purpose or when a document is created to satisfy the needs of an external auditor or assessor.

In the first scenario, a committee is typically formed to define a specific phase of the software life cycle. The template is the committee's deliverable. The template is successfully used on a few projects and is then made standard operating procedure. When the template contains more sections than it needs to, and when the larger audience is not trained in the use and purpose of the template, too many project teams "fill out the template," because "they have to," with redundant information. At this point, the resulting document can be viewed as unnecessary documentation, particularly when the information is not regularly used to manage the project.

In the second scenario, project team members believe they have to create additional documentation to prove to an external auditor or assessor that they are managing their project correctly. Memos capturing meeting discussions, Statements of Work documents summarizing product requirements, and design documents that are created after the code has shipped, are produced to "pass the audit." Documentation is viewed by the team as an activity unrelated to building the product. It's a "keep management happy" tax.

In this latter case, the cause can be due to poorly trained auditors who look for "paperwork" but don't really understand the fundamental practice that is desired of the project teams. For example, the auditor looks for a Statement of Work document even though a detailed set of requirements exist covering the same information, or minutes of meetings are examined even though the project is six months behind and none of the corrective actions during those six months have been implemented. Here, the auditor needs education and the documents required of the team need tailoring.

"Pleasing an auditor," can also occur when the project team members have not analyzed why an engineering or management practice (and its associated document) is required and how they could benefit from it. In such cases, the team "reacts" to the process requirement without understanding why the requirement is there. Here, the team needs training on the purpose and correct use of each specific document.

If the project is performing the required practices correctly, and if the required practices have been well thought out and tested, the natural documents produced should be essential for team operation and be adequate for any auditor or assessor to see how the project is being managed. The goal is "no extra paper work."

Summary

Software development is not about documentation. Software development is about creating software solutions that help customers perform their work. Process improvement is not about documentation. Process improvement is about fixing chronic problems in the organization and capturing the solutions for reuse and refinement on the next project.

Overkill and overweight documentation is often the result of poor clarity of purpose and inadequate understanding of how each document should be used.

When documents are written with a clear business goal and need in mind, they become important and useful.

Practical Solutions for your Software Development Challenges

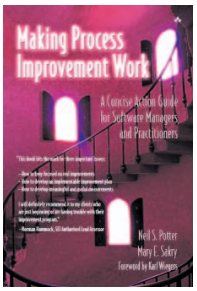
- ❑ **Understand customer needs. Clarify product requirements early.**
  In this workshop, IN SEARCH OF EXCELLENT REQUIREMENTS, software engineers, managers, requirements analysts and user representatives learn how to gather, document, analyze and manage customer requirements for software applications.
- ❑ **Decrease product development time-to-market.**
  In this workshop, ACCELERATING PRODUCT DEVELOPMENT FOR SMALL SOFTWARE PROJECTS THROUGH CYCLE TIME REDUCTION, project managers and their teams learn how to accelerate delivery through specialized schedule optimization techniques.
- ❑ **Manage projects effectively. Meet project deadlines and reduce risks.**
  In this three-day SOFTWARE PROJECT PLANNING AND MANAGEMENT workshop, project managers and their teams learn how to meet deadlines through better estimation, reduce surprises using risk management, schedule work for better optimization, understand and negotiate project trade-offs, and track progress.
- ❑ **Meet project deadlines. Scope and estimate the project work.**
  This one-day SOFTWARE ESTIMATION workshop (a subset of Software Project Planning and Management) helps teams develop more accurate estimates.
- ❑ **Avoid schedule delays caused by needless product rework. Find defects rapidly.**
  This two-day INSPECTION (PEER REVIEWS) workshop teaches teams to efficiently find defects in code and documentation. (Includes moderator skills.)
- ❑ **Hands-on SEI CMM/CMMI. Perform a mini-CMM gap-analysis.**
  The following workshops are available:
  - ❑ SEI LEVEL 2 (one day), SEI LEVEL 3 (two days), SEI LEVEL 4 (one day).
  - ❑ SEI CMMI—Overview of CMMI-v1.1 (one half-day presentation).
- ❑ **Identify critical changes to improve organizational results. Benchmark against the CMM.**
  A SOFTWARE PROCESS ASSESSMENT examines your organization's software practices and generates a focused list of the critical areas for improvement. Our SEI authorized Lead Assessors conduct customized CMM-based appraisals.
- ❑ **Goal/problem-based improvement.**
  This two-day MAKING PROCESS IMPROVEMENT WORK workshop provides a systematic approach for organizations to improve their development capability. It includes: getting management support, focusing the organization on the critical issues, planning the improvement and effecting change.
- ❑ **Tailored assistance. Dedicated phone-based assistance.**
  This service consists of customized education and coaching on your specific problems (e.g., meeting deadlines, quality and cultural change.)
- ❑ **Audio cassettes:**
  "The Role and Focus of a Software Engineering Process Group (SEPG)"
  "Making Change Happen—a 10-Piece Tool Box"

Detailed information on our services is available at ***www.processgroup.com.***

Contact us at ***972-418-9541*** *or* ***help@processgroup.com*** *to discuss your needs.*

Come see our book!

www.processgroup.com/tpgbook.htm



Here is the book's Table of Contents:

The Process Group

Mailing address: The Process Group
P.O. Box 700012
Dallas, TX 75370

Telephone number: 972-418-9541

Fax number: 972-618-6283

E-mail: help@processgroup.com
Web: www.processgroup.com

POST back issues are on line