Improve Your Process With Online “Good Practices”¹

Karl Wiegers
Process Impact
www.processimpact.com

Most software developers are allergic to paper. As organizations improve their software development and management processes, though, they typically find they need some additional project and process documentation. More structured development life cycles may present practitioners with new expectations about activities to perform and deliverables to create. An important part of successful process improvement is to develop a supporting infrastructure of process and project documentation.

People being asked to apply new practices will benefit from written guidelines for performing them. Process documentation may include process descriptions, procedures, checklists, forms and other work aids. Such items help project team members understand the expectations embedded in the improved approaches. They also provide a mechanism for sharing throughout the organization the best known ways to perform key activities, such as gathering requirements, estimating project effort and duration, and controlling changes to the product.

The contents and structure of a “good” document are not intuitively obvious. A collection of document templates and examples will help each new project to create its key documents more efficiently. By adopting some common templates, similar work products developed on multiple projects will have similar contents and structure. Each new project doesn’t have to invent a requirements specification or configuration management plan from scratch, perhaps omitting important sections along the way. Good examples of key work products collected from previous projects give developers a model to follow for their own project’s documents, and they may also save time through reuse of common text.

This article describes a project that was carried out at a large company to create a repository of useful software engineering project and process documents. This corporate resource is being used by a wide variety of software projects, thereby leveraging the investment made in developing the resource. I describe the contents and structure of the document collection, as well as the way we organized and ran the project that created the collection.

Motivation and Philosophy

Many departments in this company had already launched process improvement activities, and several had begun building their own collections of example work products and templates. A large central collection of examples had been assembled previously, but it was underused for several reasons. It existed only in hardcopy form, which made it hard to share the contents across the company, and it was difficult to search the repository for specific items. The contents varied in

¹ This paper was originally published in Software Development, December 1998. It is reprinted (with modifications) with permission from Software Development magazine.
quality, since no filtering or revision was done prior to including submitted items. Also, the contents addressed the needs of only a portion of the company’s development projects.

From a practical point of view, you should invent new procedures only as a last resort; instead, look for existing procedures to adopt or adapt to meet your needs. Several organizations were using the Software Capability Maturity Model® (CMM) to guide their process improvement activities, and they needed to create a number of documented procedures described in the CMM. We wished to leverage the procedures created by these organizations across the company, to make our process improvement activities more efficient.

To take best advantage of the available resources, we decided to build a corporate-wide intranet-based collection of software engineering “good practices”. This central resource would reduce the effort needed by individual departments to create software development procedures, because they would have good examples with which to begin. The collection also would house many good examples of key plans and other work products created on previous corporate projects. Following a pilot project to explore the concept and evaluate approaches, I was invited to lead the project to implement the repository.

We specifically avoided the term “best practices” for several reasons. First, it begs the question of who determines what is best and what the reference points are. Second, we fully expected that the initial collection of documents would be improved over time, as projects gained experience with developing and applying new processes and created ever-better work product examples. Thus, the definition of “best” is a moving target – how do you know when you’re there? Since “the best” can be the enemy of “the good,” we focused on building a collection of useful, but not necessarily perfect, artifacts that would still be better than the resources presently available.

We assembled a small part-time team of interested participants to carry out this project; as project leader, I was committed nearly full-time. Participants included members of a software engineering process group that supported a large product development division, a webmaster, and other individuals who had a strong interest in quality and process. Our initial strategy was to:

- identify the appropriate contents of the collection;
- devise a scheme for organizing and labeling the many documents we expected to encounter;
- assemble small volunteer teams of subject matter experts (SMEs) in various software practice domains to assist with evaluating, revising, and creating suitable documents;
- devise a suitable Web architecture for making the many documents as accessible as possible; and
- write a detailed and realistic project plan
- follow the plan to create and populate the repository, adjusting the plans as needed

A fundamental philosophy of the project was that “less is more,” in two ways. First, we wanted to include just a few documents of each type. If you have a dozen sample requirements specifications from which to choose, you’re likely to just create your own, rather than study all 12 and pick the format most suitable for your needs. Therefore, we included just a few templates and examples of each key document type. These represented approaches that were suitable for different kinds of projects, such as a small information system and a large embedded systems product.

The second “less is more” notion led us to trim actual project documents down to a manageable size that simply illustrated the structure, contents, and style of the document. If you’re looking for an example to mimic, you don’t want to wade through all of someone else’s
project-specific details to get the idea. We took considerable editorial license to modify candidate
document examples to meet our goals of size, detail, and layout.

Our Approach

We began by assembling a shopping list of the kinds of items we thought would be
valuable to include. The list combined items from the CMM, activities and deliverables defined by
the product development life cycle followed by many projects, and solid software engineering
approaches. The project team also identified candidate items we might harvest from the existing
local collections throughout the company.

The initial shopping list identified well over 200 different policies, process descriptions,
procedures, work aids, checklists, templates, and work product examples. Clearly, some of these
were more important than others, so we prioritized the list into three levels. Amazingly, about
one-third of the items ended up in each priority level, which made it much easier to focus on
doing the important things first. We realized that many of the lowest priority items would
probably never be acquired, since we did not have infinite resources for this project. The initial
version of the Web site included 90% of the high priority documents, 44% of the mediums, and
only 20% of the lows, demonstrating that we did a good job of investing our effort where it could
yield the greatest benefit.

The project team also applied sound requirements engineering practices to this project.
We began by brainstorming a list of use cases, each describing one objective a prospective user of
the Good Practices Web site might be trying to achieve. We have found the use case approach to
be valuable for other Web development projects, as they help the project team focus on the things
a user could do when she visits the site. Some of these use cases were:

- Find a software project plan template suitable for use on a large embedded software
  project that is also compliant with the pertinent IEEE standard.
- Find a procedure for managing requirements that complies with the expectations described
  in the CMM.
- Download forms to use for conducting a code inspection.
- See what training classes are available for a specific subdomain of software engineering.

While most of the use cases we identified were addressed by this project, we concluded
that others were out of scope. Some of those were addressed by providing hyperlinks from the
Good Practices Web site to other sites (such as our training catalog), and others simply were not
addressed. Carefully defining the scope of a project and evaluating whether each proposed
requirement, use case, or function is within that scope helps you control the scope creep that
overtakes so many software projects.

We also modeled the data architecture of the Web site using an entity-relationship diagram
and data dictionary. We modeled the user interface architecture using a dialog map, which shows
the various proposed Web pages and the navigation pathways among them. Finally, we developed
a series of evolutionary Web site prototypes, which were evaluated by our teams of subject matter
experts and eventually became the product. All of these requirements and design practices helped
us develop a usable and robust product that was well received by our user community of over a
thousand software developers and managers.

The Good Practices project team delegated much of the evaluation and improvement (and
even creation) of suitable documents to the subject matter expert teams. SMEs came from the
development, management, process improvement and quality engineering ranks. Without the
active participation of these volunteers, we would probably still be working on the project.
Good Practices Contents

We organized the contents of the Good Practices collection to make it easy for users to locate what they needed. We developed three searchable catalogs that listed all of the site contents. One catalog was organized by key process area of the CMM, since many departments in the company were using the CMM to guide their process improvement activities. Another catalog was based on the activities and deliverables defined for each phase of the product development life cycle being followed by several hundred developers in various organizations. We identified available procedures and checklists for the activities, and available templates and examples for the deliverables.

The third catalog was organized by software engineering practice area. Table 1 shows the hierarchy of practice areas we identified. Every document in the collection was uniquely identified by a label that consisted of the practice area to which it belonged, a document type code (e.g., work product example, procedure, template), and a sequence number.

By scouring the company, we found good examples of most document categories for which we were looking. Many of our best examples came from projects or organizations that already had a well-established software process improvement program, thereby increasing the return on investment from their improvement initiative.

To plug some holes in the collection, we selected items from a commercial source, EssentialSET from the Software Productivity Centre (www.spc.ca). EssentialSET includes over 50 document examples and templates from a hypothetical medium-sized project, covering the majority of the software engineering practice areas we defined. This is a good way to jump-start your efforts to build a Good Practices resource.

The publication process for preparing documents for installation on the intranet was quite elaborate. We decided to create a master copy of each document in Microsoft Word. Documents intended to be read-only were converted to Adobe Portable Document Format (PDF). Templates intended to be downloaded and adapted were published on the Web site in both Word and FrameMaker format. We wrote a detailed procedure for the many steps involved in the publication process, which might include OCR scanning of hardcopy documents, adding descriptions, editing for size or confidentiality, and reformatting.

Tracking the status of this project was not trivial. We had to monitor the evaluation and publication status of about 300 separate items, using a series of linked Microsoft Excel worksheets. For those candidates that were accepted, we tracked the following key milestone dates:

- accepted for inclusion
- permission to include it was received from the owner
- converted to Word
- all edits completed
- converted to its ultimate delivery format
- delivered to the Webmaster
- installed
- tested

We also tracked the numbers of documents each month that were delivered in each practice area category and at each priority level. Additionally, we tracked the time that all project participants, including subject matter experts, spent on the project every month, and the time spent on the many document conversion operations of various kinds. These metrics provided a complete and accurate picture of the project’s cost, achievements, and status at all times.
Table 1. Software Engineering and Management Practice Areas

<table>
<thead>
<tr>
<th>Practice Area</th>
<th>Subpractice Areas</th>
</tr>
</thead>
<tbody>
<tr>
<td>Design</td>
<td>Architecture Design</td>
</tr>
<tr>
<td></td>
<td>Detailed Design</td>
</tr>
<tr>
<td></td>
<td>User Interface Design</td>
</tr>
<tr>
<td>Implementation</td>
<td>Coding</td>
</tr>
<tr>
<td></td>
<td>Integration</td>
</tr>
<tr>
<td>Maintenance</td>
<td>(none)</td>
</tr>
<tr>
<td>Project Management</td>
<td>Estimation</td>
</tr>
<tr>
<td></td>
<td>Project Planning</td>
</tr>
<tr>
<td></td>
<td>Project Tracking</td>
</tr>
<tr>
<td></td>
<td>Risk Management</td>
</tr>
<tr>
<td>Requirements Engineering and</td>
<td>Requirements Gathering</td>
</tr>
<tr>
<td>Management</td>
<td>Requirements Management</td>
</tr>
<tr>
<td>Software Configuration</td>
<td>Configuration Auditing</td>
</tr>
<tr>
<td>Management</td>
<td>Configuration Control</td>
</tr>
<tr>
<td></td>
<td>Configuration Identification</td>
</tr>
<tr>
<td></td>
<td>Configuration Status Accounting</td>
</tr>
<tr>
<td>Software Quality Assurance</td>
<td>Auditing</td>
</tr>
<tr>
<td></td>
<td>Metrics</td>
</tr>
<tr>
<td></td>
<td>Peer Reviews</td>
</tr>
<tr>
<td>Software Subcontract Management</td>
<td>Accepting Subcontracted Materials</td>
</tr>
<tr>
<td></td>
<td>Proposals and Contracting</td>
</tr>
<tr>
<td></td>
<td>Tracking Subcontract Activities</td>
</tr>
<tr>
<td>Testing</td>
<td>Beta Testing</td>
</tr>
<tr>
<td></td>
<td>Integration Testing</td>
</tr>
<tr>
<td></td>
<td>System Testing</td>
</tr>
<tr>
<td></td>
<td>Unit Testing</td>
</tr>
</tbody>
</table>

**Web Architecture**

In the web page architecture we implemented, each practice area from Table 1 has a Practice Area Page. This describes the practice area in general, and provide information about, or links to, available training classes, tools, pertinent books, local subject matter experts, and other useful resources.

Each Practice Area Page lists the document categories available for that practice area. For example, the Requirements Management practice area includes document categories for software requirements specification, interface specification, requirements management procedure, requirements traceability matrix, and others. Each document category page provides links to several specific documents or templates, each of which has a short description to help the user select examples that are most pertinent to her needs and project type.
Lessons Learned

At the conclusion of the project, the team captured some lessons learned from the experience, and documented these to assist future such projects. Many lessons came out of things we did right, though others were learned because we missed the best approach the first time around. These lessons fell into three major categories:

**Doing It Right:**

- develop use cases to focus scope;
- develop multiple prototypes;
- provide prototype evaluation scripts to the users;
- document procedures for publishing documents on the Web.

**Controlling the Project:**

- plan the project carefully;
- build a shopping list of documents needed;
- set, and respect, three levels of work priority;
- track document and project status rigorously;
- review the risk list periodically;
- walk through the project work breakdown structure as a team to spot missing tasks;
- record action items and decisions made at meetings.

**Stability Over Time:**

- gain agreement from webmasters who own pages to which we link to inform us if those links change;
- test all external links periodically as part of the project maintenance plan;
- review and improve contents periodically to keep them current and useful.

Process improvement projects that rely on the volunteer labor of participating co-workers face a major risk: this commitment may not float to the top of a busy developer’s priority list. If a SME team was not getting the job done, we had no choice but to replace the leader or other members with people who would contribute effectively. We also decreed that a decision-making quorum for any meeting was whoever showed up, thereby keeping the project rolling along.

This project met most of its major milestones, and it was delivered on schedule and well under budget. During the past six months, hundreds of developers, quality professionals, and process improvement leaders downloaded more than 3,500 documents, suggesting that software teams find this to be a valuable resource.

It’s difficult to quantify the return on investment for such a project. However, this consolidated resource of software engineering Good Practices has clearly made the process and project improvement activities at this company more efficient. Look around for the most effective ways any of your project team members are performing key tasks. Then, take the time to capture their wisdom and share it with the rest of the organization through your own “Good Practices” collection.