1

Introduction and Overview

INTRODUCTION

Many companies, in their push to complete successful Level 2 Capability Maturity Model (CMM®)[1] or Capability Maturity Model Integration (CMMI®)[2] appraisals, have spent large sums of capital to develop and document their software processes. Many times, there is confusion regarding just what each process should contain in order to be defined as one that meets the basic Level 2 criteria as specified in the CMM® or CMMI®. IEEE standards can be used as tools to help with the process definition and documentation required in support of software process improvement. Many of the IEEE software engineering (SE) standards provide detailed procedure explanations, offer section-by-section guidance on building the necessary documentation, and, most importantly, they provide best-practice guidance as defined by those from the software development industry who sit on the panels of standards reviewers.

The CMM® for software (SW-CMM®) and CMMI®-SW are compendiums of software engineering practices that act as motivators for the continuous evolution of improved software engineering processes. It is the premise of this book that IEEE software engineering standards can be used to provide the basic beginning framework for this type of process improvement. IEEE software engineering standards, as a set, can be used to help companies define themselves as Level 2 organizations.

Moving an organization from the chaotic environment of free-form software development toward a more controlled and documented process can be overwhelming to those tasked to make it happen. This book specifically addresses how IEEE standards may be used to facilitate the development of internal plans and procedures in support of repeatable software engineering processes, or SW-CMM®/CMMI®-SW Level 2. It describes

[1]CMM® is registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.
[2]CMMI® is registered in the U.S. Patent and Trademark Office by Carnegie Mellon University (August 2002).

Using IEEE Software Engineering Standards to Jumpstart CMM®/CMMI® Software Process Improvement. ISBN 0-471-70925-5.

how IEEE software engineering standards can be used to help support the definition of best practices.

This book takes the CMM®/CMMI®-SW (Staged) Level 2 process representation and maps it to information supporting goals and practices found in the IEEE standards. The assumption is made that the standards are implemented as is, with no tailoring. This provides the reader with information regarding the value added by using the IEEE standards to implement and define software process. The identification of the strengths and weaknesses of these standards is a by-product of this comparison.

The CMM® and CMMI® do not tell the user "how" to satisfy their KPA criteria. The CMM® and CMMI® are descriptive. They do not describe how to accomplish their goals but describe the criteria that the end results should support. IEEE standards are prescriptive. These standards describe how to full fill the requirements associated with effective software project management.

It is often hard to separate the details associated with software development from the practices required to manage the effort. Simply handing the CMM®/CMMI® to a project leader or manager provides them with a description of an end result. Pairing this with IEEE standards provides them with a way to work toward this desired end. IEEE standards do not offer a "cookie cutter" approach to software management; rather, they support the definition of the management processes in use by describing what is required.

For organizations that do not wish to pursue CMM®/CMMI® Level 2 accreditation, this book will show how the application of IEEE standards, and their use as reference material, can facilitate the development of sound software engineering practices. This book is geared for the CMM®/CMMI® novice, the project manager, and practitioner who wants a one-stop source—a helpful document that provides the details and implementation support required when targeting CMM®/CMMI® implementation with the aid of IEEE software engineering standards.

What Are the CMM® and CMMI®?

The CMMI® (and in a more limited sense, the CMM®) are process frameworks. They:

- Contain the essential elements of effective processes for one or more disciplines
- Contain a framework that provides the ability to generate multiple models and associated training and assessment materials. These models may represent
  - Software and systems engineering
  - Integrated product and process development
  - New disciplines
  - Combinations of disciplines
- Provide guidance to use when developing processes

What the CMM® and CMMI® Are Not

The CMM® and CMMI® models are not processes or process descriptions. Actual processes depend on

- Application domain(s)
- Organization structure

- Organization size
- Organization culture
- Customer requirements or constraints

What are Standards?

Standards are consensus-based documents that codify best practice. Consensus-based standards have seven essential attributes that aid in process engineering. They

- Represent the collected experience of others who have been down the same road
- Tell in detail what it means to perform a certain activity
- Can be attached to or referenced by contracts
- Help to assure that two parties attach the same meaning to an engineering activity
- Increase professional discipline
- Protect the business and the buyer
- Improve the product

IEEE SOFTWARE ENGINEERING STANDARDS

IEEE software engineering standards provide a framework for documenting software engineering activities. The "soft structure" of the standards set lends itself well to the instantiation of CMM® and CMMI®-SW (Staged) Level 2 KPAs. The structure of the IEEE software engineering standards set provides for tailoring. Each standard describes recommended best practices detailing required activities. These standards documents provide a common basis for documenting organizationally unique software process activities.

Motivation for IEEE Standards

When trying to understand exactly what the IEEE software engineering standards collection is, and what this body of work represents, the following statement (taken from the Synopses of Standards section in the *IEEE Standards Collection, Software Engineering,* 1994 Edition [40]) summarizes it best:

> The main motivation behind the creation of these IEEE standards has been to provide recommendations reflecting the state-of-practice in development and maintenance of software. For those who are new to software engineering, these standards are an invaluable source of carefully considered advice, brewed in the caldron of a consensus process of professional discussion and debate. For those who are on the cutting edge of the field, these standards serve as a baseline against which advances can be communicated and evaluated.

IEEE software engineering standards attempt to capture and distill industry best practices. They consolidate existing technology, establishing a firm foundation for introducing newer technology. They increase the professional discipline though the standardization of evolving technologies and methodologies. The application of IEEE software engineering standards helps to ensure a higher quality product. Application of these stan-

dards, while keeping the CMM® or CMMI® in mind, helps to ensure that the production of a higher quality product is consistently reproduced. Applying IEEE software engineering standards and the CMM®/CMMI® processes and procedures together can help users define their software development processes while developing software products (see Table 1-1).

Categories of IEEE Standards

As described in the IEEE *Software Engineering Standards Collection* [49], all standards are prescriptive in nature, containing requirements that must be satisfied. These levels of prescription may be used to categorize the IEEE software engineering standards collection:

1. *Terminology* standards provide definitions and unifying concepts for a collection of standards. In many cases, they do not include any explicit requirements, only the implicit demands of applying a uniform terminology.
2. *Collection guides* do not provide requirements—only information. A collection guide surveys a group of related standards and provides advice to users on how suitable standards may be selected for their use.
3. *Principle standards* provide high-level requirements that might be satisfied in a wide variety of ways. They emphasize goals rather than specific means for achieving the goals.
4. *Element standards* are the most familiar form. They contain requirements more detailed than those of principle standards and prescribe a particular approach to achieving the goals prescribed in a principle standard.
5. *Application guides* emphasize recommendations and guidance. They provide advice on how element standards may be implemented in particular situations.
6. *Technique standards* are the most detailed and prescriptive. They generally describe procedures rather than processes. They provide very specific requirements, presumably for those cases in which small deviations might have large consequences.

Table 1-1 What do IEEE Standards do? Some examples

Standard	Function
IEEE 982.1 Measures for Reliable Software	Specifies techniques to develop software faster, cheaper, better.
IEEE 1008 Unit Testing	Describes "best practices."
IEEE 1061 SW Quality Metrics	Provides consensus validity for techniques that cannot be scientifically validated.
IEEE/EIA 12207 SW Life Cycle Processes	Provides a framework for communication between buyer and seller.
IEEE 1028 SW Reviews	Gives succinct, precise names to concepts that are otherwise fuzzy, complex, detailed and multidimensional [78].

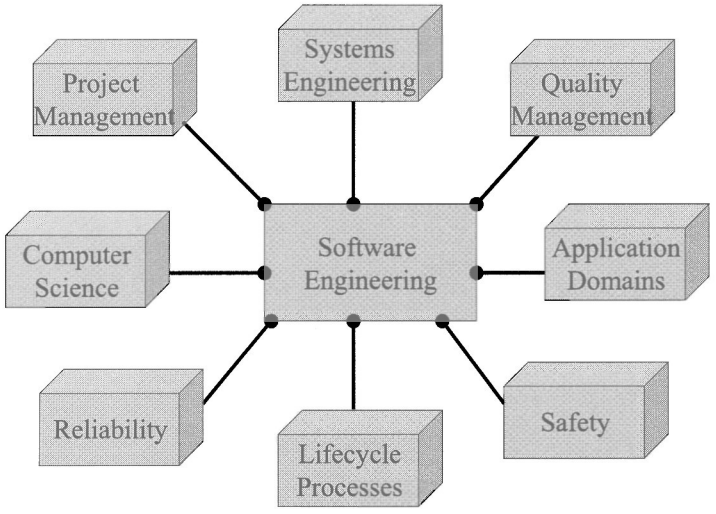

Figure 1-1 IEEE S2ESC standards support of software engineering.

IEEE Standards Development

The focal point for the development and adoption of software engineering standards is the Software and Systems Engineering Standards Committee (S2ESC), a standards sponsor under the IEEE Computer Society Standards Association (IEEE-SA). S2ESC is an organization comprised of over 2000 volunteers who are committed to providing an integrated set of software and systems engineering standards that support the practice of engineering software and systems containing software (Figure 1-1).

The purposes of the S2ESC as defined in the S2ESC Charter[3] are:

1. Codify the norms of professional software engineering practices into standards.
2. Promote use of software engineering standards among clients, practitioners, and educators.
3. Harmonize national and international software engineering standards development.
4. Promote the discipline and professionalization of software engineering.
5. Promote coordination with other IEEE initiatives.

All IEEE software engineering standards are either developed by S2ESC-sponsored working groups or adopted from other standards development organizations. In either case, each standard is submitted to the S2ESC Management Board for a readiness review prior to balloting. Following the readiness review, a balloting pool is formed and the standard is then put forward through a rigorous balloting process. Each negative ballot must be addressed prior to the acceptance of the standard for publication. A cycle of comment resolution, revision, and recirculation continues until consensus is achieved among the balloting group. IEEE defines consensus as 75% approval. Following initial publication, each IEEE standard is subjected to review at least every five years for revision or reaffirmation.

[3]IEEE S2ESC Charter Statement [42].