AN IEEE SOFTWARE LIFE CYCLE PROCESSES STANDARDIZATION EFFORT

D. J. Schultz
Computer Sciences Corporation
15245 Shady Grove Rd.
Rockville, MD 20850
(301)921-3090

A.F. Godin
Bell Communications Research
PYA 1P-208
6 Corporate Place
Piscataway, N.J. 08854
(908) 699-5523

Abstract

This paper presents an overview of the effort associated with developing an IEEE standard. It summarizes the history of the project, including the challenges of getting started and some of the major concerns and objectives which had to be addressed as the standard developed and evolved. A technical overview of the standard, and a summary of plans for future related activities, are also included.

1. Introduction

The IEEE Standard for Developing Software Life Cycle Processes is coming to a successful conclusion. After many years of intensive effort, the working group assigned to this project has produced a document and taken it through the balloting stage. This paper provides a summary of the history of the effort, a technical overview of the balloted standard, and a discussion of lessons learned during the process.

2. History

This project was initiated in August, 1984, by a group of people who had been involved in previous IEEE standards efforts, and were thus intimately familiar with the standards process. They were concerned that virtually all the IEEE Standards then in existence addressed software products (QA Plans, CM Plans, Requirements Specifications, etc.). The working group felt that a standard addressing software processes was greatly needed.

3. Background

The IEEE-STD-1074, "Developing Software Life Cycle Processes", was considered by the IEEE working group as an instrumental standard for developing and maintaining software in the future. Acknowledging the fact that creating this standard would be a great challenge, the working group was able to overcome adversity and produce a finalized standard which is in the process of going through recirculation. This standard, once approved, will be applicable to any software organization: large or small, supplier or user, providing critical or non-critical applications, with thousands of employees or a small software house with one hundred or fewer employees.

Realizing the impact this standard will have throughout the software industry, it was generated with both a significant amount of interest from around the world along with a myriad of different opinions on how to write it. Under all this attention and adversity in developing...
IEEE-STD-1074, the initial working group decided the benefit of having such a standard would greatly outdistance any problems in developing it. It was like attempting to climb Mount Everest for the first time: the task of climbing the mountain was significantly diminished by the achievement of reaching the summit of the mountain. So too the goal of creating a standard which would benefit all who choose to adopt it was worth all the adversity and bickering that the working group experienced.

4. Scope and Purpose

The working group began by agreeing upon a draft scope and outline for the standard. The initial statement of the scope was as follows:

The scope of this standard is to define the processes which comprise the software life cycle, and to describe the activities required to develop or maintain a software product in accordance with existing IEEE standards.

The group decided not to restrict the standard to the software development phase, but to include the phases of concept exploration, operation, maintenance, and retirement. It also decided not to specify a particular software life cycle model or project organization, but rather to produce a generic standard which could be consistent with any software life cycle model and project organization. Finally, it decided not to impose a time-ordering in the standard, i.e., not to require that any particular activity be completed before another could be initiated. The choice of life cycle model, project organization, and time-ordering of activities was left to the discretion of the user of the standard.

5. Organization of the Standard

The initial list of life cycle processes to be addressed consisted of:

1. Concept Exploration
2. Requirements
3. Design
4. Implementation
5. Test
6. Installation and Checkout
7. Operation and Maintenance
8. Retirement.

The working group decided that a distinct chapter of the standard would be devoted to each process. Each chapter would contain a short overview of the process, followed by a detailed description of the activities which comprise the process.

The discussion of these processes was to be preceded by a chapter entitled "Software Life Cycle Model," which would set forth the requirement for the user to select a life-cycle model. The user would be required to map the activities in the standard against the chosen life cycle model to develop a time-ordered life cycle.

In the initial outline of the standard, the following topics were deliberately not addressed:

Software Quality Engineering
Software Quality Assurance
Software Configuration Management
Verification and Validation
Project Management

It was initially intended that these topics would all be worked into the discussion of each life cycle phase. Eventually, however, the group realized that these topics deserved to be addressed as processes in their own right. The definition of a process was expanded beyond a life cycle phase, and these topics were addressed as processes which
spanned the entire life cycle. (Software Quality Engineering and Software Quality Assurance were eventually combined into a single process entitled Software Quality Management).

6. Major Concerns and Objectives

The working group operated by preparing successive drafts of the standard, distributing them to a wider interested community, and soliciting comments. During each meeting, the comments which had been received on the previous draft were reviewed and addressed. A form for submitting comments was developed, and commentors were required to use this form and to submit each comment on a separate page. This approach was found to be invaluable in that it enabled the working group to deal with each comment as a separate entity, record the disposition of each comment, and set up a configuration control system to ensure that every comment was addressed and resolved. When a comment was rejected, the reason for rejection was recorded on the comment form, in case of any subsequent dispute.

As the working group grew, it became necessary to divide into subgroups, each of which dealt with comments on a particular subset of the total processes. At first, each subgroup addressed a different set of processes at each successive meeting. In this way, the members of each subgroup became familiar with the entire document. Later, the assignment of processes to each subgroup was made permanent, so that the group developed a cadre of specialists who were intimately familiar with the unique problems and concerns of their assigned processes.

The structure and organization of the standard was initially quite fluid, undergoing significant changes as new members joined the working group and injected their ideas. Eventually, a separate "structure committee" was established, which addressed structural issues off-line and proposed a structure and organization which would meet the various concerns raised during the working group meetings. A second committee, called a "coordinating committee," was also set up, to protect the structure of the document and assist the chair in guiding the development of the standard. Finally, a separate subgroup was established, to deal with general comments which affected the standard as a whole, and to resolve differences among the other subgroups.

The group recognized at the first meeting that recruitment of new members would be an important objective. Each of the participants at the first few meetings was urged to attempt to recruit other members. In addition, the chair prepared a short description of the project, with an appeal for new members; this description was sent to several software journals (IEEE Computer, Communications of the ACM, etc.).

The group also recognized the need for maintaining a log of major technical issues which had been identified, and the corresponding resolutions. The log turned out to be a very useful tool and the group had frequent occasion, over the life of the project, to refer to it. At the second meeting, for example, three key issues were brought up:

- Is the working group committed to using the life cycle presented in IEEE Standard 729?
- How should the group address the iterative or nested structure of the life cycle?
- How should the group deal with the topics of prototyping and modeling?

The group decided that the standard would define the processes to be followed, but that the choice of life cycle would be left to the user. Iteration or nesting within a life
cycle would be addressed without specifying a specific time-ordering of the processes, or of their component activities. Prototyping and modeling were felt to be important approaches to software development which should be addressed within the standard. These three key decisions shaped our approach to the standard at our second meeting, and strongly influenced the structure of the final document.

7. Technical Summary of the Standard

This document is a process standard. It does not specify the format or contents of any software development products, but rather defines the processes that are necessary for software development and maintenance. It also provides the set of activities that constitute these processes, together with their associated input and output information.

The working group soon discovered that it was the activities, rather than the processes, which constituted the natural focus of the standard. All the "meat" of the standard—input information, description, and output information—is meaningful only at the activity level. The working group soon found it necessary, therefore, to focus on the activities, while treating the processes simply as convenient groupings of activities.

The activities are linked together by means of their required input and output information. The input information for an activity defines all the entry information which that activity requires in order to perform its function. The output information defines the exit information which the activity produces. The actual definition of each activity consists of the required input information, a description of what takes place during execution of the activity, and the resulting output information. The sources and destinations for the input and output information are also specified. (The working group even attempted to construct a set of data flow diagrams showing the flow of input and output information, but eventually gave it up as too complex and unwieldy.)

The list of processes, and their constituent activities, grew and evolved during the preparation of the standard. As it now reads, the standard addresses the following list of processes and activities:

1. Introduction
2. Software Life Cycle Model Process
3. Project Management Processes
   3.1 Project Initiation Process
   3.2 Project Monitoring and Control Process
   3.3 Software Quality Management Process
4. Pre-Development Processes
   4.1 Concept Exploration Process
   4.2 System Allocation Process
5. Development Processes
   5.1 Requirement Process
   5.2 Design Process
   5.3 Implementation Process
6. Post-Development Processes
   6.1 Installation Process
   6.2 Operation and Support Process
   6.3 Maintenance Process
   6.4 Retirement Process
7. Integral Processes
   7.1 Verification and Validation Process
   7.2 Software Configuration Management Process
   7.3 Documentation Development Process
   7.4 Training Process

Appendices
Index

The six groups of processes comprise the entire software life cycle (SLC). However, it is important to realize that
IEEE-STD-1074 is not a time-ordered document. Although one activity may seem, intuitively, to precede another, this need not necessarily be so. The activity described on page 20 of the standard, for example, need not occur before that discussed on page 21. It may occur after it, or simultaneously, depending upon the constraints of the chosen SLC. In some environments, certain activities may even be repeated so that one activity might occur both before and after another one. The organization using this standard is responsible for placing the various activities into a proper time sequence.

The processes in the last group, called the "Integral Processes," are treated in yet a different manner. These processes do not in themselves produce usable software products, but instead support the processes in the other groups. They are therefore invoked in a manner similar to a subroutine call. When an activity in, say, the Implementation Process requires an integral process (Configuration Management, perhaps), it invokes that process directly, beginning at a specified activity within the integral process. The invoking process passes the required information (source or object code, for example), to the invoked process (Configuration Management, in this example). The invoked process then performs its function (establishing properly configured, controlled libraries containing the source or object code). Once the integral process has completed its work, it returns its output products to the process which invoked it, and then passes control back to that process (Implementation), which continues working.

This standard is independent of the formal structure of the organization which is developing or maintaining the software. When a function (say, Configuration Management, to continue the example) is identified, there is no implication that an organization of people with a corresponding name must exist to perform this function. (The user of IEEE-STD-1074 need not have a "Configuration Management Department" to be compliant with the standard. The user can assign this function to the project manager, the test team, or the quality assurance group, if so desired. The only requirement is that the function of Configuration Management be performed.)

The standard is also independent of the software development methodology which may be employed. No specific requirements analysis, design, code, or test methods or techniques are required. These are all treated as environmental factors. The standard can therefore be used in any application environment, or by any software organization, without regard for the particular methods or techniques employed.

8. Use of the Standard

Before beginning a project which will use this standard, the user must review the list of activities to verify that they all apply to the specific project in question. The user must then organize these activities into a time sequence appropriate to the project. To perform this time sequencing, a software life cycle model (SLCM) is required.

The standard does not mandate the use of a specific SLCM. Rather, it requires the user to select or develop one. Having done this, the user is required to map the activities of this standard to the selected (or developed) SLCM. This means identifying precisely where in the SLC each activity will be performed, and how it will sequence with the related activities. (Several examples of such mappings, using rather different SLCMs, are included in Appendix A of the
Another concept to which the user must pay close attention is the threading of input and output information among the activities. Information is woven throughout the SLC, entering and exiting activities which frequently modify or alter it. An activity may, for example, group several items of input information into a single output information item, or turn a piece of input information into one or more pieces of output information. By studying the flow of information through the processes and activities, the user can trace the evolution and development of initial needs into requirements, design, code, and other software products.

How can a user verify compliance with IEEE-STD-1074? The careful reader will observe that each activity in the standard is categorized as either mandatory or "If Applicable". Compliance with the standard is defined as the performance of all activities specified as mandatory. This means that all input information for the mandatory activities must be processed, and all output information from them must be generated. (A customer, or an auditor, can thus use the input and output information as the basis for determining compliance with IEEE-STD-1074.) Note, incidentally, that the standard does not require the production of documents bearing particular names. The packaging of the information into documents, and the assignment of names to those documents, should be done through negotiation between the user and the customer.

Finally, the user should be aware that IEEE-STD-1074 does not exist in a vacuum. Many of the key topics discussed in this standard are elaborated in greater detail in other IEEE standards, recommended practices, and guides. The standard contains a Bibliography, in which these other related IEEE documents are listed. The user should not attempt to employ IEEE-STD-1074 without consulting these related standards, upon which it depends.

9. Future Initiatives

After IEEE-STD-1074 was completed in 1989, the group presented it to the Software Engineering Standards Subcommittee (SESS) for the purpose of acquiring industry acceptance through a process called balloting. With a significant interest in place a large ballot return was expected. One hundred and fifty-three (153) ballots were submitted by industry in response to the ballot. There were ninety-nine (99) affirmative and eighteen (18) negative ballots, i.e., eighteen balloters did not accept the standard as written. These negative ballots had to be reconciled before the IEEE would approve IEEE-STD-1074 as a new IEEE standard. Subsequent to the ballot, the working group held two meetings to reconcile negative ballots and positive comments, and modified IEEE-STD-1074 accordingly.

The working group which is presently completing IEEE-STD-1074 has initiated a separate, related project to write an IEEE Guide to Developing Software Life Cycle Processes. This Guide will elaborate on the requirements of the Standard and provide suggestions for possible ways to develop or maintain software in conformance with the Standard. The working group is also trying to harmonize IEEE-STD-1074 with related DOD and international standards.

10. Summary: Key Points to Remember

There are several key points to remember from this paper. There is no specific software life cycle model required for
IEEE-STD-1074. Any software life cycle model can have IEEE-STD-1074 mapped onto it. The key point here is that a software life cycle model must be implemented for IEEE-STD-1074 to work. IEEE-STD-1074 was designed to fit any existing software life cycle. Similarly, no software development methodology is presumed or implied. Again, it is the responsibility of the user to have a software methodology.

As stated above, this document is process-oriented and therefore the activities are neither time-dependent nor organization-dependent. The standard indicates the input required for each activity, the tasks to be performed, and the outputs to be generated. The standard does not dictate who should perform the activity or when the activity should be performed.

This paper also stresses that information does not necessarily mean documentation. The packaging of information into a particular set of documents, bearing specific titles, is a decision to be made when planning the management of the software project.

11. Acknowledgements

The authors would like to acknowledge the efforts of the following people who have significantly contributed to the writing of IEEE-STD-1074: David W. Burnett, Jean A. Gilmore, Joseph J. Guidos, John W. Horch, Lynn D. Ihlenfeldt, Dennis E. Nickle, and Robert W. Shillato.

The authors would also like to thank the following people for reviewing and contributing to this paper: John W. Horch, Lynn D. Ihlenfeldt, Dennis E. Nickle, and Colleen O. Singer.