THE PROCESS OF PRODUCING CORPORATE STANDARDS

Håkan Ljungkvist
Philips Information Systems
Business Centre Financial Industry
S-162 86 Vällingby
SWEDEN

ABSTRACT

This paper describes a corporate process in which a set of Software Engineering Standards were produced by and integrated into a multi-national organization. The paper is intended as an input to people who are about to conduct similar activities in other organizations, as well as for the members of the working groups producing and updating international standards.

The experiences gained in this standardization project will be highlighted. The rationale for using IEEE Standards as a starting-point and how these standards influenced the result and contributed to the success of the project will be described.

BACKGROUND

In the spring of 1986, Philips Data Systems (PDS) develops hardware and software for UNIX and PC-based systems. The software encompasses an extensive range of general-purpose system software, e.g., kernel improvements, device drivers, communication and networking packages, office automation products, and 4GL application development tools. The development is distributed to five Development Centers - Canada, France, Germany, Holland, and Sweden - and involves more than 400 software developers.

At this time, an activity is initiated by corporate development management to define a project for production and introduction of a set of Software Engineering Standards for the international software development. The software development process needed to become more visible and measurable for the software engineers and their managers. Existing standards in the organization are not state-of-the-art, are used to a varying extent, and have a different interpretation in each Development Center and project.

THE PDS SOFTWARE ENGINEERING STANDARDS PROJECT

Summary

The major objective of the Philips Data Systems standards project was to specify and establish a uniform approach to the documentation of software requirements, designs, and tests to be used by all software development projects within the world-wide organization. The project plan was accepted by the managers of the five Development Centers in December 1986. The IEEE Software Engineering Standards where specifically agreed upon as a starting-point for the technical contents. About a year later, in January 1988, the new standards were formally agreed upon.

Project Approach

Any new standard must be carefully fitted into the current framework of corporate culture, practices, standards, and needs. The approach chosen for this project was carefully defined and agreed upon before the project started. The approach chosen for this the project can be summarized as follows:

- The key starting-point for the project was a platform of management commitment and involvement; without this there would be no project.
- It was agreed to use the state-of-the-art standards developed by IEEE, rather than the existing set of standards; thereby, the project got a “flying start” and avoided endless discussions regarding which changes were acceptable to make to the existing standards.
- The scope of each standard was limited, well-defined, and excluded any organizational issues; if organization was to be reflected, it would have been impossible to achieve one set of standards without
local adaptation.

- There was an explicit priority setting stating that completion date was more important than the extensive completion of all defined deliverables; the major deliverables (standards) were thus identified, and minor ones were included but could be excluded, if delivery dates were at risk.

- Sufficient resources were allocated; funding was raised both from corporate budgets and local ones.

### Organization

The Project Team consisted of one full-time project manager plus consultants. A "Development Policy Council", chaired by the corporate development manager and consisting of the managers of the Development Centers, acted as steering board for the project. This provided the project manager with a direct communication channel to all development managers.

Each Development Center appointed a "Contact Person". The Contact Persons were authorized to represent their organization in the extensive discussions and reviewing, which took place in an existing Working Group for coordination of software engineering issues.

### Reviewing

The reviewing scheme encompassed three types of reviews:

- Internal reviews. All standards were extensively discussed and reviewed in the Working Group before any external reviewing took place.

- Preliminary external reviews. The external reviews were conducted at each Development Center. Based upon the review minutes, the project team made a proposal, which was presented at a Working Group meeting for consensus and approval for pilot usage.

- Final external reviews. After pilot usage in most Development Centers, a new review round was set up in order to arrive at a final standards proposal. This version was formally authorized by the Development Policy Council.

### Resources

The standards production effort includes time spent by the Project Team, developing the standards proposals, and the time spent by the Contact Persons, both in the Working Group and in local Development Center activities. All other Development Center involvement (reviews, pilots, and training) is not included; see Figure 1.

In the course of the project, travelling included attendance to seven Working Group meetings, four Development Policy Council meetings, and project manager visits to each of the Development Centers during project initiation and external reviewing of the standards.

### Project History

The following chronological list highlights the significant events during the project:

- Mar 1986 The Development Policy Council (DPC) decides to initiate a Software Engineering Standards project.
- Sep 1986 The DPC approves the approach, scope, and setup of the project.
- Dec 1986 START OF THE PROJECT. The project plan is approved by all managers of the five Development Centers.
- May 1987 Preliminary external reviews take place in each Development Center.
- Jun 1987 The standards for software life cycle, requirements and test specifications are approved for use in pilot projects.
- Nov 1987 Pilot usage is evaluated and provides input for the final reviews of the standards.
- Dec 1987 The Working Group agrees upon the contents of the standards for software life cycle, requirements and test specifications.

### MANPOWER EFFORT

<table>
<thead>
<tr>
<th>RESOURCE</th>
<th>ACTIVITY</th>
<th>EFFORT IN MAN-MONTHS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project Team</td>
<td>project management</td>
<td>7 9</td>
</tr>
<tr>
<td></td>
<td>standards development</td>
<td>10 12</td>
</tr>
<tr>
<td>Contact Persons</td>
<td>Working Group meetings, reviews, local introduction</td>
<td>10 15</td>
</tr>
<tr>
<td>TOTAL:</td>
<td></td>
<td>27 36</td>
</tr>
</tbody>
</table>

Figure 1. MANPOWER EFFORT
Jan 1988  The completed standards are submitted to the DPC for formal acceptance.

Mar 1988  The standard for design specifications is agreed upon within the Working Group for pilot usage.

Jun 1988  All deliverables, except the design specification standard, are completed (refer to Figure 2 below). Local introduction starts.

Sep 1988  END OF THE PROJECT. A new project takes over the support responsibilities and the completion of the design specification standard.

Feb 1989  The standard for design specifications is completed and submitted to the DPC for formal acceptance.

Deliverables

The resulting set of Software Engineering Standards were organized as an open-ended sequence with a limited scope for each standard. A life cycle standard was added to serve as a framework for the documentation standards as well as for the software development process.

The released deliverables [PDS-SES] of the project included standards and supporting materials as listed in Figure 2, where also the relations between the deliverables and IEEE standards are depicted.

THE PDS SOFTWARE ENGINEERING STANDARDS

The Software Life Cycle Standard

The software life cycle (SLC) standard contains a number of interesting features, such as an iterative approach providing the same view on the life cycle regardless of level, a test development cycle, and an iterative approach to the various specifications, which extends their scope to levels above software product.

The Basic Software Life Cycle

The basic SLC is similar to the traditional waterfall model; see Figure 3.

The following remarks concerning the properties of this SLC definition can be made:

**Deliverables Derived From**

<table>
<thead>
<tr>
<th>DELIVERABLE</th>
<th>DERIVED FROM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard for a Software Life Cycle</td>
<td>[IEEE-SET]</td>
</tr>
<tr>
<td>Standard Glossary of Software Engineering</td>
<td>[IEEE-SGS]</td>
</tr>
<tr>
<td>Standard for Requirements Specifications</td>
<td>[IEEE-SRS]</td>
</tr>
<tr>
<td>Standard for Test Specifications</td>
<td>[IEEE-STD]</td>
</tr>
<tr>
<td>Standard for Design Specifications</td>
<td>[IEEE-STD]</td>
</tr>
<tr>
<td>Tutorial on Software Life Cycle</td>
<td>[IEEE-SRS]</td>
</tr>
<tr>
<td>Tutorial on Requirements Specifications</td>
<td>[IEEE-SRS]</td>
</tr>
<tr>
<td>Educational Material</td>
<td></td>
</tr>
<tr>
<td>Quick-Reference Guide</td>
<td></td>
</tr>
</tbody>
</table>

Figure 2. PROJECT DELIVERABLES

- The Implementation phase contains detail design, coding, and module testing.
- In the Support phase, the software is in operational use and modified as necessary to correct faults. Development of functional enhancements of the software product is described by a separate SLC.
- Overlapping phases during the development of a software product are introduced and controlled by means of "nesting of SLCs" and by extending the basic SLC with "test development cycles". These mechanisms are described below.

**Nesting of Software Life Cycles**

Software life cycles may be nested in order to cope with large and complex products (systems). Each of the components defined during Architectural Design is decomposed further by the group of activities covered by a new SLC. The Implementation phase is thus described (or replaced) by a set of parallel SLCs, one per component; refer to Figure 4. The nesting continues until the functionality is fully detailed, in which case the Implementation phase activities include the actual production of the source code.
The nesting provides a mechanism to cope with a number of iterations between analysis, WHAT the software must do, and design, HOW the software is to solve the problem. Nesting of SLCs is also used to obtain correspondence between projects and management levels of the organization. Within the PDS organization, the development of a large system was split up into three levels, as depicted by Figure 5.

The Extended Software Life Cycle

For each type of testing in the software life cycle, a separate Test Development Cycle, TDC, can be identified. Each TDC consists of three phases covering test planning and definition of test requirements, test preparation, and test execution. The starting and ending phases coincide with phases of the basic SLC, whereas the Test Preparation phase in between constitutes the extension to the basic SLC.

The Test Preparation phase includes design and implementation of the test and is completed when the test specification, including all test data, and test tools are available. These test development activities are performed in parallel with the product development activities. Figure 6 shows the extensions to the basic SLC model to incorporate the TDC concept.

The relation between the product and test development parts of the extended SLC is depicted in Figure 7, where the product development cycle is drawn as a V, the left side constituting the analysis part of the development process and the right side the synthesis part.

The Standard Glossary

The glossary contains all definitions of terms which were considered necessary for the standardization effort. The 90 terms covered by the standard are influenced by [IEEE-SET], by other IEEE standards, and to a certain extent by current terminology within the organization.

The Documentation Standards

Each documentation standard is emanating from a corresponding IEEE standard (see Figure 2). In terms of conformance to the IEEE standards, the requirements specification is closest to its corresponding IEEE standard and the design specification furthest away.

A common feature of the documentation standards is the extension of the scope to more than one level in a system structure.

Requirements Specifications

The Requirements Specification occurs in the Software Life Cycle as a Software Requirements Specification, SRS. The SRS is produced in the Requirements phase. The following items deviate from the [IEEE-SRS]:

- The limited view on functions as simple input-processing-output items is extended to finite state machine type of processing.
The external interface requirements part has caused much discussion and is the weakest point of the standard. What should be included? With what level of detail? The answer also varies with the level of specification; see section below, "Nested SLCs and the Document Structure".

The section "Customization Requirements" is added since it is an important issue in the PDS environment.

The "user interfaces" were extended to incorporate both human and application interfaces.

The table of contents is depicted in Figure 8 and may be changed in the same manner as described in [IEEE-SRS].

Test Specifications

There are four different types of testing defined in the Software Life Cycle: Module, integration, alpha, and beta testing. Each of these tests is defined by a corresponding Test Specification. The prescribed contents of a Test Specification is shown in Figure 9.

The major changes introduced in this standard compared to the [IEEE-STD] are:

- Only the specification of tests is addressed by the standard; test plans and test reports are left outside its scope.
- The specifications of test-design, test-cases, and test-procedures are put together into one document, the Test Specification.
- A "Test Requirements Specification" chapter is added to the specification, corresponding to the first
TABLE OF CONTENTS

1. INTRODUCTION
2. TEST REQUIREMENTS SPECIFICATION
   2.1 Features to Be Tested
   2.2 Features Not to Be Tested
   2.3 Test Item Pass/Fail Criteria
3. TEST DESIGN SPECIFICATION
   3.1 Test Approach
   3.2 Global Test Environment
   3.3 Test Identification
   3.4 Intercase Dependencies
4. TEST CASE SPECIFICATION
   4.n <test case xx>
5. TEST PROCEDURE SPECIFICATION
   5.m <test procedure YYY>

Figure 9. THE TEST SPECIFICATION

phase of the Test Development Cycle.

Design Specifications

The Design Specification occurs in the Software Life Cycle as one or many Software Design Specifications, SDSs. These SDSs record the result of the design processes that are carried out during the Architectural Design and Implementation phases. The approach in [IEEE-SDD] to regard the design as a partitioning of the software into design entities with properties and relationships to each other is used in the following manner:

- The design entity attributes were divided into external attributes (function, interface, dependencies) and internal attributes (data, processing).

TABLE OF CONTENTS

1. INTRODUCTION
2. GENERAL DESCRIPTION
   2.1 Design Overview
   2.2 Design Approach
   2.3 Development Environment
3. SPECIFIC DESIGN ENTITIES
   3.n Design entity xx
   3.n.1 Function
   3.n.2 Interface
   3.n.3 Dependencies
   3.n.4 Data
   3.n.5 Processing
   3.n.6 Subordinates

Figure 10. THE DESIGN SPECIFICATION

- In case a lower level SDS exists, the internal attributes are not specified.
- The table of contents is much simplified; see Figure 10.

Nested SLCs and the Document Structure

The document structure reflects very closely the way the software is developed. In a situation where nested SLCs are employed, each level has its own set of specifications. A nested approach means that the highest level requirements specification is not complete; a number of requirement details are found in lower level requirements specifications.

In a large software system, the highest level SRS could specify functional areas required, e.g., office automation, networking, 4GL application generation tool. The architectural design identifies the design entity "office automation", which in the next level SRS is specified as groups of functions for "word-processing", "spread-sheet", etc. One of the lowest level SRSs, in this scenario, specifies in detail the functionality of the "word-processing" product.

INTEGRATION INTO EXISTING PROCESSES

The integration of the PDS Software Engineering Standards into existing processes involved two major activities:

- Standards installation; introductory training and support in the target development organizations
- Standards management; long-term support, enforcement, maintenance, and enhancements

Standards Installation

The standardization is not completed before it is introduced and accepted in its target organization. This task requires just as much calendar time as the definition of the contents of the standard. The installation process included the following actions:

- On corporate level
  - Formal statement regarding the application of the new standards; all new development must be based on the new standards
  - Modification of the standard activities and milestones used for corporate project planning and progress reporting to conform to the new
standards

- On development center level
  - Explicit detailing on how the standards will be applied and controlled in the local organization
  - Appointment of a local Standards Bearer, responsible for the support and promotion of the standards
  - Definition of a local training program

- By the standardization project team
  - Provision of educational material
  - "Commercial packaging" in terms of nice-looking binders and a Quick-Reference guide
  - Active support and review of the local installation activities

**Standards Management**

Management of the installed standards, including maintenance to "keep them alive", requires continuous attention. After the introductory marketing, the application, promotion, and follow-up of the standards must survive organizational changes, where key people in the standards production effort get new assignments and new managers are appointed.

On corporate and development center management levels, a Standards Bearer and a Reference Group provide a means to support a manager in his task to ensure the long-term existence of the standards. On corporate level, these mechanisms perform the necessary coordination and follow-up, as well as authorization of minor changes. Any extensive enhancement, on the other hand, calls for a new dedicated project.

**THE ROLE OF IEEE STANDARDS**

**Evaluation**

The impact of the IEEE Software Engineering Standards in this type of corporate standards production has proven to be beneficial. However, a number of problems were also encountered. The future role of IEEE standards in this environment would be even more important, if these problems were considered.

- Benefits
  - Existing standards, conventions, and practices in the organization will always be defended and promoted by their followers. By starting out with the IEEE standards, a "neutral" and respected platform was provided for the corporate standards implementation.
  - Given a state-of-the-art IEEE standard, the lead-time to produce the corporate standard was reduced.
  - The tutorial material in [IEEE-SRS] helped a lot for the understanding of the standard and was used as basis for a separate tutorial.

- Problems
  - The lack of supporting guides
  - The IEEE standards classification made no sense, as discussed below
  - The absence of a documentation framework
  - IEEE was referred to as a U.S. standards organization by an opponent; "Why should we follow a U.S. standard?"

**IEEE Classification of Standards**

The IEEE distinction between the standards levels "standard", "recommended practice", and "guide" was of no significance at all. A standard must make sense. Any type of statements and terminology not easily understood in the environments they are to be applied in will cause a lot of discussions, contradictory interpretations, and bad will. Therefore, adjustments need to be introduced in most cases.

In this project, the conclusion was that it is not possible to handle the enforcement of a standard stamped as "recommended". If a project manager has a good reason for not conforming to existing standards, he will be authorized to do so by his management. Therefore, all the standards were given the status "mandatory" with the rule that any deviation must be made visible by explicit statements in the project plan.

**CONCLUSIONS**

**Key Issues for a Successful Standardization Project**

The checklist in Figure 11 contains the important issues derived from the experiences in the PDS Software Engineering Standards project. These issues are believed to be essential in any size organization and should be carefully considered in order to ensure the success of a standardization project.

**Why Did This Project Run Well?**

The standardization checklist presented in Figure 11 describes fairly well the strategy for the PDS project. A number of the factors can, however, be further elaborated:

- Top management was involved from the very beginning; they, in fact, initiated the effort.
STANDARDIZATION CHECKLIST

1. Engage top management in a direct way and ensure their active support throughout the project; otherwise, forget about the whole project.

2. Obtain an early and firm commitment from all involved managers in the target organizations; there must be no "wait-and-see" attitude.

3. Ensure the active participation of representatives from the involved organizational entities; this creates both the necessary lead-time for the introduction and an important contribution of experience into the project.

4. Assemble the key people regularly for extensive discussions and reviews; actual meetings can not be replaced by telephone or mail.

5. Look for a technical starting-point which is as neutral as possible; no one can play the role of defender of existing standards.

6. Allocate full-time resources for the project; do not expect that the actual production of the deliverables can be carried out by "part-timers".

7. Allocate appropriate manpower and calendar time for the introduction in the organization; only half the battle is won when the standards are formally accepted.

Figure 11. KEY STANDARDIZATION ISSUES

- The time was right; the organization was very open to discuss new standards, whereas three years earlier it would have been impossible to conduct this standardization effort.

- The consensus process in the Working Group was of greatest importance to the result; both in terms of quality and local acceptance.

- The use of IEEE standards and the exclusion of organizational issues in the standards made it possible to quickly accomplish a version for pilot usage.

What Did Not Work out as Planned?

Not everything worked out as planned, although the overall conclusion is that the project achieved its objectives. Below, an attempt is made to highlight some of these items.

- Time Schedules. The estimate of the effort and calendar time needed to finalize the major objectives after the actual standards part was accepted accounts for the major time slippage in the project.

- Deliverables. The initially listed standards to produce "in case of time" were never addressed.

- Installation. In one of the Development Centers the new standards were not installed one year after the completion due to that "no new products were developed".

Current Status of the PDS Standards

The status of these PDS Software Engineering Standards about two years after their introduction can be characterized as follows:

- The standards are employed throughout the organization and constitute an important part of the quality system.

- Standards management has survived a major organizational change into Philips Information Systems.

- The standards have been spread outside the development organization; internally, to marketing and other development departments; externally, to consultants and software houses as a required way of working.

- No major changes are yet required; in fact, only one small change request has been implemented.

REFERENCES


