NISTIR 5546

A Perspective on Software Engineering Standards

Dolores R. Wallace
Roger J. Martin

U.S. DEPARTMENT OF COMMERCE
Technology Administration
National Institute of Standards
and Technology
Computer Systems Laboratory
Gaithersburg, MD 20899

NIST

NISTIR 5546

A Perspective on Software Engineering Standards

Dolores R. Wallace
Roger J. Martin

U.S. DEPARTMENT OF COMMERCE
Technology Administration
National Institute of Standards
and Technology
Computer Systems Laboratory
Gaithersburg, MD 20899

December 1994

U.S. DEPARTMENT OF COMMERCE
Ronald H. Brown, Secretary

TECHNOLOGY ADMINISTRATION
Mary L. Good, Under Secretary for Technology

NATIONAL INSTITUTE OF STANDARDS
AND TECHNOLOGY
Arati Prabhakar, Director

A Perspective on Software Engineering Standards

Dolores R. Wallace and Roger J. Martin
Technology Administration
National Institute of Standards and Technology
Gaithersburg, MD 20899

1.0 INTRODUCTION

The increasing dependency of United States industry and Federal agencies on computer systems is forcing a reexamination of the definition and scope of software engineering standards. This Software Engineering Standards Forum has an opportunity to recommend how to provide standards in software engineering that will be truly useful in helping customers of software systems to make the right decisions and in assisting producers (referred to by many names, e.g, vendors, developers, suppliers, builders) in implementing the best practices to build reliable, dependable software systems[1].

This paper provides information about the National Institute of Standards and Technology's (NIST) Federal Information Processing Standards (FIPS) and other standards organizations and presents a perspective on software engineering standards.

2.0 FIPS AND RELATED STANDARDS

The Computer Systems Laboratory (CSL) of NIST serves both US industry and the Federal agencies in information technology. This responsibility includes working toward appropriate standards which improve effectiveness of agencies, while simultaneously providing US industry with better competitive capabilities.

2.1 FIPS

A major class of products of NIST/ CSL is Federal Information Processing Standard Publications (FIPS). Many of the early FIPS defined physical specifications for information interchange such as calendar date and identification of State codes or representations such as Hollerith Punched Card. Today, FIPS include the physical specifications and codes but also provide uniform services, processes, interfaces, protocols, and security specifications.

Our responsibility to support Information Technology (IT) users and producers calls for standards which provide enough structure to assist agencies in defining their procurement and regulatory policies while retaining sufficient flexibility for producers to use modern technology and methods

[1]"System" here means only the completed software. Software itself is not a complete system and, for its performance, must reside in a larger system. System in the larger context can mean anything from a microprocessor to a mainframe computer to a radiation device to an entire factory. It is the repeated, indiscriminate use of the word "system" in a very general sense in many standards that can create misunderstanding.

to provide competitive products. In keeping with these objectives, the existing software engineering FIPS are guidelines, not standards, and are intended to serve as guidance to agencies. The FIPS for software engineering are designed to be used as tools to assist users in defining specific requirements for their applications and to assist producers in defining their development and maintenance strategies and processes.

2.2 Standards Organizations

Standards are developed and used at the local, national, and international levels. International and national organizations work together to arrive at standards that will be adopted in many countries. In the US, the American National Standards Institute (ANSI) coordinates the voluntary consensus standards system. ANSI influences the development and content of international standards to meet current technological needs and to facilitate competitiveness in the marketplace. ANSI is the sole US representative and dues-paying member of the two major non-treaty standards organizations: the International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC).

ANSI does not actually develop standards but relies on authorized organizations to develop them and to submit them to the ANSI process for approval as American National Standards. While the specific procedures vary, standards-developing organizations usually seek experts to develop the standard, and use a consensus activity to approve the standard and then submit it to ANSI. Some of these organizations include the Institute for Electrical and Electronics Engineers (IEEE), accredited standards committee X3 for Information Technology, the American Society for Quality Control (ASQC), the American Society for Mechanical Engineers (ASME), the American Society for Test and Measurement (ASTM), the Department of Defense (DOD), and NIST.

Increasingly, consortia (e.g., Open Software Foundation (OSF), Object Management Group (OMG), Reuse Interoperability Group (RIG)) are in the forefront of specifying and developing standards and standards-based products and are offering their specifications to standards-developers for adoption. Similarly, documents from regional workshops (i.e., Open System Implementors Workshop (OIW), European Workshop on Standards (EWOS), Asian Oceanic Workshop (AOW)) may be submitted to the international or national organizations.

3. A VISION FOR SOFTWARE ENGINEERING STANDARDS

In his book Information Technology Standardization, Carl Cargill defines a standard as "the deliberate acceptance by a group of people having common interests or background of a quantifiable metric that influences their behavior and activities by permitting a common interchange [CARG]." The definition of software engineering is "the application of a systematic, disciplined, quantifiable approach to the development, maintenance, and operation of software [IEEE]."

The problem is that software engineering does not have an adequate set of proven principles for defining quantifiable metrics. In other engineering domains, one can go to a handbook and find

algorithms or rules (i.e., the codes) for specific items. The handbooks do not specify how to put together these items in exact procedural manner, those tasks are left to the engineer. The tools are the codes, the basis for creating a product.

Software engineering has not moved forward as an engineering discipline; it does not have a complete set of requirements, each of which meets a technical basis. For example, a technical basis for acceptance criteria for digital technology in safety systems in nuclear power plants must have the following properties:

1. The topic has been clearly coupled to safe operations.
2. The scope of the topic is clearly defined.
3. A substantive body of knowledge exists and the preponderance of the evidence supports a technical conclusion.
4. A repeatable method to correlate relevant characteristics with performance exists.
5. A threshold for acceptance can be established. [BELT]

Few software engineering practices have been codified with this type of technical basis. Yet, software engineering will of necessity expand the classical concepts of engineering. While software is not a physical product, software products have a major impact on health, safety and quality of life of our society. Failure of software due to factors similar to those of physical products (e.g., structural failure, poor workmanship, unanticipated consequences) can have an economic impact as serious as failures in other engineering disciplines.

3.1 A Perspective on Government Needs

The wide range of responsibilities of Federal agencies has varying impact on their requirements for software engineering standards. Often, the agencies are both customers of the software industry and developers of software. The types of software vary widely, ranging from MIS software, laboratory instrumentation, and weapons, and spacecraft. In these instances, standards serve as tools to enable procurement and as guidelines for development.

Agencies with regulatory authority, such as the Nuclear Regulatory Commission (NRC), work with industry to develop a set of standards by which software in regulated systems should be evaluated. NRC may participate in standards committees, but this participation is not an endorsement of the resulting standard. Once a standard is accepted by industry, the NRC will review it and write a regulatory guide to the standards, and may take exceptions to some elements.

There are many other examples of how standards can be used where regulation is involved. The Food and Drug Administration is trying to define policy regarding software engineering standards in several areas (e.g., medical devices, blood banks). Under a requirement by the Office of Safety and Health Administration, a product may need to be tested by an accredited laboratory, (e.g., Underwriter Laboratories (UL)). Often, the regulatory agencies assume an independent role and may have different objectives from industry and resist using standards if the standard does

not appear to have a strong technical basis.

The agencies and industry need top-level guidance with generic terms that anyone can understand and use directly. Language and terminology in standards must be carefully chosen to be applicable across a wide range of needs and uses. This is often much more difficult and subtle than might be apparent to the casual observer. For example, the NRC expects guidance on software in safety systems to address Class 1E systems, where 1E in the nuclear industry means those systems whose failure has the most severe consequences; the term "mission-critical" is unacceptable. In one of our early documents which we developed for the NRC we created quite a stir by using the term "criticality analysis." In software engineering, this term is thought to be well-defined and understood. In this case, many nuclear scientists thought we were discussing critical masses and geometry of radioactive materials. This example is indicative of deeper non-trivial issues that agencies and industry are concerned about when standards enter the scene. The agencies find it difficult to incorporate complete standards into their policies but rather alter and adapt the standards.

3.2 A Perspective from Industry

In proposing and adopting FIPS, it is critical that CSL understand the needs and requirements of both users and industry. We employ many methods to solicit and gather input and feedback on our standards activities from users and industry. For example, last February, CSL convened a Workshop on Advanced Software Technology [WAL1] in which software industry executives were asked to identify needs and barriers to advanced software technology. To provide a basis for achieving this objective, the executives categorized the US software industry into five segments:

- Software Package segment (e.g., word processors, spreadsheets).
- Embedded consumer segment (e.g., automobiles, telephones, refrigerators).
- Large-scale government contractor (systems integrator) segment (e.g., F-22 aircraft, FAA's Advanced Automation System).
- Large-scale commercial segment (e.g., factory automation, factory process control systems).
- Service industries (e.g., America On-Line® and Compuserve®).

Customers and users of the products from these segments expect to receive high integrity software[2] delivered on time and on budget. When this expectation does not occur, the customer,

[2] High integrity software is software that can and must be expected to work dependably in some critical function (e.g., medical devices, safety systems in nuclear power plants, electronic banking, air traffic control), and whose failure to do so may have catastrophic results such as serious injury, loss of life or property, business failure

the user (e.g., Federal agencies, industry, individuals), and the software industry may suffer serious consequences. To prevent this, software engineering principles must be defined and implemented. One of the recommendations at the workshop was that standards are a means of ensuring that these principles are implemented.

While the software industry executives at the Advanced Software Technology Workshop recognized the need for software engineering standards, they also considered them to be potential barriers. To remove the barriers, the standards must be open, flexible, and adaptable. They should provide a transparent framework that helps the user community develop their own strategies for developing good systems. Standards should support customizable software, rather than custom software. In all cases, the standards must provide for high reliability and there must be a strong software engineering basis for the standards.

3.3 A Perspective from NIST

Software engineering standards should be tools which promote a quicker trip to market with a quality product. FIPS for software engineering are not mandatory for Federal agencies but are provided to Federal agencies and industry as guidelines and recommendations. It is our belief that if the standard is good, it will be used. If a standard is bad, even if it is required, people will find a way out of using it. There are many examples of both good standards and a standard that "everyone" tries to get out of using. So, our challenge is to define standards that the industry and users are eager to employ because they result in better, more reliable products.

Within the CSL Software Engineering Group, both the High Integrity Software Systems Assurance Project and the Integrated Software Engineering Project utilize standards as frameworks and guidance. While these standards should not prevent innovation and advancement of technology, they must provide consistency. They help to identify where the "codes" in software engineering already exist, and where they must be developed.

Software engineering standards should provide a common frame of reference [WAL2]:

- as guidance to developers and assurers of the software, for high level processes,
- as an evaluation tool by reviewers (e.g., customers, assurers) of the software,
- as a basis for identifying software engineering practices (techniques) that satisfy an activity, or aggregates of activities comprising high level processes,
- as guidance for developers and assurers in selecting Computer Aided Software Engineering (CASE) tools appropriate for their project,

---

or breach of security. ["High Integrity Software Standards and Guidelines," NIST Special Publication 500-204, September, 1992.]

- as a basis for identifying interrelationships among the activities, hence aiding the CASE tool integrator,

- as a basis for identifying the interfaces between software and system, and technology issues between software and other system components,

- as a basis for identification of appropriate software engineering methods (or practices) mapped to application domains or technical problems which those methods resolve, and

- as a basis for identification where current methods are inadequate and further research is needed.

Once the requirements or codes are clearly identified, perhaps more detailed standards can be developed.

Process standards must provide assurance that the product is of high integrity (e.g., reliability, safety, security). Yet, the process standard must allow flexibility and should not point to specific product requirements or terminology. The industry, or application, must specialize the high level process standards to the needs of specific application domains.

The business requirements for a standard should be defined and its benefits should be visible before the community spends time developing it. If there is no need for it, then it should not be developed. Whether product or process standards, some constraints will still be needed. Product standards should identify the specific items that may be included in a product, but should not presume that a specific product either does or does not need some of these items. A product standard may be needed because certain classes of products cannot be developed without a standard or regulation, to enable duplication, interchange, maintenance, or evaluation [CARG]. The same principles apply for software engineering process standards. Processes should not be so restrictive that the form dictates the content of the software produced.

4.0 RECOMMENDATION

At this Forum we have an opportunity to consider what we must do to ensure that software engineering standards are helpful tools (e.g., open, adaptable) for developing high integrity software products. The issues are many and we must be careful when discussing them to raise plenty of questions. An entire discussion can be generated from trying to assess whether one standard can fit *all* software needs. Typical points for discussion include:

- How much detail can be included?

- Can standards-makers decide in advance which criteria and processes are needed to assure the required quality, for *every* application?

- Is it really helpful to have a standard so encompassing that element after element must

be waived because the scope of a project does not need those items?

- Is it useful to standardize on every aspect of the software development process, to the extent that developers are so busy conforming to form that they cannot have time to put any innovation into the product? Will this conformance cost the producer the market advantage of the product? And, how have we advanced the use of automation?

For standards development and acceptance, we must identify how to get the people who will be affected by the standard to be involved, from large to small companies, across many application domains with varying levels of need for quality.

5.0 CONCLUSION

Standards are necessary to afford the consumer protection from poor quality software that may cause direct or indirect physical or financial harm. They are necessary to provide a common basis for enabling customer and producer alike to estimate costs, schedule, and expected quality, but the standard cannot replace the intellectual activity of the customer and producer that adapts the standard's basis to a specific set of circumstances. Software engineering standards must be useful tools in the software engineering tool set. They must help build and enforce a software engineering discipline. They cannot be a substitute for professional judgment, hard work, and responsibility.

We must accelerate progress toward a professional software engineering discipline. The "horns of the dilemma" that we face is how do we provide good software engineering guidance today while working for stronger, more precise engineering standards for tomorrow?

We urge you to use this Forum as a starting point to think about and discuss these issues and to develop plans for how to tackle this complex and daunting task. Your active participation in the Forum panels will help the entire community to understand better how to arrive at open, flexible software engineering standards.

6.0 REFERENCES

[BELT] Beltracchi, Leo, "NRC Research Activities," Proceedings of the Digital Systems Reliability and Nuclear Safety Workshop, NIST Special Publication 500-216, National Institute of Standards and Technology, Gaithersburg, MD 20899, March, 1994.

[CARG] Cargill, Carl F., Information Technology Standardization, Theory, Process, and Organizations, Digital Equipment Corporation, 1989.

[IEEE] IEEE Std. 610.12-1990, "Glossary of Software Engineering Terminology," The Institute of Electrical and Electronics Engineers, Inc., 345 East 47th Street, New York, NY 10017-2394. February, 1991.

[WAL1] Wallace, D.R., D. R. Kuhn and T. R. Rhodes, "NIST Advanced Software Technology Workshop, February 1, 1994," NIST IR 5500, National Institute of Standards and Technology, Gaithersburg, MD 20899, September, 1994.

[WAL2] Wallace, Dolores. R., and Laura M. Ippolito, "A Framework for Software Assurance," DRAFT NISTIR xxxx, National Institute of Standards and Technology, Gaithersburg, MD 20899, October 1994.