The IEEE Software Engineering Standards Committee has taken deliberate steps to unify and integrate its collection of software engineering standards. Encouraging results are apparent in its latest publication, which is organized around a single architecture for the SESC collection.

An Integrated Collection of Software Engineering Standards

James W. Moore, The Mitre Corporation

Over the years, there has been broad interest in creating software engineering standards. One authoritative survey discovered approximately 315 standards, guides, handbooks, and other prescriptive documents maintained by 46 different organizations (see the "Sources of Software Engineering Standards" sidebar).[1] Nevertheless, uptake of the available standards has been somewhat disappointing. Hopeful users report difficulty in finding the standards that suit their particular situation among the numerous ones available. They also report that detailed differences between standards make it difficult to apply them in unison. For example, in an area of overlap between two standards, each might emphasize a different approach or use different terminology.

We need an approach to managing a standards collection that emphasizes integrating the various standards. Since 1991, those of us working on the IEEE Computer Society's Software Engineering Standards Committee have undertaken efforts to manage the standards collection to promote consistency. Although the col-

0740-7459/99/$10.00 © 1999 IEEE

Sources of Software Engineering Standards

Software engineering standards concern the responsible practice of software engineering. They often deal with processes, but sometimes they deal with generic product characteristics or supporting resources. The subjects of the standards include phrases familiar to large-scale software developers—configuration management, quality assurance, verification, validation, and so forth. The standards generally do not deal with specific programming languages or technologies. The disciplines provided by the standards generally transcend the lifetimes of specific technologies.

Three organizations are generally regarded as the source of international standards—the International Organization for Standardization, the International Electrotechnical Commission, and the International Telecommunications Union. Two of those organizations cooperate in a Joint Technical Committee, ISO/IEC JTC1, responsible for information technology. A subcommittee, ISO/IEC JTC1/SC7, is responsible for standards related to software engineering and software systems engineering. SC7 currently manages a collection of about two dozen standards, the most popular being *ISO/IEC 12207*, *Software Life Cycle Processes*. Other technical committees and subcommittees of ISO and IEC make standards in related areas—for example, *ISO TC176* (*Quality Management*), *IEC TC56* (*Dependability*), and *IEC SC65A* (*Functional Safety*).

Standards-making in the US is not rigidly delegated as it is in many other countries. Over 500 organizations in the US make standards of some kind. Two organizations mentioned in this article are the Electronic Industries Alliance and the Institute for Electrical and Electronic Engineers. EIA has played an important role in "demilitarizing" the standards for complex software development that were originally written for use in the defense industry.

The Software Engineering Standards Committee of the IEEE Computer Society manages the world's most comprehensive collection of software engineering standards (nearly 50), developed since 1979. SESC serves as a developer of these standards, but also as an integrator of specifications and standards developed by other organizations. It has adopted, sometimes with changes, standards developed by organizations such as ISO/IEC JTC1/SC7 and the Project Management Institute.

lection has doubled in size, we have substantially improved its degree of integration. The process is not yet complete but significant progress has been made, culminating in the publication of the 1999 four-volume edition of SESC standards—packaged along the lines of the integrating principles for the collection.

This article explains the principles of the SESC collection and describes our progress toward integrating the various standards within it.

Buyer and Seller Benefits

To some, the value of using software engineering standards might be obvious—they contribute to disciplined practice, hence they improve product quality. Although these reasons validly account for using standards in the software engineering craft, they do not characterize the unique contribution of standards to the profession. For this, we must look at the value of standards to those buying and selling software engineering goods and services.

Many goods and services can be confidently purchased after simple examination or after studying the supplier's product literature. The complexity of software products, however, induces a need for a more thorough analysis. Whether purchased as a completed product or as a contracted development, the purchasing or acceptance decision is complicated because important characteristics may be effectively hidden from examination until unusual circumstances or changing patterns of usage reveal them.

Standards can provide assistance and can protect the buyer by

- providing a vocabulary for communication between the buyer and seller;
- providing objective criteria for otherwise vague claims regarding the product's nature;
- defining methods for characterizing elusive characteristics, such as reliability; and
- assuring the seller that specific quality assurance practices were applied.

The benefit of standards in protecting the seller is probably underappreciated in the software engineering community. From this viewpoint, standards are important, not because they represent best practice, but because they represent good enough practice. Courts generally view the application of standards as important evidence that engineers perform their work with appropriate diligence and responsibility. If sued for negligence or reckless conduct, an engineer can cite the standards used when he or she conducted the work to demonstrate that it was performed in accordance with codified professional practices.

By providing important benefits for both the buyer and the seller, software engineering standards support the emergence of a software engineering profession characterized by consensually validated

norms for responsible conduct. With such clear benefits, you would expect a nearly universal application of software engineering standards. Unfortunately, this is difficult due to the vast amount of available and occasionally inconsistent information.

Vision 2000 Architecture

Early in the 1990s, the SESC established a planning committee to initiate the long-range efforts needed to integrate its collection. The committee studied customer needs[2] and surveyed existing standards,[3] concluding that there was no shortage of available advice for the practice of software engineering. However, there existed no clear way for users to select the advice appropriate to their needs. Furthermore, the individually optimized nature of each standard presented obstacles to selecting and applying them together. The software engineering community needed an integrated collection of standards that could be applied in unison and from which users could easily select appropriate standards.

With this information, we were ready to develop an integrating architecture for the SESC collection, termed Vision 2000.[4] The most recently published edition of the SESC collection[5] reflects the Vision 2000 architecture, which comprises three important organizing criteria (see Figure 1). The concept of the first organizing criterion, *normative levels*, is that different standards should provide different levels of advice—sometimes detailed, sometimes general—for different uses. The second organizing criterion, *objects of software engineering*, recognizes that software engineering standards address four different objects: customer, process, product, and resource. The third organizing criterion is *relationships to other disciplines*. The SESC software engineering collection is positioned within the context of other standards selected from software engineering, quality management, and various systems engineering disciplines.

Normative levels

We borrowed the concept of normative levels from other successful standards collections. The top layer includes standards for terminology and other key concepts. Such standards are generally nonprescriptive; they simply provide definitions, taxonomies, or other reference material that can be used in other standards in the collection. The IEEE software engineering vocabulary, *IEEE Std. 610.12*, falls under this category.

The next layer is also nonprescriptive, occupied by one or a few documents that serve as an overall guide to the remainder of the collection. The document explains the collection's architecture and the key relationships among the standards within it. We decided to fill this layer by authorizing and endorsing a textbook rather than writing a standard.[6]

The third layer contains standards providing policies or principles to a user. Principles are useful because it is difficult to write detailed standards covering the entire conceivable range of usage. In a specific situation, if the details don't seem applicable, a user can apply the principles instead. In the case of the SESC collection, there are currently no principles documents dealing with resources and products. However, portions of *IEEE/EIA 12207* fill this role for the standards dealing with customers and processes.

The fourth layer, element standards, is the one most familiar to standards users. It contains documents with conformance requirements in various important areas. Most of the SESC collection's standards are grouped into this layer.

The fifth layer makes provisions for application

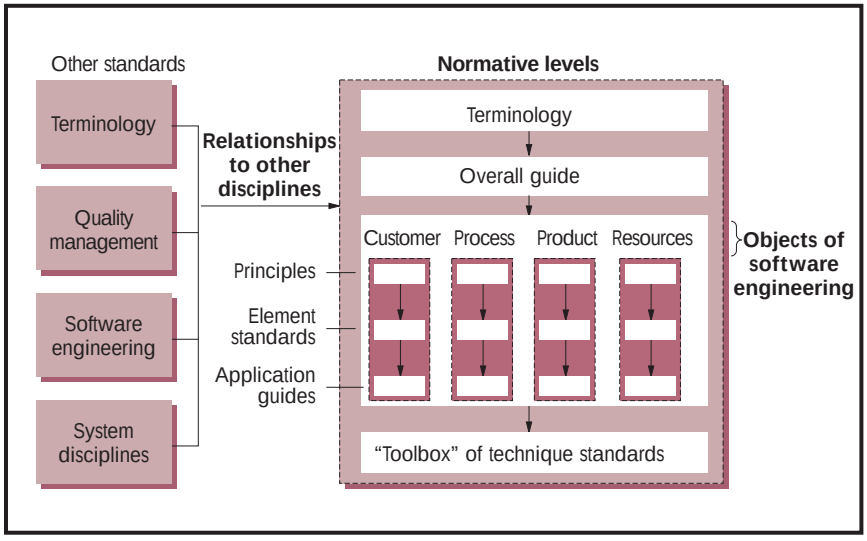

FIGURE 1. The Software Engineering Standards Committee's architecture for its standards collection. The main organizing criteria are normative levels, objects of software engineering, and relationships to other disciplines.

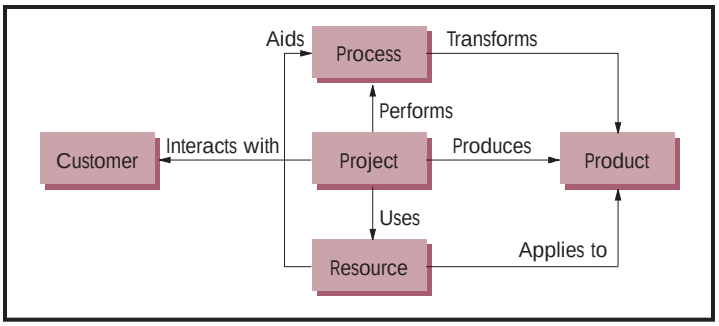

FIGURE 2. The objects of software engineering, suggesting a categorization of standards in the subject areas of customer, process, product, and resource.

guides. Sometimes, users need supplementary documents to describe how to apply an element standard within particular situations. These documents are often guides but can be standards deriving their conformance requirements from the appropriate element standards. For example, we have an element standard on a software quality assurance plan and an application guide describing the overall discipline of SQA planning.

The bottom layer is viewed as a toolbox of detailed techniques. The standards in this layer are "subroutines" that any of the other standards can invoke to provide requirements in a specific area. An example from the SESC collection is *IEEE Std. 1044* regarding the classification of software anomalies.

Objects of software engineering

The objects of software engineering result from the abstract model of software engineering depicted in Figure 2. This model centers on the software engineering project as the focal point for applying software engineering standards. In this view, a project uses resources in performing processes to produce products for a customer. The model is instructive in providing four major subject areas that can be treated by standards: customer, process, product, and resource. We organized the four volumes of the 1999 standards edition according to the four objects.

Relationships to other disciplines

Software engineering does not exist in isolation from other disciplines. Its purpose is to apply the principles of mathematics, engineering, and computer science to various application domains. In addition, it inherits principles from more general disciplines such as systems engineering, quality management, and project management. Its work is also influenced by cross-cutting disciplines such as dependability, safety, and security.

Particularly in this time of emphasis on process improvement, we cannot afford for our standards to be inconsistent with closely related standards from other disciplines. It would be a tragedy if the standards for the best practices of software engineering were capriciously incompatible with, say, the standards for quality management.

Therefore, where the relationship is strong, and where suitable standards exist, we have selected a few key standards from other disciplines as targets for integration. For example, *ISO 9000-3* provides a bridge between software engineering and the famous *ISO 9000* series of quality management standards, and the Project Management Institute's *Guide to the Project Management Body of Knowledge* (PMBOK) describes general project management principles that SESC standards adapted to the subject of software project management (http://www.pmi.org/publictn/pmboktoc.htm).

By adopting or otherwise recognizing key standards from related disciplines, we avoid the need to reinvent key principles.

AN UMBRELLA STANDARD: IEEE/EIA 12207

Not all integrating standards of the SESC collection are borrowed from other disciplines. *IEEE/EIA 12207*, *Software Life Cycle Processes*, is an umbrella for all of the customer and process standards in the SESC collection.

ISO/IEC 12207

IEEE/EIA 12207 is an adoption of a 1995 ISO/IEC standard with the same name and number. The international standard establishes a common framework for software throughout its life cycle from conception through retirement, and it addresses the organizational context of those software processes both from the system's technical viewpoint and from the enterprise's business viewpoint. The standard is widely regarded as providing a basis for world trade in software services; adoption of the standard is completed or underway in most of the world's major countries.

The *ISO/IEC 12207* standard improved over past standards in similar areas. Most importantly, it is defined at the process rather than the procedure level. Rather than provide the step-by-step requirements characteristic of a procedure, it describes continuing responsibilities that must be achieved and maintained during the life of the process. The standard addresses the functions to be performed rather than the organizations that will execute them. (For example, the standard describes a quality assurance process; this does not imply that a conforming enterprise must es-

tablish a quality assurance department.) The standard describes software development, maintenance, and operation within the context of the system, thus effectively establishing the minimum system context essential to software processes.

Three categories describe the *ISO/IEC 12207* processes:

- *Primary processes* are executed by parties who initiate or perform major roles in the software life cycle. They include both business (acquisition and supply) and technical roles (development, operation, and maintenance).
- *Supporting processes* contribute to the execution of other processes as an integral part with distinct goals. They include documentation, configuration management, quality assurance, verification, validation, joint review, audit, and problem resolution.
- *Organizational processes* inherently exist outside the individual project's scope, but the project employs instances of them. They include management, infrastructure, improvement, and training.

The standard addresses the functions to be performed rather than the organizations that will execute them.

IEEE/EIA 12207

IEEE/EIA 12207.0 adds a foreword and some annexes to the text of the international standard. Two additional guidance parts were added to the standard: *IEEE/EIA 12207.1* provides guidance on the data produced by the life cycle processes and is cross-referenced to the provisions of *12207.0*, and *IEEE/EIA 12207.2* provides guidance on implementing processes by quoting the complete text of *12207.0* and interspersing guidance notes.

12207.1 describes data but not documents. It describes 84 different information items, which the user selects and packages into documents appropriate for the project. Forty of the information items have specific content (but not format) requirements, while the other 44 information items are classified as one of seven different kinds of data that have generic content requirements. A *12207.1* user might apply it as a guide, meaning that it is presumed simply to offer good advice. On the other hand, *12207.1* also contains optional conformance provisions that permit users to cite the standard if they want to make strong claims regarding the nature of the data that their processes produce. The user can claim that one or more documents conform to *12207.1* by providing a mapping from the documents to the selected information items. The mapping must demonstrate that the document satisfies generic and specific content requirements; captures the data required by the cross-referenced provisions of *12207.0*; and achieves some general requirements for the treatment of data.

12207.1 also provides cross-references to other IEEE standards that might be helpful in implementing the provisions concerning data. For instance, a user might choose to adopt *IEEE Std. 1016*, *Software Design Descriptions*, to detail the data provisions related to the information item for software item description. Working in the other direction, the SESC has supplemented each of the referenced IEEE standards with a content map describing the extent to which the standard satisfies the data provisions of *12207.1*. Within the next few years, the SESC will revise the content of each standard so that it directly implements the relevant provisions of *12207.1*.

IEEE/EIA 12207 also plays an important role in the principles layer of the SESC architecture. The IEEE/EIA adoption of *12207* supplemented each of the 17 processes with a statement of objectives. In unusual cases, in which the more detailed *12207* requirements are unsuitable for adoption, an organization can instead choose to adopt the processes at the objectives level.

In overall terms, we have adopted policy designating *12207* as a strategic, integrating standard for its collection. All of the relevant standards of the SESC collection will be revised to improve their fit with *12207*; in particular, many of them will detail the processes of the *12207* framework. From the user's viewpoint, *IEEE/EIA 12207* will serve as a single entry point to all the process standards of the IEEE software engineering collection.

We are also using *IEEE/EIA 12207* as the baseline to articulate new processes. For example, *IEEE Std. 1517*, *Software Reuse Processes*, adds three reuse-specific processes to those of *12207*, and the planned *IEEE 1540* standard (still under development) will add a software risk management process.

SESC Four-Volume Edition

All of the SESC standards are available for individual purchase from the IEEE (http://standards.

ieee.org/catalog). Every few years, we have gathered our standards and collectively published them in a volume similar in size to a major city's phone directory. With total page count approaching 2,400, the 1999 edition needed a new approach.

The four objects of the Vision 2000 architecture suggested a four-volume packaging, with each volume containing the standards pertaining to customer, product, process, or resource. We bundled the standards from the terminology layer into the customer volume and the standards from the technique layer into the resource volume. We made the guide to the collection available at a discounted price when purchased with the four-volume edition. We also omitted from the edition a few large standards intended for specific audiences, making them available for purchase separately.

Of course, it wasn't practical to publish multiple copies of standards that legitimately fit into more than one category. So, each standard had to be forcefitted into a single category for publication purposes. Our rationale for these decisions is explained below.

The customer volume

Because *IEEE/EIA 12207* is the umbrella standard for both process and customer interaction, its placement was one of the toughest decisions in designing the edition. We decided to place all three parts of the standard into the customer volume so that the volume would be self-contained from the software acquirer's point of view. Those who desire a more detailed standard for software acquisition can also find *IEEE Std. 1062* in this volume.

Another key decision involved treating systems engineers as part of the customer set for software engineering. Certainly, *IEEE/EIA 12207* takes this view when it treats software requirements as derived from an essential systems context. So, all of the standards involving the system context of software development are in this volume:

- *IEEE Std. 1220, Systems Engineering Process*,
- *IEEE Std. 1228, Software Safety Plans*,
- *IEEE Std. 1233, Systems Requirements Specification*, and
- *IEEE Std. 1362, Concept of Operations*.

To complete its self-contained nature, this volume includes the vocabulary standard from the terminology layer of the collection, *IEEE Std. 610.12*.

The process volume

Although we placed *IEEE/EIA 12207* in the edition's customer volume, we placed another important umbrella process standard in the process volume; *IEEE Std. 1074* addresses the process architect and provides building blocks for constructing processes that meet the requirements of *12207* or other standards.

You'll find standards in this volume that provide additional details on many of the technical processes and activities of *12207*:

- *IEEE Std. 730, Software Quality Assurance Plans*,
- *IEEE Std. 828, Software Configuration Management Plans*,
- *IEEE Std. 1008, Software Unit Testing*,
- *IEEE Std. 1012, Software Verification and Validation*,
- *IEEE Std. 1028, Software Reviews*, and
- *IEEE Std. 1219, Software Maintenance*.

IEEE/EIA 12207 treats management as a process, so *IEEE Std. 1058, Software Project Management Plans,* is included, along with *IEEE Std. 1490*, the *PMBOK Guide*, which provides a broader treatment of project management issues.

Some of the tough calls in placing standards into the volumes relate to the practice in some SESC standards of expressing process requirements in terms of the content of a plan to be developed. In each case, we evaluated the standard to determine if its emphasis was truly on the process or the plan's content. As a result, we included *IEEE Std. 730* and *IEEE Std. 828* in this volume, but placed *IEEE Std. 829, Software Test Documentation*, in the resource volume. *IEEE Std. 1045, Productivity Metrics*, is in this volume, because it offers means for measuring the performance of software processes.

Generally, you'll find standards related to system engineering processes elsewhere—many of them in the customer volume. *IEEE Std. 830* regarding software requirements specifications is in the resource volume.

Some developers, particularly those in the defense community, might be interested in using *EIA/IEEE J-Std-016* to guide their software development process. We decided to omit this standard from the four-volume edition for several reasons: it is currently being revised; it is already standalone and users do not need the other material from this volume; and it is large and would have significantly increased the volume's price. *EIA/IEEE J-Std-016* is available for separate purchase from either the IEEE or EIA.

The product volume

The trend in software engineering over the past 15 years or so has been to focus on evaluating and

improving processes. Nevertheless, we should not forget that the purpose of any engineering effort is to create a product. This volume includes standards useful for software product evaluation.

Unfortunately, unlike the customer and process volumes, there is no standard providing principles to serve as a unifying umbrella for the others. The closest candidate would be *IEEE Std. 1061*, which provides a methodology for software quality metrics. *IEEE Std. 982.1* is a dictionary of metrics that can be applied to measure software reliability and related characteristics.

IEEE Std. 1465 is an adoption of *ISO/IEC 12119* and provides quality requirements for software packages—that is, prepackaged software products. Because user documentation is an important component of a software product, *IEEE Std. 1063* is included in this volume, although other documentation standards appear elsewhere.

All standards focusing on the processes for ensuring product quality appear elsewhere in the edition.

The resource volume

The term resource is deliberately broad, encompassing anything that might be used or consumed while executing a software process or creating a software product. Accordingly, we included a wide variety of standards in this volume. It is perhaps not surprising, therefore, that there is no umbrella standard to provide general principles in this area.

This volume contains the so-called "IDEF" notational standards—("Integrated Definition," a modeling language, combining graphics and text, used to analyze and define system functions and requirements). The volume contains both *IEEE Std1320.1*, specifying IDEF0, and *1320.2*, specifying IDEF1X97 (IDEFObject). The basic interoperability data model is a series of standards (*IEEE Std. 1420.1* and its supplements) providing a data model for describing and interchanging reusable software components. The volume includes these standards and their guide, *IEEE Std. 1430*.

In addition, it has standards for CASE tools: *IEEE Std. 1462* considers tool evaluation and selection and *IEEE Std. 1348* considers organizational adoption. A standard on CASE tool interconnections, *IEEE Std. 1175*, was omitted because of its size and because its audience is primarily tool developers.

Some environments do not utilize CASE tools when transferring engineering data. Instead, they apply documentation conventions to move data among the various processes and phases of a software project. The resource volume holds standards appropriate for this purpose:

- *IEEE Std. 830, Software Requirements Specifications*,
- *IEEE Std. 829, Software Test Documentation*, and
- *IEEE Std. 1016, Software Design Descriptions*.

Finally, the standard in the SESC architecture's techniques layer is included in this volume. *IEEE Std. 1044* (and an accompanying guide) describes the classification of software anomalies for a variety of purposes.

The level of integration among the IEEE software engineering standards is not yet perfect. As each individual standard is revised, on a cycle of roughly five years, it will be modified to fit more smoothly with IEEE/EIA 12207 and with the other standards in the collection. Of course, the IEEE collection continues to grow as additional subjects are treated. We also cooperate with the appropriate international standards committee to encourage the evolution of ISO/IEC 12207 in a direction consistent with the needs of SESC's users. ❖

References

1. S. Magee and L.L. Tripp, *Guide to Software Engineering Standards and Specifications*, Artech House, Boston, 1997.
2. SESC Long Range Planning Group, *Master Plan for Software Engineering Standards, Version 1.0*, Dec. 1993; http://computer.org/standard/sesc/MasterPlan (current Oct. 1999).
3. SESC Business Planning Group, *Survey of Existing and In-Work Software Engineering Standards, Version 1.2*, Dec. 1996; http://computer.org/standard/sesc/survey0.htm (current Oct. 1999).
4. SESC Business Planning Group, *Vision 2000 Strategy Statement, Version 0.9*, Aug. 1995; http://computer.org/standard/sesc/strategy.htm (current Oct. 1999).
5. Institute of Electrical and Electronics Engineers, *Software Engineering, 1999*, Vols. 1–4, IEEE Press, Piscataway, N.J., 1999.
6. J.W. Moore, *Software Engineering Standards: A User's Road Map*, IEEE Computer Soc. Press, Los Alamitos, Calif., 1997.

About the Author

James W. Moore is the standards coordinator for the WC3 Center of The Mitre Corporation. He serves as a member of the Management Board of the IEEE Software Engineering Standards Committee, as a member of the IEEE Standards Board Review Committee, and as the head of the US delegation to ISO/IEC JTC1/SC7 (Software Engineering). He received his BS in mathematics from the University of North Carolina and his MS in systems and information science from Syracuse University. The IEEE Computer Society has recognized him as a charter member of their Golden Core and has given him the Meritorious Service Award. Contact him at The Mitre Corporation, 1820 Dolly Madison Blvd., W534, McLean, VA 22102; james.w.moore@ieee.org.