Introduction to the IEEE Software and Systems Engineering Core Standards

November 2009

Prepared by:
Chuck Walrad
Meredith Newmaker
Rebecca Smith

About the IEEE Computer Society

The Institute of Electrical and Electronics Engineers (IEEE) is the world's largest technical professional society. With membership numbering more than 375,000 individuals in 150 countries, it publishes 30 percent of the world's technical papers in the scope described by its name. The IEEE is organized into 39 technical societies, some of which are active in developing standards, including the Computer Society. The IEEE Computer Society is the largest association for computer professionals in the world. Founded more than 50 years ago, it is now the largest of the technical societies of IEEE.

About S2ESC

The Computer Society's Software and Systems Engineering Standards Committee (S2ESC) develops and maintains a collection of nearly 50 standards for software and systems engineers. S2ESC is chartered to ensure that its family of software engineering standards are relevant, coherent, comprehensive and effective in use. These standards are for use by practitioners, organizations, and educators to improve the effectiveness and efficiency of their software engineering processes, to improve communications between acquirers and suppliers, and to improve the quality of delivered software and systems containing software.

S2ESC traces its roots back to 1976 when the Software Engineering Standards Subcommittee of the Technical Committee on Software Engineering (TCSE) was created. Its first standard, IEEE Std 730, Software Quality Assurance, was published on a trial use basis three years later. By 1997, the collection had grown to 44 documents. S2ESC also participates in international standards-making as a member of the US Technical Advisory Group (TAG) to ISO/IEC JTC1/SC7 and as a direct liaison to SC7 itself.

In addition to the development of standards, S2ESC sponsors or cooperates in annual US or international conferences and workshops in its subject area.

Software Engineering as a Profession

Hallmarks of a profession are that it have a recognized Body of Knowledge and that its practitioners master that body of knowledge .The Body of Knowledge often uses a shared, specialized vocabulary. Often the Body of Knowledge is augmented by standards of performance. Further, professions typically adopt of Codes of Ethics. Certification of professionals often rest on these four elements.

Both the IEEE’s Software Engineering Body of Knowledge (SWEBOK) and S2ESC’s body of standards are key elements of the Software Engineering profession. The IEEE's *Guide to the Software Engineering Body of Knowledge - 2004 Version* o

(SWEBOK) defines the field and gives coverage of the knowledge practicing software engineers should have. There is also an IEEE "Software Engineering Code of Ethics".[12] In addition, there is a Software and Systems Engineering Vocabulary (SEVOCAB),[13] published on-line by the IEEE Computer Society.

Contents

Purpose of This Standards Guide

The purpose of this Standards Guide for software producers, as well as producers of systems with embedded software, is to help informatics practitioners ensure that developed and deployed systems and software:

- can be and are verified and validated
- meet the purpose for which they are intended
- are robust, reliable and resilient enough to consistently perform to their intended use

IEEE System and Software Engineering Standards are as critical to industry as they have been to space exploration. This Guide will introduce you to the core subset of S2ESC's portfolio of standards.

The Value of System and Software Standards to Industry

The aggressive transition to technology-based Information will be successful only if software and software-intensive systems -- which encompass myriad products and processes in complex ways -- seamlessly collect, aggregate, share, analyze and present dynamic information in a timely manner.

For example, the anticipated human and economic benefits from the present rapid transition to health information technologies demands that software and systems developers of medical devices, digital medical records, and administrative, financial and regulatory systems (e.g. public health, service/healthcare providers and payers) design, develop and deliver interoperable products, processes and services that are safe, secure, reliable and robust.

The application of IEEE S2ESC systems and software standards helps software producers by *simplifying product development processes, avoiding the pitfalls that have overcome many software projects, and thus reduces non-value-adding efforts and costs.* Adoption and implementation of the core software and systems engineering standards across companies that produce or tailor Information systems and devices *increases their development organization's ability to deliver robust software in shorter timeframes.* Even more important, the consistent use of these IEEE standards lowers the risks of delivering faulty products.

Standards for Software and Systems Engineering encompass the full software and systems lifecycles, from concept and development to delivery and maintenance, and even the reuse of software components.

Standards Essential To Informatics Technology Producers

Ensuring that *delivered software meets its purpose and consistently performs to its intended use* is vital to effective information delivery. As fundamental *building blocks for international systems and software development,* IEEE Software and Systems Engineering Standards help producers assure *interconnectivity, interoperability* and verification *of new* Informatics *products and* systems *enabling the rapid implementation and trusted use of* medical technologies*.*

The essential set[1] of IEEE System and Software Standards that are key to the development and delivery of robust software are given in the table below.

Table of Essential Standards

Number	Official Designation	Name
730	IEEE Std 730-2002	IEEE Standard for Software Quality Assurance Plans
828	IEEE Std 828-1998 (R2005)	IEEE Standard for Software Configuration Management
830	IEEE Std 830-1998	IEEE Recommended Practice for Software Requirements Specifications
1008	IEEE Std 1008-1987(R2002)	IEEE Standard for Software Unit Testing
1012	IEEE Std 1012-2004	IEEE Standard for System and Software Verification and Validation
1016	IEEE Std 1016-1998(R2009)	IEEE Recommended Practice for Software Design Descriptions
1028	IEEE Std 1028-2008	IEEE Standard for Software Reviews and Audits
1058	IEEE Std 1058-1998	IEEE Standard for Software Project Management Plans
1063	IEEE Std 1063-2001	IEEE Standard for Software User Documentation

[1] This set is available from IEEE on the *Essentials* CD.

Numb er	Official Designation	Name
	(R2007)	
1074	IEEE Std 1074-2006	IEEE Standard for Developing a Software Project Life Cycle Process
12207	ISO/IEC/IEEE 12207: 2008	Systems and Software Engineering -- life cycle processes
14764	IEEE Std 14764-2006	Software Engineering--System Life Cycle Processes--Maintenance

In addition, HITSP may wish to examine these additional S2ESC standards:

Number	Official Designation	Name
829	IEEE Std 829-2008 (March 27)	IEEE Standard for Software and System Test Documentation
1044	IEEE Std 1044-2002	IEEE Standard Classification for Software Anomalies
1062	IEEE Std 1062-2002	IEEE Recommended Practice for Software Acquisition
1233	IEEE Std 1233-1998 (R2002)	IEEE Guide for Developing System Requirements Specifications
1362	IEEE Std 1362-1998 (R2007)	IEEE Guide for Information Technology-System Definition-Concept of Operations (ConOps) Document
14143-1	ISO/IEC/IEEE14143-1	Implementation Note for IEEE Adoption of ISO/IEC 14143-1:2007 Information Technology - Software Measurement - Function Size Measurement Part I: Definition of Concepts
15288	ISO/IEC/IEEE 15288(2008)	Systems and Software Engineering --System Life cycle Processes

Short Descriptions of the Standards Mentioned in This Booklet

IEEE Std 730-2002: Software Quality Assurance Plans

Abstract: The standard specifies the format and content of software quality assurance plans. It
meets the IEEE/EIA 12207.1 requirements for such plans.

The SQA plan defines the means that will be used to ensure that software developed for a specific product satisfies the user's requirements and is of the highest quality possible within project constraints. In order to do so, it must first ensure that the quality target is clearly defined and understood. It must consider management, development, and maintenance plans for the software.[2]

IEEE Std 828-2005: Software Configuration Management

Abstract: The minimum required contents of a Software Configuration Management (SCM) Plan
are established via this standard. The application of this standard is not restricted to any form, class, or type of software. This standard applies to the entire life cycle of critical software (e.g., where failure would impact safety or cause large financial or social losses), as well as to noncritical software and to software already developed. A new version of this standard that defines the individual processes that make up software configuration management is currently underway, and expected to be published in 2010.

IEEE Std 829-2008: Software and System Test Documentation

Abstract: Test processes determine whether the development products of a given activity
conform to the requirements of that activity and whether the system and/or software satisfies its
intended use and user needs. Testing process tasks are specified for different integrity levels.
These process tasks determine the appropriate breadth and depth of test documentation. The
documentation elements for each type of test documentation can then be selected. The scope of
testing encompasses software-based systems, computer software, hardware, and their

[2] Guide to SWEBOK 2004. 2.1

interfaces. This standard applies to software-based systems being developed, maintained, or
reused (legacy, commercial off-the-shelf, Non-Developmental Items). The term "software" also
includes firmware, microcode, and documentation. Test processes can include inspection,
analysis, demonstration, verification, and validation of software and software-based system
products.

IEEE Std 830-1998: Software Requirements Specifications

Abstract: The content and qualities of a good software requirements specification (SRS) are described
and several sample SRS outlines are presented. This recommended practice is aimed at
specifying requirements of software to be developed but also can be applied to assist in the selection
of in-house and commercial software products. It is very useful for development teams to use as a checklist to remember all the different types of requirements (functional, derived, operational, quality, etc.) that must be provided to the development team.

IEEE Std 1008-1987: Software Unit Testing

Unit testing verifies the functioning in isolation of software pieces which are separately testable. Depending on the context, these could be the individual subprograms or a larger component made of tightly related units. A test unit is defined more precisely in the IEEE Standard for Software Unit Testing (IEEE1008-87), which also describes an integrated approach to systematic and documented unit testing. Typically, unit testing occurs with access to the code being tested and with the support of debugging tools, and might involve the programmers who wrote the code.[3] Test cases should be under the control of software configuration management and include the expected results for each test.[4]

Key aspects of test planning include coordination of personnel, management of available test facilities and equipment (which may include magnetic media, test plans and procedures), and planning for possible undesirable outcomes. If more than one baseline of the software is being maintained, then a major planning consideration is the time and effort needed to ensure that the test environment is set to the proper configuration.[5]

[3] Guide to SWEBOK 2004. 2.1.1
[4] Guide to SWEBOK 2004. 5.2.2
[5] Guide to SWEBOK 2004. 5.2.1

IEEE Std 1012-2004: System and Software Verification and Validation

Abstract: Software verification and validation (V&V) processes determine whether the development products of a given activity conform to the requirements of that activity and whether the software satisfies its intended use and user needs. Software V&V life cycle process requirements are specified for different software integrity levels. The scope of V&V processes encompasses software-based systems, computer software, hardware, and interfaces. This standard applies to software being developed, maintained, or reused [legacy, commercial off-the shelf (COTS), non-developmental items]. The term software also includes firmware, microcode, and documentation. Software V&V processes include analysis, evaluation, review, inspection, assessment, and testing of software products.

IEEE Std 1016-2009: Software Design Descriptions

Abstract: The necessary information content and recommendations for an organization for Software Design Descriptions (SDDs) are described. An SDD is a representation of a software system that is used as a medium for communicating software design information. This recommended practice is applicable to paper documents, automated databases, design description languages, or other means of description.

IEEE Std 1028-2008: Software Reviews and Audits

Abstract: Five types of software reviews and audits, together with procedures required for the execution of each type, are defined in this standard. This standard is concerned only with the reviews and audits; procedures for determining the necessity of a review or audit are not defined, and the disposition of the results of the review or audit is not specified. Types included are management reviews, technical reviews, inspections, walk-throughs, and audits.

IEEE Std 1044-1993(R2002): Classification for Software Anomalies

Abstract: A uniform approach to the classification of anomalies found in software and its documentation
is provided. The processing of anomalies discovered during any software life cycle phase are
described, and comprehensive lists of software anomaly classifications and related data items that
are helpful to identify and track anomalies are provided. This standard is not intended to define procedural or format requirements for using the classification scheme. It does identify some classification measures and does not attempt to define all the data supporting the analysis of an anomaly.

IEEE Std 1058-1998: Software Project Management Plans

Abstract: The format and contents of software project management plans, applicable to any type or size of software project, are described. The elements that should appear in all software project management plans are identified. This standard has recently been merged into ISO/IEC 16326.

IEEE Std 1062-1998(R2002): Software Acquisition

Abstract: This standard provides a set of useful quality practices for use during one or more steps in a software acquisition process is described. This recommended practice can be applied to software that runs on any computer system regardless of the size, complexity, or criticality of the software, but is more suited for use on modified-off-the-shelf software and fully developed software.

IEEE Std 1063-2001(R2007): Software User Documentation

Abstract: Minimum requirements for the structure, information content, and format of user documentation, including both printed and electronic documents used in the work environment by users of systems containing software, are provided in this standard.

IEEE Std 1074-2006: Developing a Software Project Life Cycle Process

Abstract: This standard provides a process for creating a software project life cycle process

(SPLCP). It is primarily directed at the process architect for a given software project.

IEEE Std 1074-2006 is unique in that it provides activities for assuring building in security throughout the software life cycle.

IEEE Std 1074 is a standard for establishing the process to be used in a software development , maintenance or other type of software project, including disposal/ withdrawal of the product. This standard requires selection of a users software project life cycle model (SPLCM) based on the organization’s mission, vision, goals, and resources. IT does not impose, define or imply a particular software life cycle model or methodology. This standard describes the individual activities that are to be used within the selected model and provides examples of mapping them onto typical SPLCMs.

This standard may also be used to develop organizational processes to support software development and maintenance or to develop special, single-function processes within a project.

IEEE Std 1233-1998(R2002): Developing System Requirements Specifications

Abstract: Guidance for the development of the set of requirements, System Requirements Specification (SyRS), that will satisfy an expressed need, is provided. Developing an SyRS includes the identification, organization, presentation, and modification of the requirements. Also addressed are the conditions for incorporating operational concepts, design constraints, and design configuration requirements into the specification. This guide also covers the necessary characteristics and qualities of individual requirements and the set of all requirements.

IEEE Std 1362-1998(R2007): Information Technology – System Definition – Concept of Operations (ConOps) Document

Abstract: The format and contents of a concept of operations (ConOps) document are described. A
ConOps is a user-oriented document that describes system characteristics for a proposed system from the users’ viewpoint. The ConOps document is used to communicate overall quantitative and qualitative
system characteristics to the user, buyer, developer, and other organizational elements (for example,
training, facilities, staffing, and maintenance). It is used to describe the user organization(s), mission(s), and organizational objectives from an integrated systems point of view.

IEEE Std 12207-2008: Systems and Software Engineering —Software Life Cycle Processes

Abstract: This International Standard establishes a common framework for software life cycle processes,
with well-defined terminology, that can be referenced by the software industry. It applies to the acquisition of systems and software products and services, to the supply, development, operation, maintenance, and disposal of software products and the software portion of a system, whether performed internally or externally to an organization. Those aspects of system definition needed to provide the context for software products and services are included. Software includes the software portion of firmware. This revision integrates ISO/IEC 12207:1995 with its two amendments and was coordinated with the parallel revision of ISO/IEC 15288:2002 (System life cycle processes) to align structure, terms, and corresponding organizational and project processes. This standard may be used stand alone or jointly with ISO/IEC 15288, and supplies a process reference model that supports process capability assessment in accordance with ISO/IEC 15504-2 (Process assessment). An annex provides support for IEEE users and describes relationships of this International Standard to IEEE standards.

IEEE Std 14143.1-2007: Information technology - software measurement - functional size measurement. Part 1: definition of concepts

Abstract: IEEE Std 14143-1:2007 defines the concepts of FSM (Functional Size Measurement). The concepts of Functional Size Measurement (FSM) are designed to overcome the limitations of earlier methods of sizing software by shifting the focus away from measuring how the software is implemented to measuring size in terms of the functions required by the user.

IEEE Std 14764-2006: Software Engineering--System Life Cycle Processes—Maintenance

Abstract: The process for managing and executing software maintenance activities is described.

IEEE Std 14764:2006 describes in greater detail management of the Maintenance Process described in IEEE Std 12207, including Amendments. It also establishes definitions for the various types of maintenance. IEEE Std 14764:2006 provides guidance that applies to planning, execution and control, review and evaluation, and closure of the Maintenance Process. The scope of IEEE Std 14764:2006 includes maintenance for multiple software products with the same maintenance resources. "Maintenance" in IEEE Std 14764:2006 means software maintenance unless otherwise stated.

IEEE Std 14764:2006 provides the framework within which generic and specific software maintenance plans may be executed, evaluated, and tailored to the maintenance scope and magnitude of given software products. It provides the framework, precise terminology and processes to allow the consistent application of technology (tools, techniques and methods) to software maintenance.

IEEE Std 14764:2006 provides guidance for the maintenance of software. The basis for the Maintenance Process and its activities comes from the definitions of IEEE Std 12207. It defines the activities and tasks of software maintenance, and provides maintenance planning requirements. It does not address the operation of software and the operational functions, e.g. backup, recovery and system administration, which are normally performed by those who operate the software.

IEEE Std 14764:2006 is written primarily for maintainers of software and additionally for those responsible for development and quality assurance. It may also be used by acquirers and users of systems containing software who may provide inputs to the maintenance plan.

IEEE Std 15288-2008: Systems and Software Engineering —Software Life Cycle Processes

Abstract: This International Standard establishes a common process framework for describing the life cycle of man-made systems. It defines a set of processes and associated terminology for the full life cycle, including conception, development, production, utilization, support and retirement. This standard also supports the definition, control, assessment, and improvement of these processes. These processes can be applied concurrently, iteratively, and recursively to a system and its elements throughout the life cycle of a system.

The IEEE Software Engineering Body of Knowledge (SWEBOK)

In this Guide, the IEEE Computer Society establishes for the first time a baseline for the body of knowledge for the field of software engineering, and the work partially fulfills the Society's responsibility to promote the advancement of both theory and practice in this field. In so doing, the Society has been guided by the experience of disciplines with longer histories but was not bound either by their problems or their solutions.

It should be noted that the Guide does not purport to define the body of knowledge but rather to serve as a compendium and guide to the body of knowledge that has been developing and evolving over the past four decades. Furthermore, this body of knowledge is not static. The Guide must, necessarily, develop and evolve as software engineering matures. It nevertheless constitutes a valuable element of the software engineering infrastructure.[6]

The purpose of the Guide to the Software Engineering Body of Knowledge is to provide a consensually validated characterization of the bounds of the software engineering discipline and to provide a topical access to the Body of Knowledge supporting that discipline. The Body of Knowledge is subdivided into ten software engineering Knowledge Areas (KA) plus an additional chapter providing an overview of the KAs of strongly related disciplines. The descriptions of the KAs are designed to discriminate among the various important concepts, permitting readers to find their way quickly to subjects of interest. Upon finding a subject, readers are referred to key papers or book chapters selected because they succinctly present the knowledge.

The Guide is oriented toward a variety of audiences, all over the world. It aims to serve public and private organizations in need of a consistent view of software engineering for defining education and training requirements, classifying jobs, developing performance evaluation policies, or specifying software development tasks. It also addresses practicing, or managing, software engineers and the officials responsible for making public policy regarding licensing and professional guidelines. In addition, professional societies and educators defining the certification rules, accreditation policies for university curricula, and guidelines for professional practice will benefit from SWEBOK, as well as the students learning the software engineering profession and educators and trainers engaged in defining curricula and course content.[7]

[6] Guide to SWEBOK 2004 Foreword
[7] Guide to SWEBOK 2004 Preface