Harmonization of Software Engineering and System Engineering Standards

Raghu Singh
US Department of the Navy
Space and Naval Warfare Systems Command

Abstract

Software is increasingly performing vital functions in systems, such as consumer products, military, telecommunication, medical, and banking. However, software engineering and conventional engineering have not been integrated into a cohesive and concurrent system engineering. Consequently, software engineering is essentially developing on its own and does not fully participate in hardware-software trade-off analyses and does not fully contribute to the system at its full potential. This article presents such system-software relationship issues and presents some initial solutions that might be implemented to improve the situation.

1. Introduction

Since the "cottage" industry era of the pre-1980's, software has been steadily gaining in importance as it increasingly performs vital functions not only in information technology systems but in conventional systems as well, such as defense, transportation, medical care, and banking. Typically, such systems are diverse, complex and interdisciplinary and are configured with components, such as hardware, software, material, information, facility, and personnel. In order for a system to perform correctly, these components, including software, must contribute correctly, consistently and cooperatively. It would be difficult for software to be effective without its associated system context. Software characteristics can be specified and verified only in the context of the parent system. Software engineering standards need to be harmonized with those of other areas of system engineering. Some examples of the system areas are design, fabrication, safety, dependability, ergonomics, reliability, and quality assurance.

Software engineering standards, despite their importance, have not been harmonized yet with those in system engineering. There are several factors that may be hindering this harmonization. One key factor is the current state of system engineering, that is not conducive enough for the harmonization to occur in a routine manner. In all probabilities, existing standards in system engineering cover specific areas, address local needs, and are not well harmonized themselves. For the harmonization of software and system engineering standards to succeed, the area standards in systems engineering need to be harmonized as well. It would be ideal if software standards are harmonized concurrently. This approach would not only harmonize software and system engineering standards, but would, more importantly, integrate the two disciplines. This and other hinderance factors need to be addressed and tackled for the harmonization (and the integration) to occur -- and to succeed.

Of special concern to the engineering communities is the impact of absence of this harmonization. Software engineering, like system engineering, crosses several boundaries, including those of information technology and conventional systems. See Figure 1. Without a standard integrated system engineering context, software standards would not be fully effective in producing quality software for those systems (information technology or conventional) that contain hardware and software. Besides, software and hardware would diverge even further without an embracing system engineering discipline. The harmonization (and the integration) of software engineering and system engineering standards has to be a deliberate effort and needs to commence without delay.

The Sub-Committee 7 (SC7) (Software Engineering) of the Joint Technical Committee 1 (JTC1) (Information Technology) of the International Organization for Standardization (ISO) and International Electrotechnical Committee (IEC) has begun to address these harmonization issues. At its annual plenary in Ottawa, Canada in June 1994, JTC1/SC7 adopted a resolution to address software-system relationships. For that purpose,
it established an ad hoc study team to address the relationships and develop recommendations for improving them. The study team addressed the relationships and prepared a report for JTC1 [1]. This team determined the relationships to be lacking and in need of immediate attention and improvement.

This article is based on the aforementioned JTC1/SC7 report and addresses the status of the harmonization issues and provides general recommendations.

2. Definitions

2.1 System

An integrated composite that consists of one or more of the processes, hardware, software, facilities, and people, that provides a capability to satisfy a stated need or objective. [2]

Note: The term covers those systems that are created by humans.

2.2 Software

A combination of computer instructions and computer data definitions, that enable the computer to perform computational or control functions. [3]

2.3 System engineering

An interdisciplinary collaborative approach to derive, evolve, and verify a life-cycle balanced system solution which satisfies customer expectations and meets public acceptability. [4]

2.4 Software engineering

The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software, that is, the application of engineering to software. [5]

3. Relationships between software and system engineering

Experiences with the development of ISO/IEC 12207 [2], DOD-STD-2167A [3], MIL-STD-498 [6], and MIL-STD-499B [7] revealed several issues related to software-system relationships, system aspects of software, and the current state of system engineering. The following summarizes these issues:
a. Software itself may be a system, but in application will always be a part of a system. System is a term for description of the structure of an object; software is a medium for realization of an object. Both terms may be applicable to the same object. Standards (including laws and regulations) for systems, therefore, always have to be consistently applied to software, and those for software have to be analyzed for their applicability on the system level. See Figure 2.

b. Software cannot be active by itself. Essential characteristics of software can be identified, specified, and verified only in association with other components of a higher-level system. Values of these characteristics will usually depend on those of the higher-level system and its components.

c. The life cycle of software is very much intertwined with the life cycle of its parent system. There are no clearly marked beginning and end points for software life cycle in a system life cycle.

d. There is often incoherence between the phases of system development and software development. Non-participation of software engineers during system engineering is risk prone and a cost driver. Related follow-on corrective actions are essentially ineffective.

e. Conducting software engineering isolated from system engineering will invariably lead to software that will not fulfill its assigned functions correctly and cooperatively and will adversely impact the [total] system performance.

f. Typically a system-level standard is oriented to conventional engineering, with little, if any, consideration for software. Generally such a system-level standard is more related to a manufacturing environment than to a development environment.

g. Almost invariably, software engineering and conventional engineering have been marching their separate ways, resulting in essentially two separate but parallel cultures. Typically, these the two cultures interfere with each other to the detriment of total system performance. Areas of pronounced differences are terminologies, design representation, quality management, operations, maintenance, corrective actions, compliance, and metrics.

h. In several aspects, literal or forced application of conventional-engineering concepts to software engineering is misleading, unrealistic and not meaningful. (The converse is also true.) An example: reliability measures.

i. Conventional engineering has the physical sciences as its foundation, while software engineering does not have such a foundation. Software engineering is significantly more labor-intensive than conventional engineering.

j. Conventional engineering is much older and practically well-established. Software engineering (the terms were first coined in 1968) is relatively very new, changes rapidly, and is still maturing. Standard practices for development, production, application, and maintenance of hardware have grown up during a long period on the basis of experience; while those of software are developed mainly without traditional basis by means of systematic considerations. (Standards and regulations that are applicable to both hardware and software are yet to be developed.)

k. Procedures in software engineering are not unambiguous. (Explanation: If a project manager has a set of design prints for a hardware device and distributes them to three design teams for implementation, the three resulting devices would be very similar, if not identical, in form, fit and function. This situation is typically not true in case of software.)

During the development of the aforementioned software standards, literature searches were conducted for a top-level system engineering standard that could be used as a framework, within which the software life-cycle process would be defined and established. No such standards were found! (A summary of the state of system engineering is presented later.) Therefore, it became necessary to define a generic, narrow spectrum of system life cycle, within which the software life-cycle processes were placed. None of these software standards provides for hardware and other considerations during design, development and testing. Such ad hoc software-system
4. System engineering and its current state

There are isolated standards addressing specialty areas of system engineering, such as safety, quality, configuration management, software, etc. Organizations that have developed system-level standards or guides in the system areas of their needs and interest are: ISO, IEC, Institute of Electrical and Electronics Engineers (IEEE), US Department of Defense (DoD), US National Aeronautics and Space Administration (NASA), European Space Agency (ESA), and North Atlantic Treaty Organization (NATO) [7, 8, 9, 10, 11]. These system-level standards come from different national, international, and other organizational bodies and are born of local needs. Some areas of a system are not covered by existing standards. Some standards address inappropriate system areas. In some cases, these standards have been marching with their unique terminologies and procedures. Consequently, there are multiplicities, overlaps, and inconsistencies among these system-level standards. These unique, local, or self-serving standards furnish, at the best, a fragmented approach to system engineering.

There is a general trend with system-level standards. Such standards are derived from hardware manufacturing experience and mostly applicable to that part of the life cycle. Standards that address the total life cycle of a system are unavailable or unknown.

There is a growing realization among the engineering communities that an embracing system life cycle framework is needed. From economic viewpoints, system managers are reaching similar realization. However, a forum for addressing these system-level issues and needs is not yet available or not known.

5. Activities for harmonizing software and system engineering

This section establishes some basic technical activities that need to be conducted for harmonizing and strengthening the relationships between software engineering and system engineering. The activities are listed below:

a. Develop the following basic items that would lay the foundations for system engineering standardization (to include software engineering and ISO/IEC 12207 [2]):

(1) Classification of systems according to key characteristics of their components; for example, equipment, plant, organization, staff, etc.
(2) Classification of systems according to their complexity and definitions of minimum levels of complexity; for example, number of different types of components, criteria for the identification of an object to be a system, etc.

(3) Definition of typical characteristics for systems, such as in ISO/IEC 9126 [12].

(4) Definition of criteria for correlating characteristics to the system or one of its components; for example, time behavior of data-processing systems to system or a given hardware or software.

(5) Stipulation of rules for presentation of systems to be functional units (composed of lower-level functional units), to be physical units (composed of lower-level physical units), and for the composition of functional units from physical units.

b. Conduct a survey of existing standards to determine which areas of system have been addressed and how much.

c. Harmonize/consolidate a set of existing standards as needed. Identify new areas for integration or future standardization. The top-level system framework should integrate these areas, not necessarily subsuming them.

d. Develop a top-level process model for system engineering. The process model must address hardware, software, and manual operations. The term "hardware" includes equipment that may belong to subsystems other than information technology systems. The development of the process model may be based on generalization of the ISO/IEC 12207 [2] processes or should include them as a subprocess.

e. Stipulate tasks and requirements for system engineers; for example, supposition of knowledge of applying the system or interfaces and cooperation of sub-systems.

f. Establish liaisons with key ISO/IEC and other standardization bodies that are working on specific areas of system engineering.

6. General recommendations

1. An embracing system life cycle framework should be defined, that would cover primarily hardware, computers, software, and personnel.

2. It should be determined further whether the software life cycle processes could be integrated into the system life cycle framework. If the feasibility for integration is indicated, the software life cycle processes should be integrated into the system life cycle framework.

3. ISO/IEC 12207 [2] should be harmonized (and/or integrated) with the system life cycle.

4. An organization, preferably in the commercial sector, that has experience in standardizing the software processes and system areas should lead these harmonization and integration efforts. Ideally, the JTC1 should sponsor and lead these efforts. An effort at the international, JTC1 level would benefit the world community as a whole.

7. Further Reading

Reference 13 discusses current issues related to the integration of software engineering and system engineering.

8. References

1. ISO/IEC JTC1/SC7; Report of the JTC1/SC7 ad hoc Study Team on software-system relationships; Doc. # N1331; Feb 1995.

2. ISO/IEC 12207 (DIS); Software Life-Cycle Processes; 1993.

3. DOD-STD-2167A; Software Development.


7. MIL-STD-499B (Draft); Systems Engineering; US Department of Defense.
8. PSS-01-40, Issue 2; System Safety Requirements for ESA Space Systems and Associated Equipment.

9. AQAP 100; General Guidance on NATO Quality Assurance.


12. ISO/IEC 9126; Software Product Evaluations.