SCITECH
RESEARCH ORGANISATION|

Volume 4, Issue 3
October 2, 2015

Journal of Information Sciences and Computing Technologies
www.scitecresearch.com/journals

Intertwined Relationships between Systems Analysis & Design and Software Engineering

Mohammad A. Rob
Management Information Systems
University of Houston-Clear Lake, Houston, Texas, USA

Abstract: Systems Analysis & Design (SA&D) and Software Engineering (SE) are two comparable subjects taught in two different educational disciplines. SA&D is focused on developing an information system, while SE is focused on developing a software. Although only one course exists on SA&D, various courses are available in the field of Software Engineering. However, both of them discuss the same process models to develop a system or software. An information system cannot be built without building a software. So why the need exist to address various subject areas within Software Engineering as compared to a single course in SA&D? What is the relationship between the two subject areas and what are differences? Our analysis shows that although the process models are same in both cases; however, focusing on the 'quality' of software is the key factor in the area of software engineering, whereas the organization of the chapters according to the water fall model is a major factor in the SA&D texts. We have identified that in addition to selecting a process model, a software engineer needs to focus on five Quality Management metrics (5QMMs) to develop a quality software. Furthermore, traditionally, SA&D was focused on developing business systems while SE was focused on developing large, complex, embedded, and real-time systems. However, this trend is changing in today's software market, which is dominated by large business systems rather than traditional technical and governmental systems. As such, both SA&D and SE texts are verging towards a common direction to address the complex business needs of the information systems where big dollars are spent in the ever-growing market of demand and supply. Thus the SE texts should continue to address the topics that relate to large business systems including databases and web technologies. On the other hand, SA&D texts should continue to adapt many software engineering quality concepts and bring them in par with the SE texts.

Keywords: Systems Analysis and Design; Software Engineering; SDLC; Waterfall; Process Model; Quality.

1. Introduction

Systems Analysis & Design (SA&D) is a course typically taught within a Management Information Systems program, whereas Software Engineering (SE) could be a course or a stand-alone program in many universities. SA&D is focused on developing an information system, while software engineering is focused on developing a software. Since software itself is a major part of an information system, in order to build an information system, we must develop the software nonetheless. With this in mind, the question arises that why a whole software engineering program with many courses is necessary, in comparison to just one course on systems analysis and design? What is their connection? How do they differ and where do they overlap? This paper tries to address some of these intertwined issues relating the two subject areas. To the best of our knowledge, this issue has not been addressed before and there are knowledge able research papers available.

2. Methodology

First two leading textbooks from two subject areas [2, 7] are compared chapter by chapter, then by the important topics covered, such as the process models, project management, quality, COCOMO, Use Case, Function Points, Capability Maturity Model, Six Sigma, ISO 9000, Object-Oriented Design, and etc. A list of important topics is then created from the two texts. Then other text books are included to find the inclusion or exclusion of each of the topics. Thus a matrix is created with rows as topics and columns as textbooks, and then check marks are used for the inclusion of the topics in each text book. Studying the matrix, a comparative knowledge is developed on the two subject areas. Further study included detailed understanding of some key topics from research papers and web search. Comparative results are briefly presented in the following and a model is developed based on our findings.

3. Results

3.1. Defining Systems Analysis & Design and Software Engineering

A suitable definition for the Systems Analysis & Design is very difficult to find, although there are more than ten leading textbooks on the subject in the market [1 - 6]. A search for the subject leads to the definition of Systems Development Life Cycle or SDLC. However, according to one of the textbook authors, Hoffer et al. [1] defines systems analysis & design as “a complex, challenging, and stimulating organizational process that a team of business and system professionals uses to develop and maintain computer-based information systems.”Software Engineering is loosely defined as the application of engineering concepts, techniques, and methods to the development of software systems[7 - 10]. It is clear that in both cases, certain techniques, methods or processes are applied to develop a system or software. However, one of the definitions focuses on the term “business” and the other stresses on “engineering.”As we know, business and engineering are two well-established disciplines with clear distinctions to each. So where do SA&D and SE intersect and where do they diverge?

3.2. The Convergence and Divergence

SDLC which stands for System (or Software) Development Life Cycle (SDLC) is a phased approach of developing a system or software. The names and the number of phases outlined in two different subjects or in various textbooks might be different, but the underlying steps to develop a system or software are the same: starting with the concept of a new or upgrading to a system or software, followed by requirements gathering, feasibility study, requirements analysis, design, implementation, testing, installation and maintenance. The process models are same for both SA&D and SE: waterfall, iterative, spiral, Agile, and others are applicable to both. However, pioneers of the models are software developers [11-14]. So before the systems analysis & design, there was software engineering. However, both textbooks dedicates majority of its chapters discussing about the analysis and design of the software. Recently, some SA&D textbooks have been including some traditional software engineering topics such as COCOMO, Functions Points, CMM, Use Cases, UML, and Object-Oriented Analysis & Design. Also almost all SA&D texts include at least one chapter on project management, while SE typically includes multiple chapters on project management focusing on four P’s: People, Product, Process, and Project [7].

While SA&D focuses on business information systems, software engineering traditionally focuses on technical, real-time, embedded and mission critical systems of all domains. There is a clear focus on database and network in SA&D texts, while SE texts clearly focus on software architecture. The field of software engineering is also distinct for the scale of the systems it addresses[15]. In software engineering, there is a special emphasis on working on very large software projects, such as large enterprise or government projects that might require a team of 50 people to write the code. SA&D prepares a Systems Analyst, while Software Engineering prepares a Software Engineer or Software Architect. A Systems Analyst works with the customers and developers in defining and designing a system, while a software architect focuses on the nuts and bolts of the software from top to bottom to create an architecture for the software that fits with the specific hardware or domain.

Most SA&D textbook chapters are highly organized by the SDLC phases and activities following the traditional Waterfall model, while a basic software engineering text takes a holistic approach to deliver the knowledge in various groups such as management of software projects and methods of software engineering. A Systems Analysis & Design text is focused on one course, and it can be compared with a basic Software Engineering text; however the later can be divided into many sub-disciplines or courses.

3.3. Why Software Development is an Engineering Method?

According to Wikipedia, Software engineering is the study and application of engineering methods to the design, development, and maintenance of software. As such, it applies a systematic, disciplined, and quantifiable approach to the development, operation, and maintenance of software as with any other consumer product. Furthermore, it is an engineering discipline concerning all aspects of software production. It also establishes and uses sound engineering principles in order to economically obtain software that is reliable and works efficiently on real machines as well. The discipline of software engineering was created to address the poor quality of software, secure projects exceeding time and budget constraints, and ensure that software is built systematically, rigorously, measurably, on time, on budget, and within specifications[11, 12]. Engineering already addresses all these issues, hence the same principles used in engineering can be applied to software.

Almost all of the attributes mentioned above regarding SE can be grouped into two general areas: process model and project management, which are also discussed in all systems analysis and design texts. So what is the “big deal” about “engineering” here? In fact project management is defined as an art and science of managing the scope, time, cost and quality of a software, and no two software projects can be considered the same. Early researchers like Barry Boehm [11] and Alan Albrecht [12] designed cost estimation models for software that became the de facto standards in software size calculation; however, they are just estimations, not perfect science or engineering that can be considered universal to create multiple copies of a product in a repeatable fashion. Similar arguments apply for process models. So many models have been proposed over time, and still continue, in order to manage software developmental metrics.

3.4. The Quality Focus in Software Engineering

In project management, we typically focus on three inter-related constraints: scope, time and cost. But for software or rather any kind of consumer product, quality is the most important factor. Many times quality is considered as the fourth constraint in project management. For software engineering, quality comes above all else. One can spend all the effort, time, and money, but if the product is not of good quality, it will not be acceptable to the users in the long run.

In developing a system or software as a product, both SA&D and Software Engineering uses the same process model of SDLC that goes through various phases such as problem definition, requirements gathering, planning, analysis, design, implementation, testing, deployment, and support. However, in Software Engineering, the major focus is on “quality,” and the quality is ensured by the management of several software developmental metrics such as quality management, project management, requirements management, risk management, and configuration management. Management of these metrics ensures quality software production. This is illustrated in Figure 1.The two inner circles apply to both system and software development, but the majority of the outer circle containing various management metrics applies to software engineering. Each of these metrics is a subject or course by itself. To summarize, in order to develop a quality product, we use a particular process model and focus on various management metrics to develop that quality product.

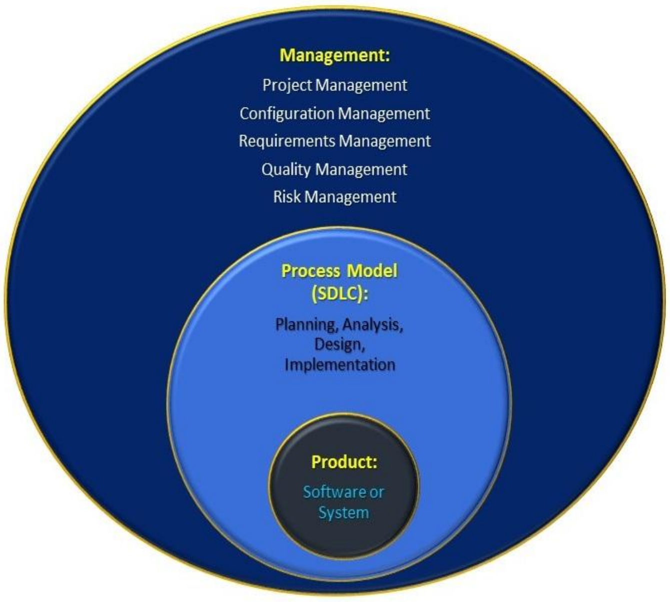

Figure1: Five Quality Management Metrics in System or Software Development

3.5.5 QMMs: The Five Quality Management Metrics in Software Engineering

The five quality management metrics mentioned above and widely discussed in the software engineering literature, are briefly defined in the following:

- **Software Quality Management:** The aim of Software Quality Management (SQM) is to manage the quality of software and its development process. A quality product is one which meets its requirements and satisfies the user. Software quality management processes include software quality assurance, software quality plan, and software quality control.
- **Software Project Management:** The art and science of planning and leading software development projects, which includes project planning, scheduling of tasks, as well as monitoring and controlling the tasks and people during the software development life cycle. Even though people are the most important factor in the system or software development project, however, their management comes under the project manager.
- **Software Requirements Management:** The purpose of requirements management is to ensure that an organization documents, verifies, and meets the needs and expectations of its customers and stakeholders. It is the process of documenting, analyzing, tracing, prioritizing and agreeing on requirements and then controlling change and communicating to relevant stakeholders.

- **Software Risk Management:** Entails the process of identification, assessment, and prioritization of risks or uncertainties, followed by coordinated and economical application of resources to minimize, monitor, and control the probability and/or impact of unfortunate events.
- **Software Configuration Management:** Involves the process of identifying, documenting, analyzing, and controlling the scope of the product, as well as its changes; and enable communication among relevant stakeholders.

3.6. Relationship between Capability Maturity Model and Software Quality

The Capability Maturity Model (CMM) suggests that the development and maintenance of software should be systematic, disciplined, and quantified[16]. The identity of a quality culture is an organizational environment where quality is viewed as everyone's responsibility. Software development is not just about developing a piece of quality software, it is a continuous improvement process of an organization that develops software. Only certain organizations that can attain a certain level of process maturity can develop certain quality of software – it involves continuous research and development effort, much like production of a quality product within a manufacturing company. Five levels of CMM developed by Watts Humphrey becomes the de facto standard to evaluate the capability of a software development company, and the U.S. government uses this standard to evaluate its software contractor(s)[16]. To achieve a level beyond the first chaotic level, it requires an organization to focus on certain Key Process Areas (KPAs). Without delving into the details of the KPAs, we would like to mention that the requirement of the process model and the Five Quality Management Metrics (5QMMs) mentioned in Figure 1 can be broken down into Level 2 and 3 key process areas.

A Software Engineering text or program typically covers CMM in detail. The question follows then, how much of that is covered within a SA&D text? Recent editions of some SA&D textbooks are starting to mention or are adding a section on CMM, and is rightfully mentioned by one of the authors that effective Project Management is essential for an organization to achieve CMM Level 2 and application of a repeatable process model can lead to CMM Level 3 [6]. Although both the Project Management and the Process Models are covered in all SA&D texts, neither are any management metrics mentioned above, nor are all the KPAs of CMM Level 2 and 3 discussed in the SA&D textbooks.

3.7. Software Engineering as a Program

As mentioned before, a Software Engineering process can be outlined in one textbook or it can be divided into many sub-disciplines or courses. A typical Software Engineering program may include part or all of the knowledge areas covered in the Software Engineering Knowledge book (SWEBOK) developed by the *IEEE Computer Society* and they include the quality metrics for software development mentioned before [17]. These include: *Requirements Engineering, Software Architecture &Design, Software Construction, Software Testing, Software Maintenance, Software Configuration Management, Software Engineering Management, Software Engineering Process, Software Engineering Tools and Methods, and Software Quality Management* [18].All these courses are developed to address one issue - that is to develop a quality software.

3.8. Changing Nature of SA&D and SE Textbooks

In recent years, most SA&D text books are adapting some traditional software engineering concepts such as CMM, Quality Assurance, Use Cases, OO Design, UML, COCOMO, Functions Points, and etc. One the other hand, SE texts are changing as well by adapting with the varying demand of the type of software used in major industries. In today’s market, most money is spent for business and web-based software development projects. Software Engineering texts are adapting to address these issues more than ever before. Table-1 provides a comparative list of important issues relating the two subject areas.

Table-1: Comparing and Contrasting SA&D and Software Engineering Texts

Systems Analysis & Design	Software Engineering
Traditionally deals with the development of business information system including software, hardware, database, network, and user interface.	Traditionally deals with the development of complex, technical, mission critical, real-time, and embedded software in all domains.
Only one course in the Management Information Systems program.	A course or a whole program with many courses in a Software Engineering program.
Clearly prepares a Systems Analyst.	Not clearly mentioned, but it prepares a Software Engineer or Software Architect.
Most textbooks are highly organized according to the phases and activities of SDLC Waterfall model, but some are taking an approach like Software Engineering.	Most textbooks take a holistic approach in organizing chapters in the textbooks.
Focused on the Systems Development Life Cycle (SDLC). Newer editions bring many quality issues like CMM, ISO and Six Sigma.	Focused on the Software Quality and Project Management.
There is at least one chapter on Project Management.	There are multiple chapters on Project Management.
Software Engineering concepts like COCOMO, Function Points, Use Case, UML, OO design, Quality Assurance are being included in recent editions.	Business and web-based systems are being added to adapt to the changing need of the market.
May fit into the CMM Level 3, as project management and SDLC models are discussed elaborately.	Can fit into CMM level 2 and higher depending on how many KPAs are discussed.

3.9. Systems Analyst versus Software Engineer

Most SA&D textbooks are very clear in an introductory chapter defining the role of a systems analyst, the knowledge required by a systems analyst, and that the information gathered from the textbook will prepare a systems analyst as a professional. However, Software Engineering texts do not define the role of a software engineer, and the term "Engineer" generally means that an individual holds a professional license. Professional engineering licenses are specific to a state and they must be renewed at regular intervals. Software engineers typically do not hold a professional license. So, can we say that software engineers are true engineers? That is a burning question. However, starting from 2013, National Council of Examiners for Engineering and Surveying (NCEES) started a professional licensure exam on Software, thereby allowing Software Engineers to be licensed and recognized[19].

4. Conclusion

This paper tries to address some questions relating two subject areas with significant overlap: Systems Analysis & Design and Software Engineering. Our analysis shows that the "Quality" focus is the key factor in the area of software engineering, while the process models and basic project management activities are the same in both subjects. We have identified that in addition to selecting a process model, a software engineer typically focuses on five Quality Management Metrics (5QMMs) to develop a quality software. On the other hand, many software engineering concepts are being adapted to the Systems Analysis & Design texts. This trend should continue and the focus of quality should be broadened in the SA&D texts to bring them in par with the SE texts.

Furthermore, traditionally SA&D was focused on developing business systems, while SE aimed to develop large, complex, embedded, and real-time systems. However, this trend is changing in today's software market which is dominated by large business and web-based systems rather than traditional technical and governmental systems. Whether or not software development is an engineering process may still be a debatable issue, but with the progression of time, the fields of Systems Analysis & Design and Software Engineering are converging closer to each other. Their distinctions are beginning to blur, not due to engineering or business per se, but subsequent to the whim of changing market demands. Software Engineering texts should continue to address the topics that relate to large business systems including database and web technologies.

References

[1] Hoffer, J.A., George, J. F. and Valacich, J. S. (2008), *Modern Systems Analysis and Design*, Pearson Prentice-Hall Publishing.

[2] Dennis, A., Wixom, B. H. and Roth, R. M. (2009), *Systems Analysis & Design*, Publisher: John Wiley & Sons, Inc., Hoboken, NJ.

[3] Satzinger, J. W., Jackson, R. B. and Burd, S. D. (2004), *Systems Analysis and Design in a Changing World*, Thomson: Course Technology.

[4] Kendall. K. E. and Kendall, J. E. (2008), *Systems Analysis and Design*, Pearson-Prentice Hall.

[5] Shelly, G. B. and Rosenblatt, H. J. (2012), *Systems Analysis and Design*, Course Technology: Cengage Learning.

[6] Whitten, J. L., Bentley, L. D. and Dittman, K. C. (2004), *Systems Analysis and Design Methods*, McGraw-Hill.

[7] Pressman, R. S. and Maxim, B. (2014), *Software Engineering: A Practitioner's Approach*, McGraw-Hill Publishing.

[8] Sommerville, I, (2010), *Software Engineering* (9th ed.), Harlow, England: Pearson Education.

[9] Tsui, F., Karam, O. and Bernal, B. (2014), *Essentials of Software Engineering*, Jones & Bartlett Learning.

[10] Laplante,P. A. (2007), *What Every Engineer Should Know about Software Engineering*, CRC Press, Taylor and Francis Group.

[11] Boehm, B. (1981), *Software Engineering Economics*, Prentice-Hall Publishers, Englewood Cliffs, NJ.

[12] Albrecht, A. J. (1971), "Measuring Application Development Productivity," *IBM Application Development Symposium*, pp. 83-92.

[13] DeMarco, T., (1978), *Structured Analysis and System Specification*, Yourdon Press.

[14] Yourdon, E. and Constantine, L. L. (1979), *Structured Design: Fundamentals of a Discipline of Computer Program and System Design*.

[15] *Software Engineering*, http://en.wikipedia.org/wiki/Software_engineering; Accessed on March 10, 2015.

[16] Humphrey, W. S. (1981), *Managing the Software Process*. Addison-Wesley Publishing, 1981.

[17] *Software Engineering Body of Knowledge*, IEEE Computer Society; http://en.wikipedia.org/wiki/Software_Engineering_Body_of_Knowledge; Accessed on February 10, 2105.

[18] *Software Engineering Program*, University of Houston-Clear Lake, www.uhcl.edu/softwareengineering, Accessed on January 20, 2015.

[19] *NCEES introduces PE exam for software*, http://ncees.org/about-ncees/news/ncees-introduces-pe-exam-for-software-engineering, Accessed on March 14, 2015.