The Impact of Programming Experience on Successfully Learning Systems Analysis and Design

Wang-chan Wong
wcwong@csudh.edu
Information Systems and Operations Management Department
California State University
Dominguez Hills
Carson, CA 90747, U.S.A.

Abstract

In this paper, the author reports the results of an empirical study on the relationship between a student's programming experience and their success in a traditional Systems Analysis and Design (SA&D) class where technical skills such as dataflow analysis and entity relationship data modeling are covered. While it is possible to teach these technical skills to students without programming experience, the results of the study strongly suggest that students with programming experience complete the course more successfully than those without.

Keywords: Systems Analysis and Design, Programming, IS Curriculum, Problem Solving

1. INTRODUCTION

For many of us formally trained in Systems Analysis and Design (SA&D), it is a logical assumption that some programming experience is a pre-requisite for taking a course in SA&D. However, as we have observed, many IS departments have relaxed their programming pre-requisite requirements, and this de-emphasis is reflected in the 2010 IS curriculum (Topi, et al, 2010). Thus, this current trend prompts the question: Does the lack of a programming background hinder understanding and subsequent success? To answer this, from 2007 to 2013 the author collected the homework scores of 15 SA&D classes from a total of 259 students. Statistical analysis of the data strongly supports the notion that programming experience is important for students to successfully complete the course. There have been similar studies conducted before, but they were typically carried out in a single class with limited sample size and for a short period of time. With a large sample size spanning six years, this study is more definitive and conclusive.

The rest of the paper is organized as follows. Section 2 is a literature review of teaching SA&D, particularly regarding its relationship to programming. Section 3 discusses the background of the study, while the methodology of the empirical study is explained in Section 4. The data analysis results are presented in Section 5. Finally, concluding remarks are given in Section 6.

2. LITERATURE REVIEW

There is a significant gap between teaching and research in SA&D (Bajaj, et al, 2005). This situation is reflected by the low number of research publications in this area. It is particularly true in finding research reports on teaching SA&D.

What do experts and scholars say regarding the relationship between programming and learning SA&D? Most acknowledge that programming is an essential foundation of SA&D. Yourdon and Constantine's book on Structured Design, a classic in structured analysis and design, discusses this relationship in the foreword: "…we assume that the reader knows how to code, and is capable of writing "good code"..." (Yourdon and Constantine, 1978, page xvi)

Booch suggests that his object-oriented method is "… most appropriate for courses in software engineering and advanced programming, and as a supplement to courses involving specific object-oriented programming languages" (Booch, 1994, page viii).

Rumbaugh, et al. propose Object Modeling Technique (OMT) as a method to develop object-oriented systems and to support object-oriented programming. They suggest that pre-requisites include "exposure to modern structured programming languages and a knowledge of basic computer science terms and concepts" (Rumbaugh et al., 1991, page x).

Booth's discussion of "folk pedagogies" in software engineering asserts that students should take a programming course before a design course (Booth 2001).

Similarly, in his study on integrating programming and system analysis, Guthrie concludes that programming is the chicken, and system design is the egg. He demonstrates that a student's design skill is directly related to their programming skill (Guthrie, 2004).

Studies have been conducted to determine the actual effect of a student's prior background on their design proficiency. Judith Sims-Knight conducted a small empirical experiment in which she taught high school students and computer science students object-oriented design without programming by using CRC cards. While she found that the high school students were able to adequately handle the design process (the study did not screen whether the students had programming exposure or not), they concluded that the computer science students created more complete designs and demonstrated a deeper understanding of the design process (Sims-Knight and Upchurch, 1993).

In an experiment teaching object-oriented analysis and design (OOAD) to both computer science and math students, Boberic-Krsticev et al. (2013) note that the math students had basic problems mastering the materials, such as having difficulty acquiring fundamental concepts, and even the UML terminology. The deficiency has been attributed to the fact that these students did not have object-oriented programming backgrounds. Even with the computer science students experienced in object-oriented programming, they observe that students created UML diagrams simply for the sake of modeling; both groups of students failed to make connections between the models and their implementations. Similar to Guthrie (2004), the authors recommend that teachers should illustrate implementation in an OO programming language so students may see the connection between models and their implementations.

Chen suggests that DFDs and ERDs are the most important skills an analyst can have (Chen 2006). Only students with programming backgrounds can have a greater appreciation for the design principles that enable them to analyze and design more complex systems.

Serva (1998) argues that the technical tasks in SA&D are comparable to the difficulties of managing a project to its completion. He offers SA&D classes for non-IS students, but the coverage of the course is to simulate the difficulties of management within an IS environment and not to teach formal systems analysis and design.

Ultimately, many IS instructors, professionals, and practitioners alike prefer students to have programming experience before taking an SA&D course (Stack Overflow 2013).

3. BACKGROUND AND EMPIRICAL STUDY

The studies surveyed in Section 2 support the notion of learning programming before analysis and design. However, they are mostly small-scale experiments conducted for a short period of time. In this paper, the author is reporting on a study spanning over six years with a comfortable sample size, providing a more conclusive and definitive response to the issue.

The SA&D course in this investigation is offered by a public university -- a comprehensive urban university primarily serving a metropolitan area. Information Systems, just like other business majors such as accounting, finance, marketing, etc., is a concentration of the College of Business, instead of a single major. CIS 372 Analysis and Logical Design is a required course for IS and IS-related majors such as Information Systems Security, and Global Logistics and Supply Chain Management. However, the department has dropped the programming pre-requisite for CIS 372. The only pre-requisite of CIS 372 is CIS 370, an introductory course on Information Systems Theory and Practice. CIS 370 is also a required course for all business undergraduate students. However, it does not provide any programming training besides a few Excel and ACCESS exercises with no coding required. As it is, CIS 372 has enrolled non-IS related students ranging from general business, marketing, human resources, finance, sports, entertainment and hospitality management, international business, accounting, biology, music, criminal justice, and sociology, in addition to computer science and information systems students. In fact, none of the non-IS students in this study had any experience with programming; even though IS and IS-related students are advised to take an introductory programming class before taking CIS 372, some do not follow this advice.

4. EMPIRICAL STUDY - METHODOLOGY

Topics	**Assignments***
Introduction to Analysis and Design	
Analyzing the Business Case	Analysis of a Business Case
Managing Systems Projects	Project management, Gantt Diagram in MS Project
Output and User Interface	UI Design
Requirements Modeling	Requirements Document
Data and Process Modeling	DFD diagrams in Visio and process descriptions
Data Design	ERD in Visio and data dictionary
Development Strategies	Short questions
Systems Architecture	Short questions
Managing Systems Implementation	Short questions
Managing Systems Support and Security	Short questions

*The requirements document, DFD and ERD amount to 25% of the weighted total.

Table 1: Major topics covered and student deliverables in CIS 372

CIS 372 follows the traditional Systems Analysis and Design undergraduate curriculum except that it focuses on the functional (system) approach while the object-oriented approach is covered in another course. We used *Systems Analysis & Design in a Changing World* by Satzinger, Jackson, and Burd (Satzinger et al., 2007) for several years and then changed to *Systems Analysis and Design* by Shelly and Rosenblatt (Shelly et al., 2010) in 2010. Topics and major deliverables covered in the course based on Shelly's book are shown in Table 1.

The course is not project or team-based; students work individually on homework assignments for each chapter. Since the author was interested in determining how well students without programming backgrounds can master technical skills, the homework scores of (1) the requirements document, (2) the dataflow diagram (DFD) design document, (3) the entity relationship diagram design (ERD) document, and (4) the weighted total of each student for the class were collected. The justification to select these four scores and the related hypotheses are as follows:

1. The requirements document is important in any system development, especially for non-IS students. In reality, a non-IS business person will have many opportunities to jointly develop Request for Proposals (RFP) and the requirements document with IS staff. Gaining technical writing skills will greatly improve the quality of the requirements document, RFPs, and any other project-related documentation.

2. DFD design documents are demonstrative of the technical skills that system analysts/developers need to have. These techniques train students how to capture the dynamics and behavior of a system. Needless to say, ERD is important because it trains students to capture the objects being modeled and relationships among them in the system.

3. The weighted total is the percentage of the scores a student receives. It includes all other homework assignments and exams. It is the overall measure of a student's success in this course.

In the requirements document assignment, students are asked to define both functional and non-functional requirements for certain systems. Students will generate the requirements document similar to the clausal form example as

shown in (Taylor, 2013). Grading for the requirements document is based on clarity in writing, organization, and the ability to capture the major functional and non-functional requirements of the system.
For the DFD and ERD assignments, questions are typically taken from the end of chapters in the textbook. The questions are based on small-scale business scenarios that are manageable for a single student. For the DFD assignment, students are asked to draw the context diagram and then decompose it to Level 0 and/or Level 1 diagrams in Visio. Students also need to write high-level process descriptions for each primitive DFD process. Structure charts are not covered, since students without programming experience have difficulty understanding parameter passing and functional decomposition. Grading for the DFD assignment is based on the syntactic correctness of the diagrams, the appropriate logical flow to capture the dynamics and behavior of the system based on the scenario given, and the clarity of process descriptions for the primitive processes.

For the ERD assignment, students are asked to create the crow's foot model in Visio along with its data dictionary. Grading on the ERD assignment is based on how well the student identifies the entities and their relationships, including meaningful entity names, salient attributes of these entities and appropriate domain or data types for the attributes, correctness of the relationships (cardinalities) for these entities, correct primary key identification and referential integrity enforcement, and correct modeling of the logical and physical data models using Visio.

Since any student may take CIS 372 without fulfilling a programming pre-requisite, for the purposes of this study students were asked about their programming experience during the first class meeting.

Hypotheses
The requirements document is in an itemized clausal form by grouping system specifications into categories and subcategories, mirroring the hierarchical structure of a program and the relationships of its components. In an introductory programming class, students have exposure to top-down modular design, structure programming, and/or other programming paradigms such as the object-oriented approach. They are also introduced to the three basic programming constructs: sequence, iteration, and selection. In fact, if a student masters these programming fundamentals, they will easily be able to learn data flow analysis. Furthermore, in programming class, we always emphasize program documentation. When students decompose a program into sub-modules, they need to document the interface, function descriptions, in-line comments, etc. Therefore, the following two hypotheses are posited:

Hypothesis 1 (H1): A student's requirements document assignment score is positively associated with his/her programming knowledge. Students with programming backgrounds have better scores than students without programming backgrounds.

Hypothesis 2 (H2): A student's data flow assignment score is positively associated with his/her programming knowledge. Students with programming backgrounds have better scores than students without programming backgrounds.

In a typical introductory programming course, students are exposed to basic data structures such as record, array, files, and relational databases. They will have seen how records are linked and processed. Therefore, the following hypothesis is posited:

Hypothesis 3 (H3): A student's entity relationship modeling assignment score is positively associated with his/her programming knowledge. Students with programming backgrounds have better scores than students without programming backgrounds.

Finally, if students are able to successfully manage an introductory programming class with the exposures we describe above, they will be able to smoothly transition into learning SA&D. Therefore, this gives rise to the following hypothesis:

Hypothesis 4 (H4): A student's weighted total is positively associated with his/her programming knowledge. Students with programming backgrounds have better scores than students without programming backgrounds.

The four hypotheses are depicted in Figure 1.

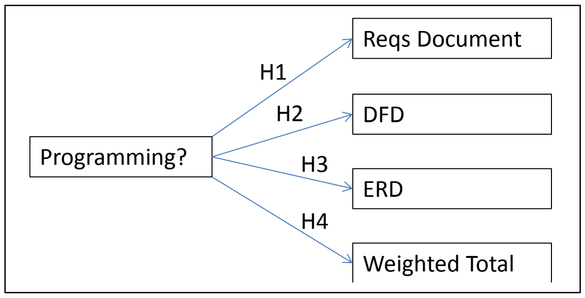

Figure 1: Hypotheses Testing

5. DATA ANALYSIS

The scores of three assignments, i.e. requirements document, DFD design, ERD design, and the weighted total of each student have been collected for the 15 classes that the author has taught since 2007. A total of 259 student records were collected, in which 10 of them had missing data and were subsequently discarded. A total of 249 of them have been used in this study. Each student record has 7 attributes. They are summarized in Table 2 below:

Student Record Structure in Analysis		
Attribute	Description	Attribute Value
ID	Identifier to the record	Integer
Class	The class that the student enrolled in.	Integer (1..15) for the 15 classes the author taught since 2007
IS?	Is the student an IS or IS-related student?	Boolean, 0=non-IS , 1= IS/IS-related
PROG?	Has the student taken any programming?	Boolean, 0=no, 1= yes
REQ	Score of the requirements document	Rounded up integer (0..100)
DFD	Score of the DFD design assignment	Rounded up integer (0..100)
ERD	Score of the ERD design assignment	Rounded up integer (0..100)
Weighted Total	The weighted average of all scores, including other homework assignments and tests of the student	Percentage (0..100%)

Table 2: Student Record Structure in the Empirical Study

The data were loaded to SPSS version 21 for statistical analysis. Among the 249 students analyzed, 114 were non-IS students, while 135 were IS-related students. All non-IS students in this sample did not have any programming background prior to CIS372, while 14 out of the 135 IS-related students did not take programming classes prior to CIS372. The summary is tabulated in Table 3 below.

		Programming?		Total
		No	Yes	
IS?	No	114	0	114
	Yes	14	121	135
Total		128	121	249

Table 3: IS/Non-IS with Programming Background Summary

Their descriptive statistics are summarized in Table 4 below.

Group Statistics					
	PROG?	N	Mean	Std. Dev	Std. Error Mean
REQ	Yes	121	78.5840	22.65005	2.05910
	No	128	69.6641	27.57173	2.43702
DFD	Yes	121	63.9669	29.03358	2.63942
	No	128	54.9375	29.82708	2.63637
ERD	Yes	121	66.1653	27.38958	2.48996
	No	128	53.6719	30.02941	2.65425
Weighted Total	Yes	121	74.3473%	13.02708%	1.18428%
	No	128	68.2066%	13.91950%	1.23032%

Table 4: Descriptive Statistics of the Empirical Study

t-test

The mean scores of the requirements document, DFD design, ERD design, and the overall weighted total in Table 4 reveals that students with programming backgrounds performed better than students without any experience. The consistency and validity of the test scores are justifiable because the assignments have similar degrees of difficulty and were graded by the same instructor for a period of six years. To further analyze the data, independent sample t-tests were conducted in SPSS where the Grouping Variable is the PROG?, and the Test Variables are the REQ, DFD, ERD, and Weighted

Total. The null hypotheses predict that the mean scores of the Test Variables are the same between students who had taken programming classes prior to taking CIS 372 and students who had not.

The t-test results are summarized in Table 5.

	t-test for Equality of Means				
	t	df	Sig. (2-tail)	Mean Difference	Std. Error Difference
REQ	2.781	247	.006	8.91996	3.20798
DFD	2.419	247	.016	9.02944	3.73338
ERD	3.424	247	.001	12.49341	3.64880
Weighted Total	3.589	247	.000	6.14071%	1.71088%

Table 5: Independent Samples t-test Results (alpha = 0.05)

The t-test results clearly support the four hypotheses shown in Figure 1. It is not surprising that students with programming experience perform better in DFD and ERD than those with no experience. After all, these two technical skills are equivalent to skills used in programming. However, it is interesting to note that students with programming knowledge also outperformed students without programming knowledge in the requirements document assignment. This may be explained by the fact that functional requirements are similar to functional and procedural descriptions in programming exercises. Requirements are written at a higher level of abstraction but are still modular in nature.

Regression Analysis

The author further performed a linear regression as a predictive model to measure the potential student completion success of the course. The dependent variable is the Weighted Total, the independent variables are the scores of the requirements document, the DFD and ERD assignment scores, and the control variables are the IS?, PROG? and Class. The regression results are summarized in Table 6.

Model Summary				
Model	R	R Square	Adjusted R Square	Std. Error of the Estimate
1	.248[a]	.061	.050	13.46328%
2	.705[b]	.497	.484	9.92102%
a. Predictors: (Constant), Prog, Class, IS				
b. Predictors: (Constant), Prog, Class, IS, REQ, DFD, ERD				

Coefficients[a]						
Model		Unstandardized Coefficients		Stand. Coeff	t	Sig.
		B	Std. Err	Beta		
1	Const	70.990	2.139		33.189	.000
	Class	-.348	.203	-.106	-1.711	.088
	IS	1.268	3.817	.046	.332	.740
	Prog	4.876	3.802	.177	1.282	.201
2	Const	42.355	2.633		16.083	.000
	Class	-.050	.152	-.015	-.331	.741
	IS	1.833	2.816	.066	.651	.516
	Prog	-.067	2.822	-.002	-.024	.981
	REQ	.118	.027	.218	4.405	.000
	DFD	.175	.023	.376	7.597	.000
	ERD	.154	.024	.329	6.474	.000
a. Dependent Variable: WeightedTotal						

Table 6: Linear Regression Analysis Results

The regression analysis results in Model 2 show that the scores of the requirements document, DFD, and ERD are all significant at the 0.05 alpha level, confirming the results of the t-test above. Hence, the regression equation is:

$$\text{WeightedTotal} = 42.355+0.175\times\text{DFD}+0.154\times\text{ERD}+0.118\times\text{REQ}$$

The coefficients of the DFD, ERD, and REQ are 17.5%, 15.4% and 11.8%, respectively. Tellingly, while these three assignments amount to only 25% of the total course requirements, the adjusted R^2 is at 0.484. This suggests that, among all other assignments and tests, the scores of these three assignments alone can explain almost 50% of a student's overall performance.

6. DISCUSSION AND CONCLUDING REMARKS

Results of the empirical study strongly suggest that students with programming experience will complete the course more successfully than those who don't have experience.

Why is programming so important in learning SA&D? In a paper written by Professor David Gries at a 1974 ACM conference, he points out that general problem-solving is very unique in teaching programming (Gries, 1974). In the same paper, Gries illustrates his arguments and summarizes a four-phase process in problem-solving proposed by Polya in 1945 (Polya, 1945). This process is nearly identical to what is known today as SDLC. See Table 7 below for the comparison.

Polya's 4-Phase Process	SDLC
Understand the Problem	Planning
	Analysis
Devise a plan	Design
Carry out the plan	Implementation
Look back	Support/Enhancement

Table 7 Polya's Problem Solving Process vs SDLC

In fact, Polya's four-phase problem-solving process already suggests the incremental and iterative approach that we currently consider best practice. In Gries' words, "In a programming course, we attempt to teach the student how to program anything that can be programmed -- that has an algorithmic solution" (Gries, 1974, page 81). This "algorithmic" discipline and training empowers students to solve programs in a systematic way. With this training, students can smoothly transition to software engineering or SA&D, in which they cope with the complexity of solving larger problems in a more conceptual and abstract manner.

Jeffries, et al. studied the processes involved in designing software, and concluded that the decomposition process is central to creating the design (Jeffries, et al., 1981). The process is similar to the stepwise refinement proposed by Wirth; decomposition and stepwise refinement are usually covered in introductory programming class (Wirth, 1971).

The recent IS 2010 curriculum excludes programming from the core requirements even though Systems Analysis and Design (2010.6) remains one of the seven core courses in the guidelines (Topi, et al, 2010). The guidelines proposed in 2010.6 have further replaced technical skills, such as structured and object-oriented approaches, with the less programming-oriented business process modeling. It is undeniable that these replaced technical skills are crucial for students intending to further develop their careers in application development and system analysis. In a recent survey conducted by *The Economist*, 38% of respondents admit that "inadequate technical skill sets" are the biggest challenges for CIOs trying to align technology use with business goals, while only 21% think "inadequate management skills" are the biggest challenge (The Economist, 2013). Another study from the U.K. reports that almost 75% of current IT leaders -- an overwhelming majority -- are unsure that the CIOs of today will still be the right people to lead IT businesses in 2018 (Nice, 2013). Nearly half (43%) of the respondents are concerned by their potential deficiency in technical skills.

Time and time again we are reminded that the critical skills of an IT manager include project management, communication, writing, etc. While we all agree that these soft skills are crucial competencies for an IT manager, they do not necessarily mandate an undergraduate IS curriculum. Realistically, most undergraduate IS students will not be hired for management positions right after graduation. A technical position is still most likely be the first job for many IS graduates. Consider the current IS/IT job markets: reports show that programming and application development is one of the top 10 hottest IT skills for 2013 and 2014 (Pratt, 2012; Brandel, 2014; Simoneau, 2014; Wakefield, 2014). Another report suggests that 60% of the surveyed companies claimed they would hire more developers in 2013 (Pratt, 2012). Most recently, *U.S. News & World Report* named computer systems analysts as second place in their ranking of 2014's best jobs (Best Jobs, 2013).

If we intend to maximize the ability of our IS students to successfully find employment, it is our responsibility to properly equip them with sufficient technical skills. For that reason, it is necessary to continue encouraging programming as a pre-requisite to Systems Analysis and Design.

7. REFERENCES

Bajaj, Akhilesh, Batra, Dinesh, Hevner, Alan, Parsons, Jeffrey and Siau, Keng. (2005). Systems Analysis and Design: Should We Be Researching What We Teach? *Communications of the Association for Information Systems* (Volume 15, 2005)478-493

Best Jobs 2013 (2013). US News. Retrieved from: http://money.usnews.com/careers/best-jobs/computer-systems-analyst

Boberic-Krsticev, Danijela and Tesendic, Danijela (2013), Experience in Teaching OOAD to Various Students, *Informatics in Education*, 2013, Vol. 12, No. 1, 43-58

Booch, Grady (1986). "Object-Oriented Development", *IEEE Transactions on Software Engineering*, Vol. SE-12, No. 2. February, 1986.

Booch, Grady (1994). Object-oriented Analysis and Design with Applications. 2nd Edition, the Benjamin/Cummings Publishing Company, Inc., 1994, page viii.

Booth, S. (2001). "Learning Computer Science and Engineering in Context." *Computer Science Education*, 11(3), 169-188.

Brandel, Mary. (2014). 8 hot IT skills for 2014. Computer World, September 23, 2013. Retrieved from: http://www.computerworld.com/s/article/9242548/8_hot_IT_skills_for_2014

Chen, Brady, 2006. Teaching Systems Analysis and Design: Bringing the Real World into the Classroom, *Information Systems Education Journal*, Volume 4, Number 84, September 27, 2006.
http://isedj.org/4/84/

Gorgone, John, Davis, Gordon, Valacich, Joseph, Topi, Heikki, Feinstein, David and Longenerecker, Herbert (2002). "IS 2002 Model Curriculum and Guidelines for Undergraduate Degree Programs in Information Systems", by ACM, AIS and AITP. Retrieved from: http://www.acm.org/education/is2002.pdf

Gries, David (1974). "What should we teach in an introductory programming course?", *SIGCSE Bull*. 6, 1 (January 1974), 81-89. DOI=10.1145/953057.810447
URL: http://doi.acm.org/10.1145/953057.810447

Guthrie, R.W. (2004). Integrating programming and systems analysis course content: resolving the chicken-or-the-egg dilemma in introductory IS courses. *Information Systems Education Journal,* 2(1).

Hammer, Michael (1990). Reengineering Work: Don't Automate, Obliterate. *Harvard Business Review*, July-August 1990.

Hodgson, Lynda, and Wynne, James (2012). "Adopting the IS 2010 Model Curriculum: A Survey of Top MIS Programs", *Southeast Decision Sciences Institute (SEDSI)* 2012 Annual Meeting, Conference Proceedings, February 2012, pp. 177-188.

IDC Research (2010). IDC Research, July 2010
http://www.enterpriseittools.com/sites/default/files/SAP_whitepaper1012.pdf

Information Week (2013). 2014 IT Budget Survey of organizations with 50 or more employees, October, 2013.
http://www.informationweek.com/strategic-cio/executive-insights-and-innovation/2014-it-budget-survey/d/d-id/1112715

IS 2010 Task Force (2009). Response Document. Retrieved from: http://cis.bentley.edu/htopi/IS2010Response_10-27-2009.pdf

Jeffries, R., Turner, A. A., Polson, P. G. & Atwood, M. E. (1981). "The processes involved in designing software", Cognitive skills and their acquisition. Edited by J. R. Anderson, Erlbaum, Hillsdale, NJ, 1981, pp. 255-283.

Keen, P. G. W. (1999). "Middle-Out Ideas," *Computerworld* (56), April 12, 1999.

Nice, Steve (2013). "Skills Gap: What Does 2018 Hold For The CIO?", Business Computing World. Retrieved from: http://www.businesscomputingworld.co.uk/skills-gap-what-does-2018-hold-for-the-cio/

Polya, G. (1945). How to Solve It. Princeton University Press, Princeton, N.J., 1945.

Pratt, Mary (2012). 10 hot IT skills for 2013. Computer World, September 24, 2012. Retrieved from: http://www.computerworld.com/s/article/9231486/10_hot_IT_skills_for_2013

Reich, B. H., and Benbasat, I. (2000). "Factors that Influence the Social Dimension of Alignment between Business and Information Technology Objectives," *MIS Quarterly* (24:1), March 2000, pp. 81-111.

Robey, Daniel (2001) "Blowing the whistle on troubled software projects", *Communications of the ACM* (44)4, pp. 87-93.

Rumbaugh, James, Balha, Michael, Lorensen, William, Eddy, Frederick , and Premerlani, William (1991). Object-Oriented Modeling and Design, Prentice-Hall, Inc., 1991. Page x in Preface.

Samson, Ted (2013). "Software developers expected to see the highest IT job growth come 2020", February 12, 2013. Retrieved from: http://www.infoworld.com/t/it-jobs/software-developers-expected-see-the-highest-it-job-growth-come-2020-212709

Satzinger, Jackson, and Burd (2007). Systems Analysis & Design In A Changing World, 4th Edition, 2007, Course Technology,

Shelly and Rosenblatt (2010). Systems Analysis and Design, Eighth Edition, Video Enhanced, Course Technology, 2010

Serva, Mark (1998). "Teaching Systems Analysis and Design to Non-IS Majors: A Management Simulation", Decision *Line*, July 1998.

Simoneau, Paul (2014). "The Top Ten IT Skills for 2014", Global Knowledge. Retrieved from: http://www.globalknowledge.com/training/generic.asp?pageid=3635

Sims-Knight, Judith and Upchurch, Richard (1993). "Teaching Object-oriented Design Without Programming: A Progress Report", *Computer Science Education*, 4, 135-156.

Stack Overflow (2013). Stack Overflow Discussion Forum: "Teaching systems analysis and design - how much programming experience is needed?" Retrieved from: http://stackoverflow.com/questions/382797/teaching-systems-analysis-and-design-how-much-programming-experience-is-needed

Sulinier, Bruce and White, Bruce (2011). "IS 2010 and ABET accreditation: an analysis of ABET-accredited information systems programs", *Journal of Information Systems Education*, Dec 22, 2011.

Taylor, Dick (2013). Requirements Document Example. Retrieved from: www.ics.uci.edu/~taylor/ICS_52_FQ02/ics52_Fall02_Req_Ver3.doc

The Economist (2013). The strategic CIO-Risks, opportunities and outcomes. The Economist Intelligence Unit Limited 2013

Topi, H., Valacich, J. S., Wright, R. T., Kaiser, K. M., Nunamaker, Jr., J.F., Sipior, J.C., and de Vreede, G.J. (2010). IS 2010 Curriculum Guidelines for Undergraduate Degree Programs in Information Systems, ACM and AIS, 2010.

Wakefield, Kylie Jane (2014). Top IT Skills for 2014. Forbes, Transformational Tech, 4/22/2014. Retrieved from: http://www.forbes.com/sites/emc/2014/04/22/top-it-skills-for-2014/

Wirth, N. (1971). "Program development by stepwise refinement" *Communications of the Association for Computing Machinery*, Vol. 14, 1971, pp. 221-227.

Yourdon, Edward, and Constantine, Larry (1978). Structured Design – Fundamentals of a Discipline of Computer Program and Systems Design, Prentice-Hall, 1978. Page xvi.