A Cross-Collegiate Analysis of Software Development Course Content

Timothy Burns
New Jersey Institute of Technology
University Heights, Newark, New Jersey 07102
tjb9295@njit.edu

Robb Klashner
New Jersey Institute of Technology
University Heights, Newark, New Jersey 07102
klashner@njit.edu

ABSTRACT
Many undergraduate IT programs recognize that their graduates will find jobs as software developers. As such, software development (analysis and design) courses are often a core requirement of IT undergraduate degree programs. This paper presents an analysis and comparison of the content of software development courses at several colleges and universities. In particular, the software development methodologies presented in the courses are examined. The comparison is based on two main criteria; the development methodologies presented through the textbook assigned to the class, and the development methodologies presented through the class lectures.

A software development methodology is "a recommended collection of phases, procedures, rules, techniques, tools, documentation, management, and training used to develop a system" (Avison 2003). There are hundreds of software development methodologies. An ongoing debate has existed for several years as to what methodology is best. Prior research has shown that no methodology is best all the time and that practitioners are modifying methodologies to suit their needs.

This research shows that IT software development courses have not been modified to reflect current trends in the practical application of development methodologies and students may not be learning the tools and techniques that they need to become successful practitioners. Key inadequacies and the associated pedagogical significance are discussed.

Categories and Subject Descriptors
D.2.10 [**Software Design**] Methodologies

General Terms
Management, Design, Reliability.

Keywords
Software Development Teaching, Software Development Methodologies

1. INTRODUCTION
The field of information technology (IT) is interdisciplinary, with applications to both hardware and software. Furthermore, the IT field is an applied discipline. Unfortunately, the divide between the IT academic and the IT practitioner is an ever-widening chasm (Fitzgerald 1997, 2003). We need to ask ourselves as educators if we are teaching our IT students what they need to know to become successful practitioners.

In this paper, we perform an analysis of the software development techniques that are currently being taught in colleges and universities. Many undergraduate IT programs recognize that their graduates will find jobs as software developers. This is evidenced by the fact that software development courses (analysis and design) are often a core requirement of the major. For this study, only those IT programs that have a required software development course have been included.

The importance of this topic is evident. In practice, there is little doubt that a software crisis exists (Brooks 1987). Research has shown that nearly a third of all software development projects are cancelled before they finish and those that do finish will be over budget (Standish Group 1994). Less than one fifth of software development projects are completed on time and on budget (Standish Group 1994). Furthermore the ones that are completed, often times do not meet the requirements of the users and/or the business in which they are implemented. Only about twenty five percent of all software development projects are considered successful (Whiting 1998).

An often-cited cause of software development project failure is the software development methodology employed. A software development methodology is a recommended collection of phases, procedures, rules, techniques, tools, documentation, management, and training used to develop a system (Avison 2003). There is a plethora of development methodologies (Fitzgerald 2003), all of which claim to be the "silver bullet" (Brooks 1987) to the software crisis. In many cases they have the potential to be the silver bullet if used in the right environment. However, research has shown that none is the silver bullet all the time (Cockburn 2002, Fitzgerald 2003). Furthermore, recently there has been much discussion on the success of new development methodologies that have been lauded by their proponents.

Research has also shown that there is a divide between the software development methodologies that are being developed and taught by academics and those used by practitioners in the field (Fitzgerald 1997, 2003). This fact becomes more disconcerting when it is coupled with the fact that the supply of IT professionals is well below the demand, particularly in the area of software development professionals (Roberts 2000). We need to

ask ourselves as educators if we are teaching our IT students what they need to know to become successful practitioners and to combat the software crisis.

This paper presents an analysis and comparison of the content of software development courses at several colleges and universities. In particular, the software development methodologies presented in the courses are examined. A key point of analysis is if the most current techniques are being taught to students. The comparison is based on two main criteria; the development methodologies presented through the textbook assigned to the class, and the development methodologies presented through the class lectures as evidenced by literature on the course.

2. RELATED RESEARCH

Software development methodologies can be classified several ways (Avison 2003, Fowler 2002). Examples are by methodological era, by approach, or by degree of agility. Software development methodologies have progressed through several distinct eras (Avison 2003). The methodological era approach would categorize the initial era as a period when no methodologies were used. During this period, hardware limitations placed limits on the size and complexity of software systems allowing systems to be effectively developed without methodologies.

The advent of more powerful computers in the 1960's and 1970's was a precursor to more sophisticated and complex software systems. This era marked an early methodology period where a software development life cycle (SDLC) such as the waterfall approach (Royce 1970) was predominant (Avison 2003). Under this approach the project is divided into several distinct stages to be completed in a linear progression. While the waterfall approach was a significant step in formalizing software development, it soon became apparent that there were many inherent shortcomings to this methodology. Chief among the shortcomings is that the lack of flexibility can allow design flaws to not be discovered until late in the development cycle (Racoon 1997). Changes made late in the development cycle are often more costly (Beck 2000). Another shortcoming is the lack of planning for future changes in the requirements of the software, which are almost always present (Highsmith 2000).

In response to the shortcomings of the traditional SDLC, the methodology era was born. During this era many new methodologies were introduced. Methodologies from this era can be classified according to approach. Significant approaches include structured, data-oriented, process-oriented, prototyping, participative, object-oriented, and systems (Avison 2003).

Many people argue that in recent years we have entered a post-methodology era (Avison 2003, Fowler 2002). This era is marked by researchers and practitioners questioning the philosophies of the methodologies from the methodology era. The most serious criticism of these methodologies is that they are bureaucratic and labor intensive (Fowler 2002). For this reason, these earlier methodologies have recently been classified as the "heavy" methodologies (Fowler 2002). In response to this, several new methodologies have been introduced in this post-methodology period. These new methodologies are referred to as lightweight or agile methodologies because of the lack of bureaucratic contingencies inherent to the heavy methodologies (Fowler 2002).

These agile methodologies are considered by many to be "a" methodological (i.e., a negative construct that connotes an open set of attributes that are essentially not methodical) (Truex et al 2000). They are distinguished by their emphasis on adjusting to the project environment through adaptation and have less emphasis on prescribing a plan to follow (Fowler 2002). The biggest criticism of the agile methodologies has been the lack of empirical evidence supporting the claims of their benefits and their lack of theoretical foundation (Abrahamsson et al 2003). However, there is a growing body of literature both supporting and repudiating the claims of success of the agile methodologies (Abrahamsson et al 2003, Choi et al 2002).

3. METHOD

Sixty colleges and universities in the United States were randomly selected and used to provide data for this study. Care was taken to make sure that public and private, as well as small, medium, and large institutions were chosen from several geographic areas. Each of the institutions selected has an IT undergraduate program that has at least one required software development (analysis and design) course. Some of the programs have more than one required software development classes. In that case, data from both courses was combined and included in the study.

Applying a grounded theory approach (Glaser and Strauss, 1967), the data for this study was gathered using three methods and inductive results were obtained. The first method was to review the course catalog description. This served two purposes. First, it confirmed that the course matched in nature the type of course we wanted to include in this study. Second, in some cases it provided data as to what development methodologies were covered by the course.

The second data collection method was to ascertain the textbook that is used for the class. Several of the institutions provide the textbook assigned to a class via their websites. If the website did not provide this information, then instructors were contacted via e-mail and asked to provide that information. The textbooks identified were reviewed as to their content and the methodologies covered by the book were recorded.

The third source of data was from the course syllabi. In many cases the syllabi were gathered from the course web sites. If a syllabus was not available via the Internet then the instructor was contacted via email and asked to provide a course syllabus and/or to provide information as to what development methodologies were covered in the class.

4. RESULTS

We were able to identify the textbooks used in thirty of the sixty institutions that were reviewed. Table one provides a summary of that data. The table is presented in descending order by the number of times that the book was used out of the thirty courses identified. The book by Satzinger et al was the most often used text, as it was used in seven of the courses. Table one also identifies the author(s), the name of the text, and the current edition of the book.

Each textbook was reviewed and categorized according to its content. Categories that were identified included the primary development methodology that the book focuses on, other traditional methodologies covered (or at least mentioned), agile methodologies covered or mentioned, and whether or not the

Table 1. Summary of Textbooks Used

# Times Used	Author	Name	Curr Ed	Focus	Other Traditional	Agile	Open Source
7	Satzinger, Jackson, Burd	Systems Analysis & Design in a Changing World	3	Object Oriented / SDLC	SDLC, Spiral, RAD, JAD, Prototyping	XP, Agile Modeling	No
6	Whitten, Bentley, and Dittman	Systems Analysis and Design Methods	6	All	SDLC, OO, JAD, RAD, Prototyping	XP, Scrum, Agile Modeling	Yes
4	Shelly, Cashman, & Rosenblatt	Systems Analysis and Design	6	SDLC	RAD,JAD, Prototyping, OO		No
3	Kendall and Kendall	Systems Analysis and Design	6	SDLC	OO, JAD, RAD, Prototyping	XP, Scrum, Agile Modeling	Yes
2	Dennis and Wixom	Systems Analysis and Design	2	SDLC	Waterfall, OO, RAD, JAD, Prototyping	XP, Scrum, DSDM	No
2	Dennis, Wixom, and Tegarden	Systems Analysis and Design: An Object-Oriented Approach with UML	2	Object Oriented	Waterfall, RAD, JAD, Prototyping	XP, Scrum, DSDM	No
2	Hoffer, George, and Valacich	Modern Systems Analysis and Design	4	SDLC	OO, JAD, RAD, Prototyping	XP, Scrum, FDD, Crystal, Adaptive	No
1	Brown	An Introduction to Object-Oriented Analysis: Objects in Plain English	2	Object Oriented			No
1	Larman	Applying UML and Patterns	2	Object Oriented			No
1	Schach	Introduction To Object-Oriented Analysis and Design	1	Object Oriented	SDLC	XP	Yes
1	Valacich, George, and Hoffer	Essentials of Systems Analysis and Design	2	SDLC	Prototyping, RAD, JAD, PD,OO		No

book discusses the open source development paradigm. Table one shows each of these categories for the textbooks listed.

Once the textbooks had been categorized, the data was then aggregated by methodologies covered. Table two represents the results. The first column of table two lists the methodologies and categorizes them into SDLC/waterfall, object oriented, other traditional methodologies (spiral, prototyping, RAD, JAD, etc.), agile methodologies, and open source development. The second column lists the total number of institutions out of the thirty that cover the related methodology in column one. The final column displays the percent of the thirty institutions that cover the methodology based on the content of the textbook selected for the course.

Table 2. Methodologies Covered Based on Textbook Selected (N=30)

Methodology	Number of Institutions	Percent
SDLC	28	93
Object Oriented	30	100
Other Traditional	27	90
Agile Methodologies	23	77
Open Source	10	33

The second focus of this study was on the methodologies covered based on the course literature. Table three summarizes that data. We were able to gather distinct data from thirty of the sixty institutions sampled. The first column of table three lists the methodologies and categorizes them into SDLC/waterfall, object oriented, other traditional methodologies (spiral, prototyping, RAD, JAD, etc.), agile methodologies, and open source development. The second column lists the total number of institutions out of the thirty that cover the related methodology in column one. The final column displays the percent of the thirty institutions that cover the methodology.

Table 3. Methodologies Covered Based on Course Literature (N=30)

Methodology	Number of Institutions	Percent
SDLC	20	67
Object Oriented	17	57
Other Traditional	14	47
Agile Methodologies	2	7
Open Source	0	0

5. DISCUSSION

An analysis of the textbook data reveals that the authors of the most often used textbooks have updated their recent editions and reflect current trends in system development methodologies. A review of table one shows that textbooks almost always cover the SDLC, Object Oriented, and other traditional methodologies. Seven out of the eleven textbooks also cover the agile methodologies. Open sources development is somewhat neglected as it is only covered by two out of the eleven books. This number is somewhat surprising as the case can be made that open source development addresses many aspects of the software crisis, in that reliable, high quality software may be produced quickly and inexpensively (Feller 2000).

It appears that colleges and universities have succeeded in selecting textbooks that sufficiently cover development methodologies. Ninety percent of the participating institutions cover SDLC, object-oriented, and other traditional methodologies in their software development courses, based on the textbook selected. Another three fourths expose their students to the agile methodologies. However selection of a textbook does not necessarily reflect the actual material covered by the instructor of the course.

A review of the course literature tells a different story. An analysis of table three reveals that a third of the participating institutions do not cover SDLC and nearly half do not cover object-oriented or any of the other traditional methodologies, according to course materials. Furthermore, there is almost a complete lack of reference to the agile methodologies and the open source paradigm. These results appear to be in contradiction to the textbook authors who, while continuing to emphasize the SDLC and object-oriented methodologies, have updated their texts to include the more current methodologies. It appears that course instructors have not done this.

6. CONCLUSION

The objective of this research was to establish if collegiate software development courses are providing students with the tools that they need to be successful software developers. Several factors go into becoming a successful software developer (Roberts 2000). A formal education or lack thereof, is not necessarily an indicator of whether an individual will be a successful developer (as evidenced by such success stories as Bill Gates and Steve Jobs) (Roberts 2000).

However, if we accept the definition of IT as an applied discipline, then one of the objectives must be to improve its practice (Phillips 1998). Some argue that an applied discipline like IT should be conducted like other applied disciplines, for instance, such as is done in the medical profession (Moody 2000). This logic would suggest that students of the field should get exposure to the most current techniques as well as to a multitude of surrogate approaches.

Current research has shown that there is a plethora of software development methodologies (Cockburn 2002, Fitzgerald 2003). The agile methodologies are the most current techniques and open source development also shows much promise (Feller 2000). As an applied discipline then perhaps undergraduate IT programs should cover these topics in their core software development courses.

This paper has shown that while the textbook authors have done a good job of remaining current with regard to software development methodologies, course literature shows that the affiliated course content may not be current.

7. ACKNOWLEDGMENTS

Our thanks to Robin Keller from Ramapo College of New Jersey for assisting in collecting the data for this study.

8. REFERENCES

[1] Abrahamsson, P., Warsta, J., Siponen, M.T., & Ronkainen, J. (2003). New Directions on Agile Methods: A Comparative Analysis. ICSE 2003. USA.

[2] Avison, D., Fitzgerald, G. (2003). "Where now for development methodologies?" CACM 46(1): 78-82.

[3] Beck, K. (2000). Extreme Programming Explained, Pearson Education.

[4] Brown, D. (2002) An Introduction to Object-Oriented Analysis: Objects and UML in Plain English, 2nd Edition, Wiley

[5] Brooks Jr., F.(1987), "No Silver Bullet, Essence and Accidents of Software Engineering", Computer Magazine

[6] Choi, S., Deek, F. (2002). Extreme Programming Too Extreme? New Jersey Institute of Technology.

[7] Cockburn, A. (2002). Agile Software Development, Addison-Wesley.

[8] Dennis, A., Wixom, B. (2003). System Analysis and Design - Second Edition, Wiley, New York

[9] Dennis, A., Wixom,B., Tegarden (2004). Systems Analysis and Design with UML, Second Edition, Wiley, New York

[10] Fitzgerald, B. (1997). "The use of systems development methodologies in practice: A field study." The Information Systems Journal 7(3): 201-212.

[11] Fitzgerald, B. (2003). "Software Method Tailoring at Morotola." Communications of the ACM 46(4): 64-70.

[12] Feller, J., Fitzgerald, B. (2000). "A Framework Analysis of the Open Source Software Development Paradigm", The 21st International Conference in Information Systems (ICIS 2000), pp. 58- 69

[13] Fowler, M. (2002). "The New Methodology." 2002, from http://martinfowler.com/articles/newMethodology.html.

[14] Glasser, B. G. & Strauss, A. L. (1967). The discovery of grounded theory: Strategy for qualitative research. Hawthorne, NY: Aldine Publishing Company.

[15] Highsmith, J. (2000). Adaptive Software Development - A Collaborative Approach to Managing Complex Systems. New York, NY, Dorset House Publishing.

[16] Hoffer, J., George, J., Valacich,J. (2005). Modern Systems Analysis & Design, Fourth Edition, Prentice Hall.

[17] Kendall, K. E., Kendall, J.E. (2004). Systems Analysis and Design,sixth edition. Upper Saddle River, New Jersey: Prentice Hall, 2002.

[18] Larman, C. (2002). Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and the Unified Process (2nd Edition), Prentice Hall.

[19] Moody, D. (2000). "Building Links Between IS Research and Professional Practice: Improving the Relevance and Impact of Research". The 21st International Conference in Information Systems (ICIS 2000), pp. 351-360

[20] Phillips, P. A. (1998) "Disseminating and Applying the Best Evidence," *Medical Journal of Australia*, March 16, 1998.

[21] Racoon, L. B. S. (1997). "Fifty years of progress in software engineering." ACM SIGSOFT Software Engineering Notes 22(1).

[22] Roberts, E. (2000). "Computing Education and the Information Technology Workforce". ACM SIGCSE Bulletin 32(2):83-90.

[23] Royce, W. W. (1970). Managing the Development of Large Software Systems: Concepts and Techniques. IEEE, Westcon.

[24] Satzinger, J., Jackson, R., and Burd, S. (2004). Systems Analysis and Design in a Changing World (3rd ed.). Boston: Course Technology.

[25] Schach, S. (2003). Introduction to Object Oriented Analysis and Design.McGraw Hill.

[26] Shelly, Cashman, and Rosenblatt (2005). Systems Analysis and Design, Sixth Edition. Boston: Course Technology.

[27] The Standish Group. (1994) from http://www.standishgroup.com/sample_research/chaos_1994_1.php

[28] Truex, D., Baskerville, R., Travis. J. (2000). "Amethodical Systems Development: The Deferred Meaning of Systems Development Methods." Journal of Accounting, Management, and Information Technologies Tech 10: 53-79.

[29] Valacich,J., George, J., Hoffer, J. (2002). Essentials of Systems Analysis and Design Second Edition, Prentice Hall.

[30] Whiting, R. (1998). "Development in Disarray", Software Magazine, 20

[31] Whitten, J.L., Bentley, L.D. and Dittman, K.C. (2004), Systems Analysis and Design Methods, (6th edition.), McGraw-Hill, Boston, MA, USA.