AN ANALYSIS OF THE GAP BETWEEN THE KNOWLEDGE AND SKILLS LEARNED IN ACADEMIC SOFTWARE ENGINEERING COURSE PROJECTS AND THOSE REQUIRED IN REAL PROJECTS

Stephanie Ludi[1] and James Collofello[2]

Abstract *— This paper describes how the Software Engineering Body of Knowledge (SWEBOK) can be used as a guide to assess and improve software engineering courses. A case study is presented in which the guide is applied to a typical undergraduate software engineering course. The lessons learned are presented which the authors believe are generalizable to comparable courses taught at many academic institutions. A novel approach involving large-scale software project simulation is also presented a way to overcome some of the course deficiencies identified by the guide.*

Index Terms — simulation, software engineering education, software requirements, SWEBOK

INTRODUCTION

The creation of quality software by development organizations is a complex process requiring effective collaboration of many software engineers. The responsibility of educating new software engineers lies primarily in Computer Science departments and their software engineering courses. In particular, the typical undergraduate software engineering course provides an introduction to software engineering principles and techniques as well as the opportunity to work on a "realistic" team project. Unfortunately the constraints of the academic environment rarely provide an opportunity to replicate the size and complexity of a typical industry project.

In an attempt to better understand the knowledge and skills required of software engineers, the professional societies are helping to define the Software Engineering Body of Knowledge (SWEBOK). SWEBOK is intended as a guide to the subset of generally accepted software engineering knowledge. The guide does not dictate curricula, but it can assist in the development of curricula as each Knowledge Area is decomposed into topics and associated with ratings from Bloom's taxonomy.

In an effort to prepare students for industry projects, educators need to know how the knowledge and skills acquired in the small team projects assigned in their undergraduate Software Engineering courses compare to the breakdown of SWEBOK's Knowledge Areas using Bloom's taxonomy. To illustrate this, a typical undergraduate Software Engineering course's content and project was analyzed and mapped to SWEBOK using Bloom's taxonomy. From this mapping, gaps were noticeable in the nature of the course project that may leave students unprepared for industry projects. This paper describes the nature of these gaps as well as possible approaches for bridging them. In particular, the strategy of student participation in a simulated project environment is presented.

SOFTWARE ENGINEERING BODY OF KNOWLEDGE

The Software Engineering Body of Knowledge Project (SWEBOK) is a joint effort by the IEEE Computer Society and the Association for Computing Machinery (ACM) to develop a guide to the subset of generally accepted knowledge that defines the Software Engineering profession. The project's intent is not to define the body of knowledge or to dictate the curricula for university programs. However such the guide can assist in the development of curricula and accreditation criteria. The overall goals of the Guide to the Software Engineering Body of Knowledge are to:

- Characterize the contents of the Software Engineering Body of Knowledge;
- Provide topical access to the Software Engineering Body of Knowledge;
- Promote a consistent view of software engineering worldwide;
- Clarify the place of and set the boundary of software engineering with respect to other disciplines such as Computer Science, Project Management, Computer Engineering, and Mathematics;
- Provide a foundation for curriculum development and individual certification and licensing material. [4]

The project consists of three phases: Strawman, Stoneman, and Ironman. The Strawman phase has been completed and has resulted in a guide presenting the Knowledge Areas and Related Disciplines. This Strawman phase intended to bring together the discipline in order to move the project forward. As on this writing, the Stoneman phase is near completion. The objective of the Stoneman version of the guide is to organize the body of knowledge into Knowledge Areas, a list of topics relevant to the materials for each Knowledge Area (see Table 1) and a list of Related Disciplines [2]. The ten Knowledge Areas, and

[1] Stephanie Ludi, Arizona State University, Department of Computer Science and Engineering, Tempe, AZ 85287-5406 sludi@asu.edu
[2] James Collofello, Arizona State University, Department of Computer Science and Engineering, Tempe, AZ 85287-5406, james.collofello@asu.edu

0-7803-6669-7/01/$10.00 © 2001 IEEE **October 10 - 13, 2001 Reno, NV**
31st ASEE/IEEE Frontiers in Education Conference

the topics that comprise them, are considered to be the core knowledge. The knowledge that software engineers need to know from related disciplines is outside of the scope of the Guide and will be left to the other working groups to define. The Ironman phase will facilitate experimentation and trial usage of the guide, the promotion of the guide, and the development of "performance norms" for professionals [1]. The entire effort has been the result of the continuing collaboration of individuals from industry, academia, and standard setting bodies from all over the world.

TABLE I
KNOWLEDGE AREAS

SWEBOK Knowledge Areas
Software Configuration Management
Software Construction
Software Design
Software Engineering Infrastruction
Software Engineering Management
Software Engineering Process
Software Evaluation and Maintenance
Software Quality Analysis
Software Requirements Analysis
Software Testing

The keys to the Guide are the Knowledge Areas and the mapping of topics within these Knowledge Areas. Each Knowledge Area is organized according to Figure 1, consisting of a hierarchical breakdown of topics, reference topics, a matrix of the topics and the reference materials. The topics for each Knowledge Area are decomposed, described, and classified according to Vincenti's taxonomy, rated by Bloom's taxonomy, and referenced to related disciplines [2].

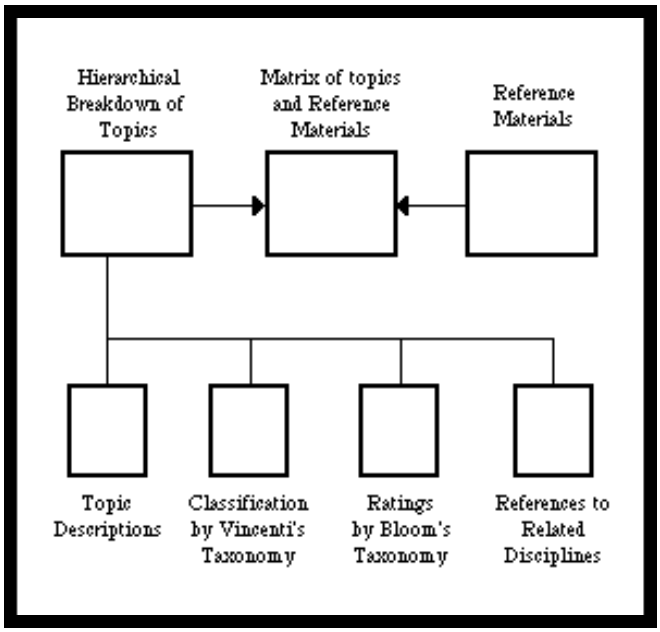

FIGURE. 1
THE ORGANIZATION OF A KNOWLEDGE AREA DESCRIPTION.

As of the current public version of the Stoneman Guide (version 0.7), the mapping of topics to the Knowledge Areas is complete and available online [3].

The purpose of the guide is not to dictate curricula. Instead, the guide inventories the topics and the depth of knowledge for these topics based on Bloom's taxonomy for a graduate with four years of experience [1]. The topics, organized by Knowledge Area, and the classification according to Bloom's taxonomy can be found at the end of each Knowledge Area section [3].

This information can provide a base whereby a curriculum can be designed for an undergraduate software engineering program and existing undergraduate software engineering courses assessed.

USING SWEBOK TO ASSESS AN EXISTING UNDERGRADUATE SOFTWARE ENGINEERING COURSE

To illustrate how SWEBOK might be used to assess and improve an existing course, the authors applied the guide to an existing Introduction to Software Engineering course (CSE 360) at Arizona State University. This course is a one-semester project course in which students work in teams of 5-6 members to develop a software application. The course project typically spans the entire semester starting with the teams defining the projects' requirements and ending with acceptance testing. The development of the course project follows a defined and documented methodology presented by the instructor. The teams are organized as self-directed work teams and are responsible for planning and tracking their activities. The course project experience is not carried out exactly as an industry project would be due to limitations of time and other factors, but an approximate experience is provided to students. The course content typically follows one of the leading texts such as Pressman's "Software Engineering: A Practitioner's Approach". The course at Arizona State University has evolved over 20 years and is similar to courses offered in many universities.

The first task the authors performed was capturing the informal desription of the course's topics and objectives in terms of Bloom's Taxonomy.

Subsequently a mapping of the CSE 360 course to Bloom's Taxonomy and the SWEBOK was created and verified by the course instructors. The matrix containing the SWEBOK topics and the CSE 360 curricula mapping using Bloom's Taxonomy is available from the authors. As the breakdown is extensive, a sample of the topics from the Software Requirements Analysis Knowledge Area is presented in Table 2.

TABLE 2
BLOOM'S TAXONOMY MATRIX FOR SWEBOK AND AN UNDERGRADUATE SOFTWARE ENGINEERING COURSE

Software Requirements Analysis	SWEBOK	CSE 360
I. Requirements Engineering Process		
A. Process models	Knowledge	Knowledge
B. Process actors	Knowledge	Knowledge
C. Process support	Knowledge	Knowledge
D. Process quality and improvement	Knowledge	Not Applicable
II. Requirements Elicitation		
A. Requirements Sources	Comprehension	Comprehension
B. Elicitation Techniques		
1. Interviews	Application	Application
2. Scenarios	Application	Comprehension
3. Facilitated Meetings	Application	Comprehension
4. Observation	Application	Comprehension
III. Requirements Analysis		
A. Requirements classification		
1. Functional & Nonfunctional	Comprehension	Comprehension
2. Derived from 1+ high-level req. or imposed by a stakeholder/other source	Comprehension	Not Applicable
3. Product or Process	Comprehension	Knowledge
4. Prioritizing req. (mandatory, highly desirable, desirable, optional)	Comprehension	Knowledge
5. Scope	Comprehension	Knowledge
6. Volatility / Stability	Comprehension	Not Applicable
B. Conceptual modeling	Comprehension	Application
C. Architectural design & requirements allocation	Analysis	Application
D. Requirements negotiation	Analysis	Not Applicable
IV. Requirements Specification		
A. The requirements definition document		
1. For customer	Application	Application
2. For other stakeholders	Application	Not Applicable
B. The software requirements specification (SRS)	Application	Application
C. Document Structure	Application	Application
D. Document Quality		
1. Selecting appropriate indicators	Analysis	Comprehension
2. Gathering and Analyzing Metrics from reviews.	Analysis	Comprehension
V. Requirements Validation		
A. The conduct of requirements reviews		
1. Group composition is appropriate (may include customer)	Analysis	Application
2. Use of guiding documents like checklists to guide review and to doc findings	Analysis	Comprehension as docs used for recording only
3. Review process is at specified checkpoints and redone as appropriate	Analysis	Application but done once
B. Prototyping	Application	Comprehension
C. Model validation	Analysis	Comprehension
D. Acceptance tests	Application	Application
VI. Requirements Management		
A. Change management		
1. Understanding the role of Change Management throughout lifecycle	Analysis	Analysis
2. Have procedure in place	Analysis	Comprehension
3. Analyze proposed changes	Analysis	Comprehension
B. Requirements activities	Comprehension	Comprehension
C. Requirements tracing	Comprehension	Comprehension

The mapping of Bloom's Taxonomy to the SWEBOK topics and to the material currently covered in CSE 360 was used to identify areas where improvement is needed. While the course addresses most of the topics to some extent, several gaps exist between the level of knowledge expected from SWEBOK (and large projects) and the current course. For example, the topics relating to Requirements Analysis in Table 2 shows that students generally only have minimal knowledge of the skills and knowledge needed to work with stakeholders from their perspectives. Under the sub-topic of Elicitation Techniques, students gather requirements via interviews. However the use of observation, scenario-based techniques, and facilitated meetings is non-existant due to the current structure and timeline of the class.

Besides gathering requirements, the ongoing tasks involved in monitoring quality and maintaining the requirements documents are also not addressed in the current course. Metrics gathering is discussed, but gathering data and assessing the quality of the requirements documentation is not accomplished. Also formal inspections are performed, but not to the degree of rigor that occurs in mature organizations. As such the inspection-related entries in the matrix have a special note associated with them.

Beyond the tasks carried out in the Requirements phase, the need to address the management of change in requirements is clear. Due to the nature of a college course, project requirements are provided to students during the initial phase, with only a minor amount of change occuring later in the project. Industry projects deal with requirements changes throughout the lifecycle. As a result, students only have basic knowledge regarding managing change, but the application of the knowledge to the project is absent.

CURRICULUM IMPROVEMENT USING SWEBOK

An analysis of the gaps in the matrix in Table 2 suggests that CSE 360 needs improvement. The authors feel, however, that these gaps are not unique to CSE 360 but typical of academic courses with time and size constraints on course projects. There are several possibilities to bridging the gap including teaching multiple semester courses with a project spanning the courses, more focus on case study analysis of large projects or the utilization of industry internships integrated with the classes. Another approach under investigation by the authors is the utilization of simulation technology to create a large project learning environment.

A research project to develop a large project simulator is under development to address some of the gaps between academic courses and SWEBOK guidelines. Although simulating an entire project's lifecycle would be an overwhelming task, a portion of the lifecycle, Requirements Analysis, is being approached initially. Typical courses such as CSE 360 do not address requirements issues regarding areas such as requirements management for a large project. This includes requirements negotiation, requirements allocation, and the roles that stakeholders play in development. As a result of not presenting these issues adequately the students may not appreciate the tasks and activities required to produce a set of requirements necessary to will lay the foundation for a large project.

Addressing these topics in a simulator will allow students to apply the concepts learned in class to a simulated, large software project in a more realistic context than if applied to a small student project alone. During the course of the simulation, the selected topics will also build upon one another as the simulation progresses. Students will see the consequences of their choices.

The immediate planned use of the simulator is for use as a supplemental aid to lecture material. While the importance of conducting requirements analysis and specification activities is discussed in class and students are required to traverse through a variety of activities during the course of their small group project, the simulator will provide opportunities that can be more realistic than the project - such as the industry experience that SWEBOK reflects. The simulator will reinforce the role that the activities play in a large project in industry in ways the lecture and a small group project cannot present.

This project will take the Guide to SWEBOK into consideration so that the course can best take advantage of the Guide and provide a more useful experience for students using the simulator as part of instruction.

FUTURE WORK

The next step is the design and implementation of the simulator. Upon completion, the simulator will be tested in the undergraduate Software Engineering course in order to assess the extent of the effectiveness of large project simulation in the instruction. Lecture and the small team project will not be eliminated. Instead the large project simulator will supplement the course in order to fill in some of the gaps between the knowledge outlined in SWEBOK and the course.

REFERENCES

[1] Abran, A., and Moore, J. (eds.) "*Guide to the Software Engineering Body of Knowledge", A Stone Man Version (version .6).* Available at http://www.swebok.org/ironman/ Guide to SWEBOK

[2] Bourque, P., Dupuis, R., et al. "The Guide to the Software Engineering Body of Knowledge". *IEEE Software*. Vol. 16, No 6, November/December 1999, pp. 35-44.

[3] IEEE Computer Society, Guide to the SWEBOK Web Site: Documents, Available at http://www.swebok.org/documents/

[4] IEEE Computer Society, Guide to the SWEBOK Web Site. Overview, Available at http://www.swebok.org/overview/