AC 2012-5188: ASSESSING EVOLVING CONCEPTUAL KNOWLEDGE IN SOFTWARE ENGINEERING STUDENTS

Prof. Kevin A. Gary, Arizona State University, Polytechnic

Kevin Gary, Ph.D., is an Associate Professor in the Department of Engineering within the College of Technology and Innovation at Arizona State University's Polytechnic campus. At ASU, Kevin led the development of the new bachelor's of software engineering program, and the revised design of the master's of computing studies. Gary designed and implemented the Software Enterprise, an NSF-funded hybrid pedagogical method for conducting project-based courses. More than 40 industry projects have been conducted by the Software Enterprise over the past eight years, and the Enterprise is the foundation of the project spine in the software engineering degree program. In addition to teaching software engineering, Gary also teaches in the areas of web/enterprise systems and database management systems. In 2009, Gary conducted one of the first Startup Weekend events in a university setting, promoting entrepreneurship and innovation among ASU students. Gary's research has focused primarily on software architectures, agile methods, and open source software for the operating room of the future. In cooperation with the Georgetown University Medical Center, Gary has received funding from the National Institutes of Health to develop the Image-Guided Surgical Toolkit (IGSTK). For his 2011-12 sabbatical appointment, Gary has been working as Chief Software Architect in the Bioengineering Initiative of the Sheik Zayed Center for Pediatric Surgical Innovation at the Children's National Medical Center in Washington, D.C., in the area of microrobotics for surgical applications. Gary has authored over forty peer-reviewed publications and received funding from the National Science Foundation, the National Institutes of Health, IBM, the Kaufmann Foundation, and the state of Arizona. He is an active member of the ACM, IEEE, and ASEE, and serves on a number of program committees for the community.

Miss Yegeneswari Nagappan, Unicon, Inc.

Yegeneswari Nagappan works as Software Developer at Unicon, Inc. She holds a master's degree in computing studies from Arizona State University.

Supreet Verma, Delasoft, Inc.

Supreet Verma was born and raised in India, mostly lived in Lucknow, capital of Uttar Pradesh (one of the states in India). His father did his bachelor's of science and master's of science in mathematics that influenced me to choose my career in the field of engineering. He has completed senior secondary schooling from City Montessori School in Lucknow and choose science, mathematics, and computers as my main subjects. He cracked IIT-JEE entrance and joined Indian Institute of Technology, Roorkee, India, to do his undergrad in electrical engineering (B.Tech.). In his second year of college, he got more interested in computer science (CS). He took some courses of CS, which his department offered and did his final year project in the same field. To choose this as his career path, he did his internship in a software firm SRM TechSol Pvt, Ltd., Lucknow, India, and joined Sasken Communication Technologies, Ltd., Bangalore, India, after his graduation. To acquire more domain knowledge, he came to Arizona State University as a master's student in computing studies. He was a research assistant under Kevin A. Gary. Under his guidance, he wrote algorithms and enhanced the existing one for concept map comparison which is used for assessment of software enterprise course. Currently, I'm working as a programmer analyst with Delasoft, Inc. in Newark, Del., developing distributed applications on different platforms (.Net, Java, and Android).

Russell J. Branaghan, Arizona State University

Russell Branaghan is an Assistant Professor of cognitive science and engineering at Arizona State University in Mesa, Ariz. His research interests include measuring the development and refinement of structural knowledge.

Assessing Evolving Conceptual Knowledge in Software Engineering Students

Abstract

Is it more important to "collect more knowledge" or to gain an understanding of how concepts relate so that a sustainable learning structure is formed? Active learning proponents would suggest it is more important to create a durable conceptual foundation by which new experiences are assimilated to grow the personal knowledge base in an orderly, well-formed way. But how can one assess that this is happening? In our project-centric courses we have adopted concept maps as a technique for evaluating the evolution of student understanding of conceptual knowledge in software engineering. This paper motivates the application of this existing technique in software engineering education, and presents a concept mapping assessment protocol based on a unique expert ranking process. An instance of the protocol implementation with results is presented and preliminary conclusions drawn.

1 Introduction

Over the past decade, the software engineering education community has focused significantly on defining the body of concepts that both undergraduate and graduate students should (in some sense) "know" and be able to apply upon graduation. Efforts such as the SWEBOK[1], the SEEK[2], and the GSwE2009[3] together with associated certification processes from IEEE (CSDP and CSDA) and ISO (ISO/IEC 24773:2008) go to great lengths to define outcomes primarily as coverage of the maturing body of knowledge (BOK) in software engineering. The evolution of the software engineering BOK and certifications over the past decade+ is a good thing as it speaks to the maturing of the profession. However, defining program outcomes in terms of content taxonomies flies in the face of emerging trends toward active and discovery-based teaching and learning techniques (be it called "hands-on", "project-centric", or "self-paced").

One criticism[4] of active or discovery-based learning approaches is that they are not the most cost-effective modality for transmitting information. To say it directly, an instructor can cover far more content in a traditional lecture format than in a project (or other applied) format. In light of recent pervasive budget pressures in higher education, the ability to cover more with less for more customers (students) is a practical reality. A second criticism of the active approach is that individual assessment is difficult as project work is typically done in teams. Further, it is necessary for an instructor to continuously monitor class progress as a whole toward learning outcomes, which is complex in an environment where team progress evolves distinctly and nonlinearly. Teams do not learn at the same rate, go about applied work in the same way, and come upon the hoped-for "aha" moment at the same time. The authors believe hybrid pedagogical approaches such as scaffolding or the Software Enterprise[5,6] are the best ways to address this issue.

Concept maps[7] may be a tool that can help formative and summative assessment related to active learning. Concept maps are not classic assessment mechanisms like assignments, labs, and exams. Concept maps offer an alternative assessment tool that can be used to show that a student has acquired knowledge, and can organize that knowledge into an evolving structure. The ability

to evolve is pedagogy independent and content "volume" independent. That is, it is not as important how much "stuff" the student learns or by which method s/he learns it, it is more important that the foundation of that knowledge is organized so it may be extended as the student matures. Concept maps are a formative tool educators may use to determine if the student is on a proper learning trajectory.

Concept maps have been around for some time and have been employed in a number of disciplines, so what is unique about their application in computing sciences? Computing sciences are relatively immature compared to traditional STEM disciplines, and therefore there is still a lot of debate about what content is appropriate for 7-12th grade students and lower-division undergraduates. Computing sciences are also fast-moving disciplines. The ability to keep curriculum content current is a slippery slope for educators. Worse, for students, the technologies they may learn in their freshman class may be obsolete by their senior (junior? sophomore?) year. Therefore it is imperative to consider how students come to understand computing concepts and rapidly assimilate new technologies from a stable knowledge framework.

2 Concept Maps

Concept maps are a tool for organizing knowledge in a graphical format[7]. They consist of concepts shown as nodes in a graph, and (labeled) relations between concepts shown as arcs. Generally, a label is a word or group of words, and sometimes it can also have mathematical symbols like + or %. Propositions are used as a statement about some concept, they help in providing a meaningful relationship between concepts by connecting them using linking words or phrases. Figure 1 shows example maps relating the concepts of *province* and *state*. The map on the left defines a similarity relation between the two, while the map on the right introduces a new concept, or organizing node, to define an intermediary through which the origin concepts are related.

Figure 1. Sample concept maps

Concept maps are quite similar to mind maps[8], though the context of application is different. Mind mapping activities are often done during discovery, where one is trying to understand a new domain. As examples, students may create a mind map to diagram comprehension of reading a textbook chapter, or business analysts might create a mind map as part of an idea generation or business modeling activity during requirements elicitation. Concept mapping, at least in our context, is an activity that attempts to reflect the structural aspects of a student's evolving conceptual understanding. Note in this description that we do not say that concept maps themselves are an assessment of any kind; indeed they can be interpreted in many ways. For instance, student teams might use concept maps to bring to light differences in conceptual understanding when working on a collaborative assignment or project. Concept map assessment

requires a process of interpreting the "goodness" or "badness" of a concept map with respect to some expectation or goal, and may be used in a formative or summative assessment process. Given the visual and subjective nature of concept maps, it is critical for an educator to explicitly define the assessment process.

2.1 Concept Maps and the Software Enterprise

We introduced concept map evaluation as part of the Software Enterprise at Arizona State University. The Software Enterprise is described in several other papers[5,6] so here we highlight only the relevant aspects. The Enterprise is a pedagogical model that requires students to perform five time-bound learning phases: *Preparation*, *Dissemination*, *Practice*, *Project*, and *Reflection*. Preparation is a pre-class meeting activity, such as reading or brainstorming or researching, usually concluding with an online quiz. Dissemination is a traditional class meeting time where the instructor emphasizes content through lecture or discussion. Practice is a collaborative lab activity, and Project is the application of the concept in the context of a scalable project. Reflection is an activity that ideally occurs throughout the phases for a given module, though in practice we tend to require students to post reflections to an e-portfolio after the Practice activity and again after the Project activity. The key features of the Enterprise pedagogy are the modularization of the course content, and the proximity in time in which the learning phases of a module are conducted; three weeks elapse from the time a student is exposed to a new concept to the time when they are asked to implement and reflect on it in a scalable project. The intent is to immediately contextualize the learning and gain exposure to the external forces that may distort its application in that context.

The Software Enterprise is not a pure active or project-based pedagogy, but nonetheless these are the most significant aspects. A ramification of employing this pedagogy is that the instructor simply cannot cover the same volume of content as s/he might in a traditional class (lecture + homework). The expected benefit is that somehow the students will learn the concepts *better*, in the sense they are learned in context. This contextual learning, our argument goes, should lead to 1) rapid integration into the workforce upon graduation[5], and 2) better structural foundation of software engineering knowledge that will evolve in an orderly way even after graduation. That is, the students understand less "stuff" but understand it in a way that is deeper and better organized so as to be better prepared to assimilate new and changing conceptual information, a requirement in the computing sciences. Concept map evaluation was utilized to try and assess this second benefit of contextualized learning.

2.2 Creating Concept Maps

We used the Cmap tool from the Florida Institute for Human & Machine Cognition (IHMC[7]) Software Enterprise students were asked to create concept maps from a template. The *scaffold* concept maps had nodes corresponding to the concepts to be covered in the coming semester. Students were asked to organize these concepts by defining relations between concepts. Student were allowed to introduce new nodes (aggregators or organizing nodes) in a different color (green). The map authoring process allowed for additional annotations such as capitalization to

represent link strength (capitalized labels represent stronger relationships) and directionality of relations (arrows). Figure 2 shows an example map from a Software Enterprise student.

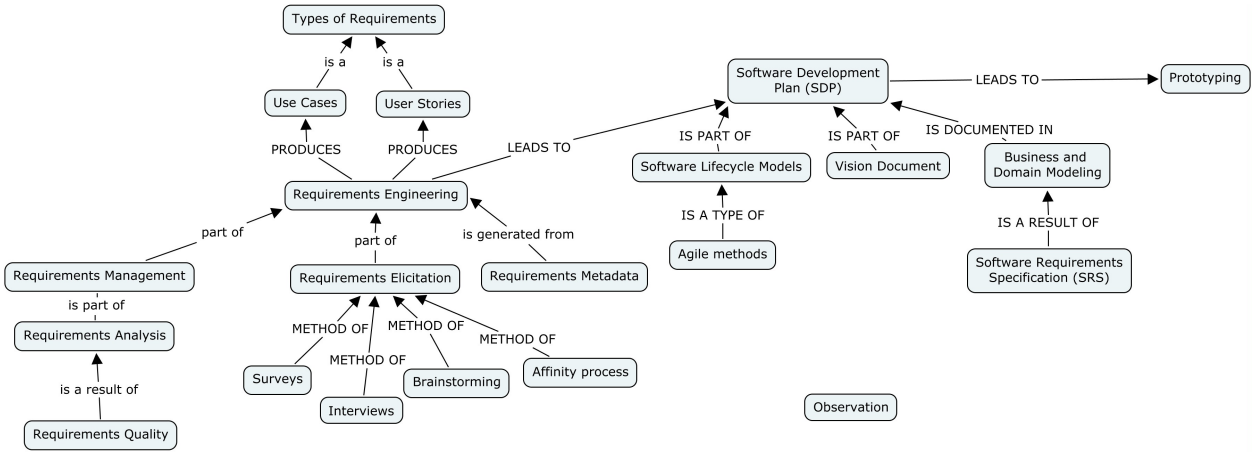

Figure 2. Example concept map from a software enterprise student.

The methodology was a pre-post process defined as follows:

1. At the beginning of the semester, students were given a lab where they were asked to create a concept map from a scaffold map.
2. Students were asked to arrange the concepts in a manner that best reflected their understanding of the concepts.
3. Students repeated the lab at the end of the semester, 15 weeks later, using the same scaffold concept map and under the same directions. They were not permitted to reference their concept maps from the beginning of the semester.

We explore ways to assess concept maps in the next section.

3 Assessing Concept Maps

The literature contains significant discussion of using concept maps for assessment (see *Journal of Research in Science Teaching*, December 1990), examples of software tools and algorithms for assessment[8,9,10], and applications in STEM disciplines[11,12,13].

There are multiple things to consider when assessing concept maps – what to look at, what the output of the process is (quantitative or qualitative, formative or summative), and how to assign "goodness" to it. McClure, Sonak, and Suen (1999) conducted a study of 6 scoring techniques[14], and categorized 3 ways of describing what to look at:

1. *holistic* – a rater examines the concept map as a whole and awards a score from 0 to 10
2. *relational* – a rater examines individual propositions and scores them each between 0 and 3 based on appropriateness
3. *structural* – a rater examines hierarchical structures and cross links to evaluate concept organization

These approaches differ in what they ask an evaluator to focus on, and also suggest a quantitative score be assigned based on rubrics. The authors also presented modified versions of these that compared them against a master (expert) map.

Using this framework as a reference model, we believe that holistic and relational approaches made sense but defining rubrics with quantitative scores seemed somewhat arbitrary. We did not feel we had sufficient experience with concept maps to define rubrics and attach scoring criteria. Further, it was not the intent of the assessment exercise to assign grades to students, instead our goal was to determine if the Enterprise pedagogical model was achieving its goal of accelerating student preparedness for the workforce. Finally, we first started down an assessment process path of using automated graph evaluation algorithms, but ended up relying on expert evaluation by humans instead of machine evaluation (automated evaluation remains ongoing work). We defined an assessment process addressing the question "is the student organizing and evolving knowledge in a way that shows progression towards a maturing software professional?"

Consider an example of a science class. Upon entering the class, students would certainly understand the concept of temperature. Further, they might understand volume from using things like measuring cups. They might even understand the notion of gases under pressure from blowing up balloons. But, they would not likely understand the relationships among these concepts. As a result these nodes would probably not be linked in their concept maps. They would however, be linked in the instructor's concept map. As the course progresses, we would expect that the relationships among these items would be learned and linked in the student's map. As a result, her concept map would begin to resemble that of her teacher's. Further, as all the students approximate the canonical model of their instructor, they will become more similar to each other. Indeed research has shown that students with knowledge structures most similar to their instructor tend to earn better grades in class[15].

From the Software Enterprise perspective, students entering the Enterprise (start of their junior year) would have an understanding of basic computing and software development concepts – discrete math, data structures, programming in a small number of languages, etc. Rudimentary software engineering concepts, such as writing test harnesses, structuring and documenting code, procedural and/or object-oriented design, and problem understanding have been introduced but are often not the primary learning objectives in lower division courses. The upper division focus in the Software Enterprise will, we hope, accelerate their contextualized understanding of these concepts and the more advanced concepts upon which they build – Software Maintenance, Validation & Verification, Software Architecture, and Requirements Analysis. The students' concept maps should exhibit an arrangement of these concepts that is on an evolutionary trajectory toward that of a software professional.

Concept map evaluation may be made stronger with the presence of discipline-specific metadata to inform algorithms. The algorithms may be made more intelligent by not only looking structurally at the map but semantically. Understanding of the concepts and the relationships – how closely they are related, relevant or correct – can lead to deeper evaluation of constructs. For example, one may be able to localize areas of concern within a large concept map, identifying areas where the structure is sound (implying a well-learned concept) but in fact the relationships are incorrect. These areas would be places where the educator may need to help the student

"unlearn" an incorrect formation and relearn the proper one. This is just one example. To illuminate the idea further, we give a brief example from one of our software engineering courses.

Figure 3 shows an example concept map generated by a senior student at the beginning of a software project management class. The student was given a set of concepts and asked to arrange them in a graph and label relationships. Relationship labels were not pre-specified, and capital letters are used to indicate a stronger relationship than lowercase letters. New nodes could be introduced if the student feels it enhances the representation. This network, despite having disconnected components is actually fairly strong structurally. It introduces new central nodes (*Development Process* and *Process Components*) that were not part of the scaffolding. However, an experienced software engineer may have some issues with some of the structures based on the relationships themselves. For example, why are *SCRUM and* Agile methods not connected? Risk Management is more than just a Process Component based on contingency planning. Software Maintenance does not "ensure correct functionality" for Testing, and so on. One could look at the sub-graphs here and like the structure, but would need to evaluate semantically to see if those structures made sense.

4 Assessment Protocol and Results

Our assessment protocol was as follows:

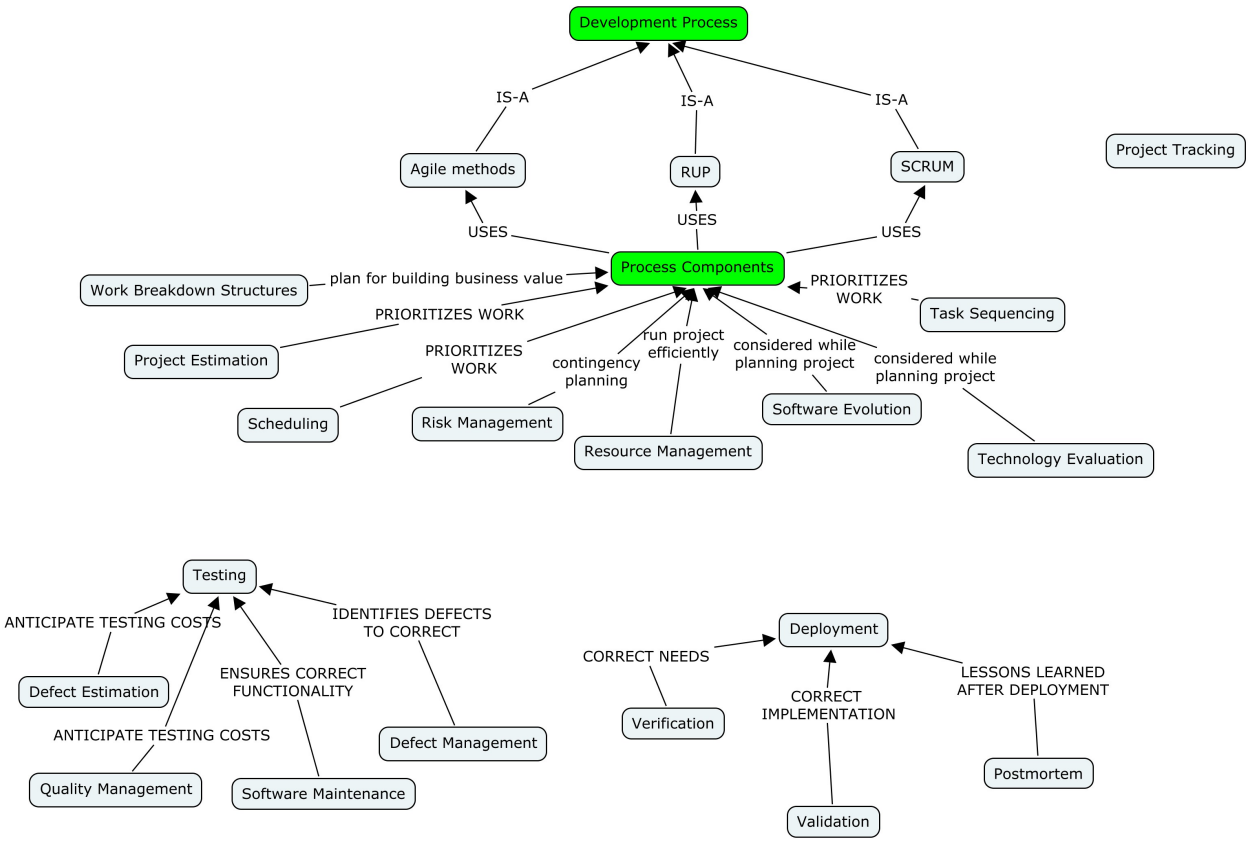

Figure 3. Software Process and Project Management concept map

1. Expert evaluators were recruited from the software engineering faculty.
2. The evaluators were given a short pre-briefing that included reviewing the student exercise (see student create map protocol above), and some description of what a concept map is and what it should capture. This did not include any rubrics or heuristics for evaluation however.
3. The evaluators were asked to visit a website that had two rating tasks defined for each evaluator.
    a. The tasks were identical, but with different data sets.
        i. The first data set contained concept maps from junior-level students, the second from seniors.
        ii. Pre and post maps were included in the same data set, so in effect each student had two maps present in a given data set.
        iii. Concept maps from former students who were now working in industry were added to each data set (4 maps for the juniors, 3 maps for the seniors). These alumni had 2-4 years of professional experience.
        iv. Identification information was anonymized.
    b. The evaluator was asked to ordinal rank each data set using an insertion sort-like process. The evaluator was shown maps in a random order, one per screen at a time, and provided a tool to place it in a ranked list.

This protocol was designed to measure the knowledge organization trajectory. As a single data set included both the pre and post map, with the hypothesis that a maturing trajectory of knowledge organization would be shown by each student's post map being ranked higher than her/his pre map. Of course the protocol design is not perfect. For one it does not even attempt to quantify the difference between a student's pre and post map. It also does not set a target threshold for "goodness" of a student's maps. Again, this goes back to the discussion on quantitatively scoring concept maps in this space; we do not feel we have the experience or a significant enough data set to try and normalize a scoring function for the maps.

However we still wanted to measure whether our students were converging towards the knowledge maturity shown by recent college graduates, as accelerating the preparation of students for the workforce is as explicit goal of the Software Enterprise. So we modified the protocol by asking alumni to create maps for each data set, and seeded these maps in the data sets under anonymous ids. It is our intent to evaluate student maps against these maps for evolving similarity, but our concept map similarity engine is not yet complete.

The project recently conducted its first round of data collection from 18 students in two data sets. The senior student data set had a total of 11 students and 26 concept maps (2 per student plus 4 for alumni) while the juniors had a total of 7 students and 17 concept maps (3 alumni maps). While these data sets are small, we see it as a benefit for our first evaluation as more than this number of maps may bias the evaluation process from rater exhaustion. The results were compiled in a spreadsheet showing the raw ranking and also the relative position for each student by each rater. The results showed that:

1. The raters were generally consistent on most students, where consistency means they agreed on the trajectory of the student but not necessarily the magnitude. In the senior dataset the

expert raters were consistent for 8 of the 11 students. Of the 3 inconsistent students, 2 were significant inconsistencies. In the junior dataset the raters were consistent for 5 of the 7 students with 1 significant inconsistency. This is a positive result as it shows that expert raters are generally consistent with only a 20 minute pre-briefing of the exercise.

2. The alumni maps do not have trajectory information, but we note that the rankings for them were generally in the same neighborhood for the junior concept maps but not so for 2 of the 4 senior maps. In fact in the senior maps the alumni rankings were clustered around the median of the rankings for the most part, while in the junior rankings the alumni did much better (as expected). A possible explanation for this is that our junior concept map is based on content more closely related to best practices in software development (for example, unit testing, configuration management, etc.) while the senior concept map centers around the broader software process, namely requirements and project/process management. We know the alumni who participated and they are all working industry jobs in which they do software development (coding) on a daily basis but are not primarily responsible for gathering requirements or managing projects. It would be useful to include more seasoned engineers in a future study.
3. Students generally stay in a consistent position relative to each other, though there are a few exceptions. That is, if a student produced one of the better pre maps, then s/he is likely to produce one of the better post maps. This is not unsurprising.
4. While 1-3 are generally positive results, we had the disappointing result that each rater evaluated 8 students as improving from pre to post assessment, and 10 students not improving. 3 of the 10 students not showing improvement were essentially ranked in the same place (neighboring slots in the ranking) and 1 of the improving students, for an adjusted tally of 7 students improving and 7 students not improving. This is a disappointing result as it does not validate the hypothesis that trajectories should show improvement.

Although there are some positive takeaways from our first evaluation pilot, we had the disappointment that the primary hypothesis we were testing, trajectory of organizational knowledge should improve, was not validated by the pilot study. Further, it is difficult to discern from the assessment process exactly why the trajectory was not improving in a particular student. However, the experience did reinforce our belief that concept maps are a potentially valuable method for assessing whether students are building conceptual knowledge of the field in an organized, maturing way. We recognize that the protocol described in this paper should be validated against other forms of assessment, and that there are aspects of the exercise design (level of scaffolding, amount of rater training or pre-briefing, etc.) that require further investigation.

5 Summary and Future Work

In this paper we motivate the need for evaluating students' organization of disciplinary concepts and suggest it is as important if not more important than the volume of content they are exposed to in their software engineering classes. Indeed, in a discipline where the half-life of content is perhaps the shortest of any STEM area due to its newness and rapid technological change, one can argue that a strong organizational foundation is key to assimilating and evolving with the discipline. To this end we have incorporated concept maps as an assessment tool in our upper-division project-based Software Enterprise curriculum. Concept maps reify students' organizing

principles of the material learned in context. At present we employ this method to evaluate the student only as a resulting artifact of the pedagogy; future work may include creating formative and perhaps self- assessment processes to assist students in organizing knowledge better. Concept maps may also one day form a basis for summative assessment at the course or program level, though at present the authors do not feel they have the significant experience necessary to conduct such a process.

This paper also contributes a unique ranking-based assessment process that focuses less on student understanding at a particular point in time but instead on the student's trajectory of learning. Again, this process should be adaptable to formative-style assessment. The current assessment requires expert evaluation however, which itself introduces bias. Our current research activities include developing custom concept map similarity algorithms that can be automated against larger data sets to help diffuse any bias while also making it easier to evaluate concept maps on a regular basis. We plan to use these similarity measures to evaluate the convergence of student maps toward those of alumni as per the goal of the Software Enterprise. We have also collected additional datasets that await evaluation to see if our initial results are repeatable or merely an anomaly.

Sheppard et al.[16] suggests engineering education transform to a model more like medical education, where theory and practice are blended through practice-oriented techniques. The Software Enterprise courses we are applying this method in today are compelling because project-based learning sacrifices "coverage" for deeper applied understanding. A concern when switching from a transmission-based approach (lecture) to a "pedagogy of engagement" such as this is whether you can "cover enough stuff"[17]. A student who has learned through project-based pedagogy, instead of by traditional lecture, may not be exposed to as much content but hopefully has a deeper framework of understanding. While concept maps may not assess how well a student can apply what s/he has learned, they can assess whether the conceptual knowledge gained is a foundation by which additional concepts may readily be integrated.

But how does one measure *how well*? Certainly traditional assessment mechanisms attempt to do this, but it is difficult to map out a student's learning trajectory using these methods. One might conduct pre and post assessments, but these are awkward in upper division curricula as one would not expect a student to perform well on a pre-assessment with concepts in which they are not familiar. Portfolios hold some promise, and these are also being introduced in the Software Enterprise. Concept maps are ideal for addressing one of the key questions – even if students are not getting as much "stuff," are they getting it in such a way that it is organized into a solid foundation upon which new knowledge may be added?

Bibliography

1. IEEE Computer Society, *Guide to the Software Engineering Body of Knowledge (SWEBOK)*, P. Bourque and R. Dupuis (Eds.). IEEE Computer Society Press, 2004.
2. ACM/IEEE, *Software Engineering 2004 Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering,* 2004.
3. GSwE2009. *Curriculum Guidelines for Graduate Degree Programs in Software Engineering, Version 1.0*, 2009.

4. Kirschner, P.A., Sweller, J., and Clark, R.E., “Why Minimal Guidance During Instruction Does Not Work: An Analysis of the Failure of Constructivist, Discovery, Problem-Based, Experiential, and Inquiry-Based Teaching” Educational Psychologist 41(2):75-86, 2006.
5. Gary, K. “The Software Enterprise: Practicing Best Practices in Software Engineering Education”, The International Journal of Engineering Education Special Issue on Trends in Software Engineering Education, Volume 24, Number 4, July 2008, pp. 705-716.
6. Gary, K., “The Software Enterprise: Preparing Industry-ready Software Engineers” *Software Engineering: Effective Teaching and Learning Approaches*, Ellis, H., Demurjian, S., and Naveda, J.F., (eds.), Idea Group Publishing. October 2008.
7. Novak, J.D. and Cañas, A.J., "The Theory Underlying Concept Maps and How to Construct and Use Them" Technical Report IHMC CmapTools 2006-01, Florida Institute for Human and Machine Cognition, 2006.
8. Eppler, M.J., “A Comparison between Concept Maps, Mind Maps, Conceptual Diagrams, and Visual Metaphors as Complementary Tools for Knowledge Construction and Sharing “ Information Visualization 5(3):202-210, 2006.
9. Cañas, A.J. and Novak, J.D. (*eds*.), “Assessing concept maps: First impressions count”. Proceedings of the Second International Conference on Concept Mapping, San Jose, Costa Rica, 2006.
10. Schvaneveldt, R. W. (ed.), *Pathfinder Associative Networks: Studies in Knowledge Organization.* Norwood, NJ: Ablex, 1990.
11. Taricani, E.M. and Clariana, R.B., A Technique for Automatically Scoring Open-ended Concept Maps, *ETR&D,* 54(1):65-82, 2006.
12. Turns, J. & Kirlik. A., “Structural Assessment to Support Engineering Education,” National Conference of the American Society for Engineering Education (ASEE ‘98), 1998.
13. Turns, J. Atman, C.J., and Adams, R.. Concept Maps for Engineering Education: A Cognitively Motivated Tool Supporting Varied Assessment Functions, *IEEE Transactions on Education*, 43(2), 2000.
14. Williams, C., Using Concept Maps to Assess Conceptual Knowledge of Function, *Journal for Research in Mathematics Education*. 29(4):414-421, 1998.
15. Markham, K.M. and Mintzes,J.J., The Concept Map as a Research and Evaluation Tool: Further Evidence of Validity, *Journal of Research in Science Teaching*. 31(1):91-101, 1994.
16. McClure, J. R., Sonak, B., and Suen, H.K., "Concept Map Assessment of Classroom Learning: Reliability, Validity, and Logistical Probability", *Journal of Research in Science Teaching*, vol. 36, no. 4, pp. 475-492, 1999.
17. Goldsmith, T. E., Johnson, P. J., & Acton, W. H. (1991). Assessing structural knowledge. *Journal of Educational Psychology*, 83(1), 88-96.
18. Sheppard, S.D., Macatangay, K., Colby, A., and Sullivan, W, *Educating Engineers: Designing for the Future of the Field.* Hoboken, NJ: Jossey-Bass, 2008.
19. Smith, K.A., Sheppard, S.D., Johnson, D.W., and Johnson, R.T., Pedagogies of engagement: Classroom-based practices. *Journal of Engineering Education*, 94(1):87– 101, 2005.