Changes in Transferable Knowledge Resulting from Study in a Graduate Software Engineering Curriculum

Ray Bareiss, Todd Sedano, and Edward Katz

February 9, 2012

CMU-SV-12-002

Carnegie Mellon University
Silicon Valley

Silicon Valley Campus
Carnegie Mellon University
Mountain View, CA 94035

Changes in Transferable Knowledge Resulting from Study in a Graduate Software Engineering Curriculum

Ray Bareiss, Todd Sedano, and Edward Katz
Carnegie Mellon University
Silicon Valley Campus
{ray.bareiss,todd.sedano,ed.katz}@sv.cmu.edu

Abstract

This paper presents the initial results of a study of the evolution of students' knowledge of software engineering from the beginning to the end of a master's degree curriculum in software engineering. Students were presented with a problem involving the initiation of a complex new project at the beginning of the program and again at the end of the program's formal coursework. After considering the problem each time, students were asked what questions they had as the senior software engineer, which software engineering processes need to be put into place, and to enumerate any other issues they saw. Statistical analysis indicates that their enumeration of software processes in the post condition is very significantly richer than in the pre condition. They also gave significantly more responses about requirements, design, and engineering management in the post condition. Qualitative analysis suggests that the students' questions in these areas were also more sophisticated in the post condition, suggesting, in accordance with the theory of "transfer as preparation for future learning," that they are moving along a trajectory towards expertise.

1. Introduction

We presented a paper at CSEET 2011 that explored transfer from a formal software engineering curriculum to a relatively unstructured, real-world practicum project [1]. Perhaps naturally, our research has given rise to a more fundamental question: What do students learn in this kind of curriculum that is potentially available for transfer to the solution of new problems?

Our previous work adopted a classic definition of transfer: Transfer of learning occurs when learning in one context enhances (positive transfer) or undermines (negative transfer) a related performance in another context [2].

To summarize, we saw that while knowledge and skills do transfer, our students exhibited some notable failures to transfer, particularly in the areas of definition of acceptance criteria, testing, quality assurance more broadly, and metrics. We implemented a relatively simple approach to scaffolding transfer in which students answered six general questions before undertaking the project and were then coached to improve answers that the supervising faculty member judged to be inadequate; this simple intervention improved transfer substantially in all areas except acceptance criteria and testing (in which we did see some improvement).

There is, however another view of transfer that merits consideration: *transfer as preparation for future learning* [3]. Becoming an expert takes literally years of "deliberate practice" [4]; it is

hopelessly optimistic to expect even a high-quality educational program to produce practitioners who can immediately bring exactly the right knowledge to bear across a range of situations.

The theory of transfer as preparation for future learning regards transfer from previous learning experiences as providing a foundation for the learning that must take place in a new situation rather than as simple activation of previously acquired knowledge and skills in the new situation (which only an expert with wide-ranging experience might be able to do routinely). Thus, according to this theory, an educational experience moves someone along the path to expertise, preparing him or her to more effectively learn the knowledge that is required to perform appropriately in a new situation rather simply pouring expertise into his or her mind from which it can simply be recalled and applied. A key hypothesis of this theory is that as students become more expert, they will exhibit more complex mental models of a discipline and will ask better questions when confronted with a new, complex problem situation. This paper reports on a study to apply this idea to the assessment of changes in our students' expertise.

2. Methodology

2.1 Subjects

The study was conducted on our 16 full-time graduate software engineering students who completed their master's degree studies in our Software Engineering Technical Track during the 2010-2011 academic year. Their work experience prior to entering the program ranged from 0 to 5.8 years with a mean of 1.9 years (and a standard deviation of 2.0).

2.2. The survey

To follow-up on the work reported in our CSEET 2011 paper, we created a new survey designed to probe students' mental models of software engineering through analysis of their question asking and initial, high-level project planning when confronted with a novel problem. During the first week of their first class, Foundations of Software Engineering [5], students were presented a fictional scenario announcing a substantial new development project: The development for a new software system to coordinate and manage emergency response at the NASA Ames Research Center. Students received a one-page description of the project vision, and a summary of high-level customer goals for the project:

- To integrate various system functions so that information can be entered once and shared, as appropriate, everywhere
- To enable mission-appropriate access to information by front-line responders and for that information to be continuously updated and "pushed" to the appropriate responders
- To enable information input by front-line responders, probably using mobile devices rather than traditional laptop computers
- To provide an abstracted "commander's overview" of an entire emergency situation with the ability to drill down selectively into the details displaying event conditions and resources.
- To enable continuous, seamless, secure data sharing with the emergency operations centers of other governmental entities.

The students were then asked three questions:

1. What questions do you have as the senior software engineer? (Please list all you can think of.)
2. What software engineering processes do you need to put in place?
3. What other key decisions must you make?

The survey was administered via SurveyMonkey (http://www.surveymonkey.com/). Each question called for an open-ended text response. The survey was administered again at the beginning of the third, and final, semester of their studies as they began their practicum projects [6]. (We used the identical survey to ensure direct comparability of results.)

In addition to Foundations of Software Engineering, students took six courses prior to the practicum:

- Avoiding Software Project Failures
- Project and Process Management
- Architecture and Design
- Metrics for Software Engineers
- Requirements Engineering
- The Craft of Software Development

and most worked as research assistants, developing software for faculty research projects. (They took two additional elective courses in parallel with the practicum.)

Note that there was a flaw in the survey administration process. Students were asked to complete the survey the second time without explanation. Feeling that they had already done the same work before, many students did not take the task seriously, and we were faced with a situation in which the typical response had significantly fewer and sloppier responses. To correct this, the authors met with the students and explained in general terms that the survey is intended as a very high-level assessment of what students have learned during their studies and that the goal is to improve the learning experience for future students. All students were then asked to redo the second survey. It is possible that this intervention positively affected the quantity and quality of answers reported in this paper.

3. Analysis

Thirteen of sixteen students completed the survey at the beginning of the practicum project. Our analysis was restricted to those 13 students.

3.1. Numbers of responses

Our initial analysis examined changes in the mean numbers of responses to each question.

	Pre Responses Mean (SD)	Post Responses Mean (SD)	$p=$
Question 1	6.5 (3.3)	6.8 (4.1)	0.76
Question 2	3.9 (3.4)	6.7 (2.8)	0.01
Question 3	3.1 (2.8)	4.1 (2.0)	0.17

Based on a two-tailed, paired T test, only the pre to post survey difference for question 2 ("software engineering processes to put in place") is significant (at the 99% confidence level). Four students gave a single response in the pre survey; in all cases, these responses identified a high-level development approach, e.g., Extreme Programming or the Rational Unified Process, which implies a number of component engineering processes. (Our subsequent analyses did not expand their responses to include these component processes.) No student gave a single response in the post survey.

Our next analysis examined correlations between student work experience prior to the program and the number of responses to each question. These correlations are low in all cases.

Correlation to work experience	*r* for Pre Responses	*r* for Post Responses
Question 1	0.17	-0.19
Question 2	0.32	-0.14
Question 3	0.37	-0.23

The negative correlations for the post survey are largely due to four students with significant work experience who provided fewer responses in the post survey.

We also conducted correlation analyses of the numbers of pre and post responses to each question and found this to be a more powerful predictor than work experience.

Correlation of pre to post responses	*r* for Pre/Post
Question 1	0.78
Question 2	0.53
Question 3	0.54

While much higher than correlations to work experience, only the pre/post correlation for question 1 accounts for the over half of the variance (r^2=0.61) in responses. These correlations are presumably due to underlying factors that are outside of the scope of the current study.

3.2. Types of responses

After looking at pre/post differences in the *numbers* of responses, we analyzed differences in the *types* of responses. Working together, the authors categorized each student response according to a taxonomy based on SWEBOK 2004 [7]: Requirements, Design, Construction, Testing, Maintenance, Configuration Management, Engineering Management, Engineering Process, Tools and Methods, Quality, Related Disciplines, and Other.

Eleven of these categories are employed exactly as defined in SWEBOK 2004. The other two, which we added, merit some discussion. Responses are assigned to "Related Disciplines" when they mention knowledge and skills that, while they may be important to project success, are not typically considered to be core competencies of software engineering; for example, one student listed human-computer interaction as a process that must be put in place. (SWEBOK 2004 identifies seven such disciplines.) Responses are assigned to "Other" when they are not a clear fit to a SWEBOK category; for example, one student asked "Do I have to sign a nondisclosure agreement with NASA?" The authors categorized all responses by consensus.

3.3. Statistical analysis of responses

Because of the relatively small numbers of responses in each category, we summed the numbers of responses across the three questions. We then conducted paired, two-tailed T tests on the summed numbers of responses. The difference in the total number of response categories pre to post, summing responses across questions 1-3 was not significant:

	Pre Responses Mean (SD)	Post Responses Mean (SD)	$p=$
Response categories Q1+Q2+Q3	4.77 (1.59)	5.08 (1.66)	0.63

Looking at individual categories, and again summing responses across categories, Design and Engineering Management are significant; Requirements is significant if one outlying student, who went from 19 requirements responses down to 8, is excluded:

	Pre Responses Mean (SD)	Post Responses Mean (SD)	$p=$
Design Q1+Q2+Q3	0.62 (0.77)	1.54 (1.05)	0.01
Engineering Management Q1+Q2+Q3	4.46 (2.85)	6.38 (2.87)	0.03
Requirements Q1+Q2+Q3 (excluding one subject)	3.67 (3.17)	5.83 (2.89)	0.05

Looking at individual questions, no response categories exhibit significant pre/post differences for question 1. Design and Engineering Management are significant for both questions 2 and 3:

	Pre Responses Mean (SD)	Post Responses Mean (SD)	$p=$
Design Q2	0.08 (0.28)	0.54 (0.66)	0.05
Design Q3	0.31 (0.48)	0.77 (0.73)	0.05
Engineering Management Q2	1.08 (1.19)	2.08 (1.44)	0.01
Engineering Management Q3	1.31 (1.38)	2.23 (1.09)	0.004

3.4. Qualitative analysis of responses

Our statistical analysis suggests that the Requirements, Design, and Engineering Management categories merit further study. We performed a qualitative analysis of the responses of each student in each of these categories using the aggregated question 1-3 data. (There is simply too little data to look at each student's responses question by question.) Adapting an idea introduced by Bransford and Schwartz [3], we employ a "trajectory towards expertise" to characterize the likely state of each student's knowledge on the "path" from novice to journeyman proficiency in each of these areas.

Working together, the authors manually clustered students' responses, based on data-driven, subjective, holistic assessments of their responses in each category (cf. idea consolidation in brainstorming [8]). This process was repeated separately for pre and post responses. We then summarized the emergent common attributes of each cluster and named the cluster, again based on the authors' judgment, to provide a still higher-level summary. Finally we plotted each student's trajectory from pre to post survey between (or sometimes staying within) clusters to provide a visual summary of our assessments of student learning.

3.4.1. Requirements

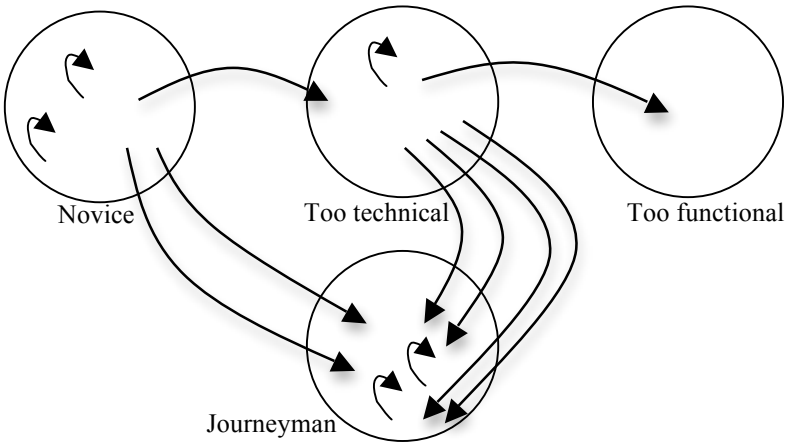

Figure 1. Trajectories of students' requirements knowledge.

Figure 1 presents our categorization of the students' requirements knowledge as manifested in the pre and post surveys. Each arrow represents the pre-to-post trajectory of a single student. The categories are defined as follows:

- Novice: The student gives no or very little thought to requirements
- Too technical: The student focuses almost exclusively on technical requirements (e.g., platform, system interfaces, data formats, etc.) to the exclusion of functional requirements; most students in this category also overlooked many common system quality attributes (e.g., performance, scalability, security, etc.)
- Too functional: The student focuses almost completely on functional requirements

- Journeyman: The student exhibits a relatively sophisticated and balanced view of requirements including awareness of users/stakeholders, the requirements process, features, prioritization, quality attributes/technical requirements, etc.

To illustrate the change in thinking about requirements, consider a student who transitioned from "novice" to "journeyman." In the pre survey, the student only considered change management policy. In the post survey, the student considered user roles, user pain points, processes for eliciting and analyzing user requirements, and weighing the strengths and weaknesses of candidate technologies. Another student transitioned from the "too technical" to "journeyman." In the pre survey, the student considered platforms, system interfaces, throughput, government standards, and security technologies. In the post survey, the student considered the targeted users, interview techniques, and requirements prioritization, as well as throughput, and data types.

3.4.2. Design

Eight of our students stayed at what we believe to be the appropriate level for this early stage of a project: They either did not discuss design at all, given that they do not yet completely understand the system's requirements, or they simply noted that an architecture and design phase would be required during the project. The other five mentioned such things as platform, language choice, and software frameworks, all of which are probably premature at this point in the project. We do not believe the task we posed to the students is suitable for assessing if they have achieved a journeyman's knowledge of design, and thus the task does not provide insight into their learning trajectory in that direction.

3.4.3. Engineering Management

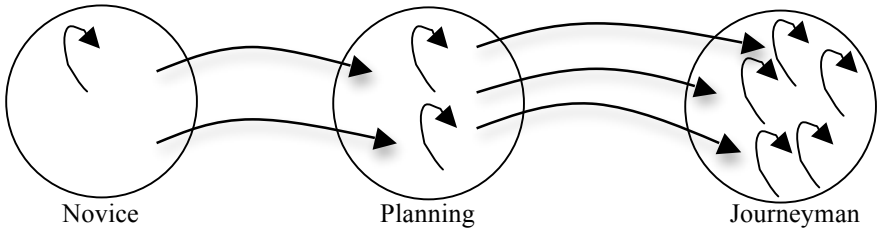

Figure 2. Trajectories of students' engineering management knowledge.

Figure 2 presents our categorization of the students' engineering management knowledge as manifested in the pre and post surveys. The categories are defined as follows:

- Novice: No or very little thought given to engineering management
- Planning: Focuses almost exclusively on aspects of project planning such as estimation, schedule, team, and project roles
- Journeyman: Sophisticated awareness of aspects of engineering management, including those listed in Planning, plus risks and risk mitigation, client relationship management, awareness of team strengths and weaknesses, metrics, meetings, etc.

To illustrate the change in thinking about engineering management, consider a student who transitioned from "Novice" to "Planning." The student initially did not give engineering management any thought (i.e., none of his responses were categorized as pertaining to engineering management). In the post survey, the student was concerned about project due-dates, estimation, customer feedback mechanisms, and the work break-down structure, but did not mention such things as metrics, risk mitigation, and team-related issues. Another student transitioned from "Planning" to "Journeyman." In the pre survey, the student considered planning, scheduling, and prioritization. In the post survey, the student again considered planning, scheduling, and prioritization and went on to consider client relationship management, estimation and re-estimation, budgeting, meeting management, and how to create and train the development team.

4. Discussion

Our statistical analyses provided limited insight into the state of students' knowledge, due in part to the relative "coarseness" of our survey-based approach and in part to our small sample. Most notably students' models of software engineering processes became richer as indicated by significantly more responses to question 2, "What software engineering processes do you need to put into place?," and by the highly significant pre to post difference in the number of engineering management responses to questions 2 and 3.

Combining the results of our statistical analysis with our qualitative analysis provided somewhat more insight. Students did not ask significantly more questions pre to post, but most asked better questions as evidenced by their "trajectories towards expertise" in the requirements and engineering management categories. We believe that asking better questions will provide better information and, thus, contribute to more effective problem solving (cf. the reported correlation between the quality of questions asked and achievement reported in [9]).

Our qualitative analysis based on (admittedly subjective) holistic clustering of the students provided the most interesting (but methodologically least sound) picture, suggesting that our students are, indeed, progressing fairly consistently towards journeyman-level expertise. In future work, we must better define the criteria and process by which this categorization is performed, ensuring that other researchers, as well as the authors, can perform it consistently.

Taken together, these results seem to confirm key hypotheses derived from the theory of transfer as preparation for future learning: Our students exhibit richer models of software processes after completion of our formal curriculum, they ask better questions, and nearly all are on trajectories towards expertise. That said, however, much work remains to be done to gain insight into the nature of our students' mental representations of software engineering knowledge and how those representations evolve.

5. Future Work

We are really just at the beginning of a very long project to understand the nature and evolution of our students' knowledge. Our preliminary results suggest several directions for future work. As noted above, our current survey-based method has provided a very limited window into the state of students' knowledge. As a methodological refinement, we will

consider ways of eliciting a richer picture of student knowledge; requiring students to provide explanations of their problem solving rather than simply short answers seems promising as a means of eliciting richer data (see, e.g., the elicitation and analysis of causal explanations of medical problem solving discussed in [10]). Further studies of this type should also assess the elaboration and refinement of students' mental models, as well as evolution of the relational structure among concepts and their component parts; we must adopt a richer ontology than SWEBOK – something in the spirit if not the letter of the Computing Ontology Project (e.g., [11]) – to provide a richer framework for the analysis of student knowledge.

Our (again admittedly subjective) approach to the qualitative analysis of student responses was sufficient for an initial study, but we must formalize our approach and codify the expertise-related category definitions. Doing so will enable us to determine if our approach is repeatable by other researchers and if our category definitions suffice to describe new populations of subjects.

The problem we posed, which deals with the inception of a new project, is most appropriate for eliciting responses in the Requirements and Engineering Management categories. We must formulate additional problems to which a broader range of knowledge (e.g., design, construction, and testing knowledge) can appropriately be brought to bear.

As noted earlier, our sample of 13 students was too small to yield statistically significant results in many cases in which pre to post response frequencies seemed to differ markedly. We must expand the sample of students. Our 16 new full-time software engineering students have all completed the pre survey; since they are demographically very similar to members of our current sample, we should be able to treat them as a single population for an expanded analysis.

Perhaps most important, we are attempting to characterize students' trajectories towards expertise, but we have not yet recruited a group of expert subjects to whom they can be compared. We propose to recruit and test such a group. Possibilities include recruiting early graduates from our software engineering program or graduates of Carnegie Mellon Pittsburgh's MSE program with at least five years of experience after receiving their master's degrees.

6. Acknowledgments

We are grateful to Carnegie Mellon Silicon Valley's full-time software engineering class of 2011 for their time and effort in completing the survey. We would also like to thank Natalie Linnell for many insightful comments on an earlier draft of this paper and especially the CSEET reviewers for their careful reviews.

7. References

[1] R. Bareiss and E. Katz, “An Exploration of Knowledge and Skills Transfer from a Formal Software Engineering Curriculum to a Capstone Practicum Project,” Proceedings of CSEET 2011, Honolulu, HI, May 2011.

[2] D. N. Perkins and G. Salomon, “Transfer of learning,” International Encyclopedia of Education (2nd edition), Oxford, UK: Pergamon Press, 1992 (Accessed via http://learnweb.harvard.edu/alps/thinking/docs/traencyn.htm).

[3] J. Bransford and D. Schwartz, “Rethinking Transfer: A Simple Proposal with Multiple Implications,” *Review of Research in Education*, 24(1), 1999, pp. 61-100.

[4] K. Ericsson, M. Prietula, and E. Cokely, “The Making of an Expert.” Harvard Business Review. 2007.

[5] R. Bareiss and T. Sedano, “Developing Software Engineering Leaders,” Proceedings of the First International Symposium on Tangible Software Engineering Education (STANS-09), Tokyo, Japan, October 2009.

[6] E. Katz, "Software Engineering Practicum Course Experience," Proceedings of CSEET 2010, pp.169-172.

[7] A. Abran, J. Moore, P. Bourque, and R. Dupuis, editors, “Guide to the Software Engineering Body of Knowledge: 2004 Version,” IEEE Computer Society, 2004 (Downloaded from http://www.computer.org/portal/web/swebok).

[8] J. Tabaka, Collaboration Explained: Facilitation Skills for Software Project Leaders (Chapter 15), Boston: Addison-Wesley, 2006.

[9] A. Graesser and N. Person, “Question Asking During Tutoring,” American Educational Research Journal, 31(1), 1994, pp. 104-137.

[10] C. Hmelo, “Problem-based learning: Effects on the early acquisition of cognitive skill in medicine,” Journal of the Learning Sciences, 7, 1998, pp. 173-208.

[11] L. Cassel, G. Davies, R. LeBlank, L. Snyder, and H. Topi, “ Using Computing Ontology as a Foundation for Curriculum Development,” Proc. SWEL@ITS ’08: The Sixth International Workshop on Ontologies and Semantic Web for E-Learning, Montreal, 2008, pp. 21-30.