Teaching Computer Science through Problems, not Solutions

SAMUEL B. FEE & AMANDA M. HOLLAND-MINKLEY*
Information Technology Leadership, Washington & Jefferson College, Washington, PA, US

Regardless of the course topic, every instructor in a computing field endeavors to engage their students in deep problem-solving and critical thinking. One of the specific learning outcomes throughout our computer science curriculum is the development of independent, capable problem solving – and we believe good pedagogy can bring such about. Our experiences indicate to us that students improve their ability to analyze and solve complex computational problems when we pursue pedagogies that support them in developing these skills incrementally. Specifically, we pursue a problem-based learning approach that we apply individually in each course as well as across the entire curriculum of our department, instead of solely considering our pedagogy on a course-by-course basis.

Keywords: *problem-based learning; computer science curricula*

1. What is Problem-Based Learning?

Problem-Based Learning (PBL) is a pedagogy that centers student learning around open-ended, student-driven problems facilitated by an instructor in order to achieve the learning outcomes of a course. It appeals to a cognitive constructivist epistemology which concludes from study and experience that learners gain more through relating educational material to their own real-life experiences, and that such experience informs their ability to conceptualize content (Duffy & Jonnasen, 1992). Constructivism calls for learning opportunities that are experiential, active, collaborative, and that also develop problem-solving skills (Jonnasen, 2000). The goal here for the learner is not to passively absorb and regurgitate information; but rather to actively engage with the content, work through it with others, relate to it through an analysis with personal experience, and effectively solve problems with the corresponding knowledge gained. Thus the ultimate goal is the development of critical-thinking abilities.

This of course means that the student is an active participant in the learning process (Bonwell & Eison, 1991). The result is a necessary relaxing of the traditional classroom structure so that students can pursue ideas in a fashion that makes sense to them individually, rather than the specific prescribed approach that the instructor may have in mind. Indeed, many approaches could be relevant for attaining the knowledge developed by the intellectual task at hand. So, students need to be free to develop those knowledge constructions in their own way. This does not mean that there is no structure to the process as some might suggest (Kirschner, Sweller & Clark, 2006). But rather, that a looser structure governs the endeavor and allows the student to maneuver in several different directions under the guidance of an engaged instructor. Of course, there are numerous pedagogical approaches that enable this way of knowing. But one of the more promising approaches that has served us well in the need for developing critical thinking skills is problem-based learning.

Problem-based learning has seen its largest and earliest adoption in the medical education field. Specifically, the approach orients students "toward meaning-making over fact-collecting" (Rhem, 1998). However, much of the recent literature clearly articulates the pedagogical approach and specifically discusses it in relation to various fields of inquiry outside of medical education. For an in-depth discussion, the inaugural issue of *The*

* Corresponding author. Email: amh@washjeff.edu

Interdisciplinary Journal of Problem-based Learning contains a particularly useful introduction that lays out the essential elements of PBL (Savery, 2006).

For our work, we have found several underlying themes to be particularly cogent toward our instructional activities. Specifically, PBL provides a *context* for the content of our courses; meaning that many of the individual concepts that each course identifies are drawn together to solve the problems that are submitted to students through the various projects that they work on during their coursework. It is important then to draw the distinction that PBL presents specific problems for students to solve – this is not the same as working on projects. The advantages of project work in computer science courses, such as the increased engagement of students and the relationship to professional practice, are well understood (Fincher & Petre, 1998). Thus it is common to assign projects as a way for students to illustrate their learning, and indeed to assess that learning. We do the same. The importance here is recognizing that we are not solely doing project work, but also solving specific problems based upon the content knowledge that has been developed in class. Such scaffolding is integral to the process as students use the content provided along with their own experience, to construct knowledge relating to the problem at hand (Collins, et al. 1991). PBL and project-based learning are similar in that there is a shared goal of successfully completing the activity; however, different in that projects typically posses more structure to a supplied and more explicit goal with more direct guidance from the instructor. Problems on the other hand focus on the learner's role in identifying outcomes and parameters for success (Savery, 2006).

When implementing PBL, we are trying to help students find solutions to problems contextualized specifically within the content of their courses. Typically, these experiences take the form of group projects; but the work can be pursued effectively on an individual basis as well. The process is by no means easy – developing good problems for student to solve is a critical step in providing effective instruction (Duch, Groh & Allen, 2001). These problems need to be reasonably well understandable as students begin developing their problem solving skills, but increasingly ill-structured as students progress throughout their coursework. Developing quality problems is quite hard – especially when considering the disparity of student knowledge in any given class – and considerable instructor time must be devoted to the process. Furthermore, PBL requires considerable amount of effort working directly with students, as the instructor must be available to mentor students in the problem-solving process. Such mentoring is increasingly diminished as the students become more capable of such thought on their own; but at the introductory level this mentoring requires considerable effort on the part of the instructor to remain effective.

While PBL can be applied in any discipline, its appeal within computer science is clear. Many of the courses, such as programming or software engineering courses, are essentially courses designed to teach problem solving. With the rapid advances within the field, it is also of particular concern that students understand how to be good independent learners. We will particularly value a pedagogy that results in graduates who are able to educate themselves about new technologies and integrate them into their repertoire of problem-solving tools. In the ACM's 2008 computer science curriculum interim revision, six characteristics of computer science graduates were described: a systems-level perspective, an appreciation of the interplay between theory and practice, a familiarity with common themes and principles, significant project experience, attention to rigorous thinking, and adaptability (ACM, 2008). The PBL approach directly supports the last three characteristics, and makes itself open to, with a careful selection of problems, development of the first three characteristics as well.

2. Applying Problem-Based Learning at the Curricular Level

Often problem-based learning is described as it applies to a single course. Instructors are encouraged to give students small initial problems and gradually add complexity while removing specific guidance. The hope is that by the end of the course students will be able to tackle real-world problems, and ideally broadly-specified problems that they have selected for themselves. Students will take increased ownership of their learning and will experience the pitfalls and dead ends of a realistic problem-solving process. This is a very appealing picture, but when faced with the realities of the average college classroom, a pure application of the problem-based learning pedagogy can be challenging.

Students often enter the ITL major at W&J with an interest in the particular tools that they will be using, or a desire to construct the types of products they see other students generating in the courses. While this is natural, and in fact provides good motivation to the students as they pursue difficult tasks, few students enter our courses with a stated intention of becoming better problem solvers. They enter wishing to learn Java, SQL, or PHP, or perhaps to build a dynamic website, analyze their network traffic, or develop a video game. An appreciation for the general skill of problem solving only comes later in the curriculum, or after they have graduated and entered the work force. This provides a particular challenge when making this skill central to our curriculum.

One of the frequent complaints that we hear from our students is that they are being asked to do things we "have not shown them how to do." Inevitably, what this means is that we have shown the students all of the pieces that will be required to solve a problem, but we have not shown them explicitly how to coalesce these together in order to reach their goal. From an instructor's perspective, this is purposeful. The process they are expected to follow will have been illustrated and practiced, but the student is now invited to explore how this process applies to a new setting. Without experiencing the process of experimenting with possible solutions and pursuing paths to determine if they are fruitful or not, students will not be able to become self-sufficient problem solvers. But the focus on independent solving and student-directed exploration encouraged in problem-based learning can lead to novice students feeling overwhelmed by the degree of flexibility they have been permitted (Kay et al., 2000). There is also a conception on students' parts that assignments will ask them to do highly structured problems that closely resemble problems they have been shown, and that can be solved by applying the same steps with few adjustments. They may not be familiar with the complexity of decomposing a problem into its parts and searching for a solution, particularly when an aspect of the problem is defining what a good solution looks like. There is value in relating this disconnect between a student's perception of what a problem is and an instructor's, or an employer's, perception to the notion of a culture clash between students and faculty that must be broached (Kolikant & Ben-Ari, 2008). The early stages of PBL can be treated as a tool for closing the cultural gap by exhibiting for students an alternate model for problem definition and solution. Instructors then must expect to introduce their students to this skill and help them develop their problem solving abilities. While it is appropriate to allow students a certain degree of mental discomfort and uncertainty, it is important that the creative leaps they are being asked to make are within their grasp and that they can see a path to success with reasonable effort on their part. Our challenge is to allow students to experience these frustrations while preventing their frustration from becoming so severe they give up. The scaffolding provided by a PBL approach – especially when applied across numerous courses within a curriculum – leads students effectively along this path.

Given the centrality of problem-solving to all academic endeavors, and to the entire breadth of a computer science curriculum, it seems unrealistic to expect students to achieve a high level of proficiency during a single course, particularly given the modest initial starting point of many new college students. It seems

preferable to view problem-solving skills as something that must be developed over a longer stretch of time - most likely the student's entire college career. For novice problem-solvers, simply familiarizing them with the idea that there may be more than one way to approach a problem, and that exploration is encouraged, may be as large a step as they are prepared to take. It is, however, a step that would set these students up for greater success in later courses that presuppose that initial comfort with the tenets of problem-based learning. For this reason, we propose using problem-based learning as the framework for an entire curriculum.

There are secondary advantages to integrating problem-based learning across a curriculum. This approach offers an ideal opportunity to encourage students to pursue self-regulated learning. Students are supported in self-regulated learning when they are given opportunities, ideally explicitly through classroom tasks, to practice the necessary skills of managing their own learning, including time management, goal awareness, and appropriate use of peers and faculty in supporting their learning (Pintrich, 1995). Like problem-solving, self-regulated learning must be developed incrementally over time and the same activities that model the breaking down of problem solving can also be used to illustrate to students a general process for learning. Additionally, because of the tensions described above that some students may experience when first enrolling in a PBL course, by implementing the approach across all courses in the discipline, students are shown that this is a valid way of pursuing computer science education and come to expect this approach in their later courses without significant orientation. Some PBL practitioners have also expressed concern that when PBL is implemented in introductory courses but not followed at the higher level, the positive retention effects seen early on may be reversed as students are faced with a more conventional course format (Kay et al., 2000).

It is worth noting that implementing a problem-based curriculum represents a significant commitment on the part of the faculty involved. As we discussed above, mentoring students through their individual struggles with problem solving experiences can be time intensive. When students are encouraged to develop their own problems and solution strategies, faculty must plan on spending additional time during the semester to support these efforts. For these reasons, a problem-based curriculum is likely to be most effective as a conscious effort on the part of an entire department, where the additional workload will be recognized and supported. Course-load decisions and scheduling can be done with an awareness of and respect for the obligations of PBL, particularly at the introductory level.

At a high level, then, when applying problem-based learning across a curriculum, we propose that instructors should plan on having entry level courses simply introduce students to the process of problem solving and the fact that they may be asked to approach novel problems with a willingness to explore and innovate. Instructors should expect to provide a fair degree of guidance at this level about how particular problems ought to be solved, leaving students with modest creative gaps to fill in. As students progress through a degree program, the scope of the problems they will be expected to tackle will increase in complexity as the degree of guidance they are given about how to solve the problem will decrease. The goal, by the end of their college career, is to be able to take a significant real-world problem requiring an entire semester's worth of effort and complete control over the process and produce results satisfying to a real-world standard.

3. Information Technology Leadership and the Liberal Arts

At Washington & Jefferson College, the Information Technology Leadership department selects from more traditional IT and computer science curricula to design an interdisciplinary computing major that integrates strongly with the liberal arts tradition. The ITL major allows students to customize a selection of courses that meet their long-term goals, whether they intend to pursue a directly technical career or one where technical skills play a valuable supporting role. Graduates of our program have gone on to graduate studies in

information security, IT management, and entertainment technology. They have pursued traditional programming and system administration careers but have also pursued more entrepreneurial ventures and branched out into human resources and technical writing. This variety of goals is supported by offering students a flexible curriculum. ITL majors can select upper-level courses from any of three different emphases including computer science (covering data structures, security, artificial intelligence, and systems analysis), data discovery (covering data mining, geographic information systems, advanced databases and web-based database development), and new media (covering digital imagery, digital video, web development and graphic design). This flexibility enables students to build connections between these areas of study in a way that makes sense to them intellectually, and in a fashion that also supports their diverse career goals. In addition, we have worked to minimize long chains of pre-requisites in the major requirements to allow students multiple entry points to the curriculum as well as the ability to take those courses that most interest them early while they are still gauging their interest in the program.

With these goals in mind, there are three important learning objectives that underlie all of our courses. The first is that students will graduate with a robust understanding of the vastly interdisciplinary nature of computing. The second is that students will graduate with strong leadership skills, including effective technical communication with both peers and non-technical people and project management abilities. The third is that students will not only be able to use particular technical skills to solve familiar problems but will have well-developed and flexible problem solving skills.

Within our curriculum we allow students to select from courses in three highly divergent emphases to form one cohesive major. By offering a single major, we are asserting that regardless of one's emphasis there is a common set of knowledge and abilities that all students will have, and furthermore that set of knowledge and abilities should support the area-specific work within each of the emphases. For example, every student receives basic instruction in programming, databases, ethical and historical issues in computing, and human-computer interaction. We view a central theme of the major to be developing problem-solving skills, using computing as a tool in that process. Given the rapid growth in technology, it is impractical to try to graduate students with the specific skills they will need in future employment. This is particularly true in an interdisciplinary program where we cannot predict the specific domains within which our students will apply their knowledge. By focusing primarily on the problem-solving skills that are used across domains, we equip students for the diverse set of problems they may find themselves facing.

4. A Problem-Based Learning Liberal Arts IT Curriculum

The Information Technology Leadership department at Washington & Jefferson College follows a curriculum that implements problem-based learning across the scope of the entire major. Taking this approach, we see different types of work expected of students in different levels of courses as they proceed through the ITL curriculum. At the 100-level, students do frequent small-scale exercises illustrating specific skills, and then as the course proceeds begin to combine these specific skills to solve more complex problems. While students may be given larger problems to solve late in the course, these are generally accompanied with guidelines on how to decompose the problem into pieces of a magnitude more familiar to the student. A clear description of a successful solution is provided.

Moving on in the curriculum at the 200-level, students begin to be expected to practice specific skills on their own after instruction. The problem-solving starts at an intermediate level, but specific guidance is still given on large-scale problem solving. There is still fairly explicit instruction on what features a successful solution will have. As students reach the 300-level courses, they are expected to build on the skills they have obtained

earlier in the curriculum to develop solutions to open-ended problems with feedback from the instructor to help keep them on the right track. There are fewer small assignments, as students are expected to identify the component tasks or skills within the larger problems that they are solving. There is significant responsibility on the student to shape the problem being solved and determine for themselves what a successful solution will look like.

The application of problem-based learning to the ITL curriculum culminates in requiring all students to take part in a 400-level capstone course titled Service Learning Project Management. The capstone requires all students to tackle full scale problems without advance guidance in how the problem should be decomposed. Specifically, students work in small teams to address the computing needs of a local non-profit as part of a service-learning project. By this point in the program, students have the technical skills to complete the projects, and the larger challenge is exercising the project-management skills to determine for themselves how to manage the entire problem-solving process.

To help illustrate how we implement problem-based learning across our curriculum, we focus below on how the computer science emphasis courses in particular shepherd students through successively high-level and independent problem-solving tasks. Given below are some illustrative examples of problems, but this is not intended to be an exhaustive listing of every such problem students would encounter. Instead we have focused on the most significant project assigned for each representative course discussed.

Representative Course: Introduction to Programming (100-level)

Example Problem Given: Implement *Missile Command* using threads and arrays. (final course project, following several shorter assignments, 2.5 weeks duration)

Level of Guidance:

- Relevant sample code for threads given.
- Ordered step-by-step description of subtasks.
- Specification to use arrays to store data on incoming missiles and active targets.
- Provision of basic equations for distances between points, etc.
- Indication of time needed per subtask.
- Instructions on testing for each subtask.
- In-class exercises generating code directly related to the more complicated subtasks.

Learning Objectives/Outcomes:

- Understand and add to provided code.
- Solve a complex problem following a detailed solution specification.
- Create a longer implementation out of components the size of previously completed assignments.
- Successfully work with a specified data structure across all components of a project.
- Manage completion and testing of subtasks on a schedule.

Representative Course: Data Structures (200-level)

Example Problem Given: In 2-3 student teams, select one of a set of proposed projects requiring a significant degree of data manipulation and create an efficient implementation, e.g. book recommendation engine. (significant course project following half dozen traditional assignments, 6 weeks duration)

Level of Guidance:

- Sample projects suggested, but students must provide details.
- Weekly graded deadlines set for project review and feedback by instructor.
- Explicit statement of task assignment to team members for each week required and approved.
- Data structure choice and implementation strategy approved at first weekly deadline.
- Class time spent meeting with groups, discussing approach and timeliness of progress.

Learning Objectives/Outcomes:

- Propose a feasible task decomposition of a significant project.
- Select and argue for the appropriate data structure and algorithms for the selected project.
- Meet weekly team obligations, with occasional instructor support.
- Implement and use studied data structures and algorithms following agreed upon interfaces.
- Develop group problem solving skills in a structured environment.

Representative Course: Artificial Intelligence (300-level)

Example Problem Given: In 3-4 student teams, propose and implement a project requiring the use of one of the studied AI algorithms, e.g. robot path planning. (significant course project following half dozen traditional assignments, 6 weeks duration)

Level of Guidance:

- Initial project proposals are given feedback before final proposal to be met is submitted.
- Proposal for particular algorithm(s) to employ and reasons for rejecting competing alternatives submitted and approved prior to implementation.
- Team plans for scheduling and workload distribution are approved but not monitored.
- Instructor available for outside-of-class group meetings as initiated by teams.

Learning Objectives/Outcomes:

- Set and manage own problem decomposition, with guidance.
- Select and justify choice of the most suitable algorithm out of competing approaches.
- Definite standards of success for an open-ended AI problem.
- Deliver an extensive implementation as part of a student-run team.

Representative Course: Service Learning Capstone (400-level)

Example Problem Given: In 3-4 student teams, design and implement a solution to a current technology problem facing a local non-profit organization. (single course project, entire semester duration)

Level of Guidance:

- Instructor introduces team members to organization liaison.
- Deadlines given for initial project scope document and final deliverable.
- Presentation of scope document to department faculty with feedback provided.
- Instructor leads general course discussions related to readings on project management.

Learning Objectives/Outcomes:

- Student teams manage relationship with and needs assessment for organization.
- Student teams internally manage establishing deadlines and task assignments.
- Appropriate training provided for organization.
- Successful resolution of organization's problem, with specification of further steps needed as called for.

As has been noted, in the lower level courses students generally do not spend the entire course solving a single problem. In Introduction to Programming, each concept is practiced during class with on-the-spot activities directly connected to the material and then weekly homework generally requires students to combine a few of these concepts to solve a relatively small problem. As the class progresses, the in-class activities often take the form of giving students a broad problem to try to solve, and then after the students have had time to think about the problems, having the instructor model talking through their own problem solving process at the front of the room, writing code, and naturally having false starts and bugs to fix. These classroom and homework activities provide students with the necessary scaffolding to succeed on the larger problem presented at the end of the course.

We can also see above the way in which scaffolding is provided by the progression of courses. When students are asked in Data Structures to create a plan for solving their problem, they are explicitly told to think back on the specification they were given at the end of Introduction to Programming and to model their plan after that. Students not only have a model of what they should be doing, but then have experienced the advantages of viewing a problem as an organized ordered set of component steps, reducing the perception of the task as busywork delaying them from really starting the project.

This pattern continues when it comes to the types of interactions instructors have with student teams in 200-level versus 300-level courses. For the data structures project described about book recommendations, given the content of the course, the project focused on storing and efficiently accessing the collected data about each of the texts. It would have been simple for this project to expand to a point where it was intractable given the knowledge of the students and the timeframe for the project, and a significant amount of instructor effort went into ensuring that the scope of the project was kept at a reasonable level. It was not assumed that the students could yet anticipate what could or could not be accomplished during the time allotted and with the techniques they had learned. Rather, the instructor's role was to act as a reality check on the regular progress reports and proposals for what to accomplish in the coming week. However, this same project could be used very suitably in our 300-level artificial intelligence course. In that setting, both because of the topic of the course, and more

importantly because of the level of the course, it would be expected that students could determine for themselves what the appropriate scope for the project ought to be. While advice from the instructor would, of course, also be available, students would be expected to lay out what they believed to be a reasonable set of features to include. Deviations from that feature set would be permissible only through a written update to their originally submitted project specification. In this way, the students become responsible not just for finding a solution, but also for defining the problem itself – often an integral part of the problem-solving process that students miss out on.

We would also like to note that not all of the problem-solving that we ask students to do takes place in a programming or technical context. For example, all majors and minors in our program are required to take a 100-level course, IT & Society that introduces students to the history of and social issues in computing. This is a reading and writing intensive course. While students usually enter this course having already completed an English composition class, we focus on the particular difficulties students have in reading technical content and judging the credibility of various sources. We also begin instructing students on how to write technical content that is accurate and appropriate for the intended audience. As in our introductory programming course, our goal in this initial course is largely just to make students aware of the complexities they will have to face when reading and writing about technology.

For example, often students enter this course having learned to read a story or historical account, but poorly equipped to read a description of a software or hardware technology and extract the important details that would allow them to assess the applicability of that technology to a particular problem or analyze its social impact. They are well shy of the ability they will need in upper-level courses of reading about a technique or algorithm and being able to implement it in order to reach a higher-level goal. In order to work students towards this necessary analytical skill, in IT & Society we explicitly model for students the work that goes into effective reading. We may give samples of the questions that they ought to be asking themselves and require them to practice answering those questions. We may spend class time actively reading together and discussing what we are focusing on and why. By explicitly modeling the analytical complexities of reading at the introductory level, we set the stage to expect them to practice this skill effectively and independently at the upper-levels.

While our tables above illustrate how four courses students in the CS emphasis would experience our curriculum, the same general breakdown of the level of guidance and learning objectives, with slight shifts for subject content, would also apply in our New Media and Data Discovery emphases. In fact, with 100-level courses within the department required of students in all emphases, all students are guaranteed to get the same initial experience of solving a significant problem while being supported through detailed guidance. We believe it is a sign of the success of our approach that the 400-level capstone is again a shared experience, with second-semester seniors from all three emphases enrolled in the same course and often working on teams mixed across the emphases. Regardless of the subject matter specialization students have selected, they are all expected to enter that course prepared for the independent, real-world problem solving required of them.

The capstone projects epitomize the type of real-world robust problem solving that problem-based learning strives to have students ultimately be able to accomplish. Rather than being the end product of a single course, these projects are really the end product of an entire curriculum – again, not just the content of that curriculum, but also the independence and critical analysis skills that are intentionally developed through the progressive courses. The course instruction is spent helping students make the final transition of applying their problem solving skills in a real world setting where money, organizational mission, and the needs and

priorities of a client become part of the equation as well. This is possible because of the shared preparation that all of the students taking part have had, whether they have taken the same prior courses or not. While they may start unaware that problem-solving is a skill they must learn, by the time they graduate they are capable of explicitly discussing this issue and how they must adapt their problem-solving strategies to the requirements of their team and their client.

5. The Successes of a Problem-Based Curriculum

The successes of our senior students in their capstone course give us confidence in the success of our curriculum. Over the past five years all 52 students who have taken part in the ITL capstone have successfully completed it. The capstone projects require students to define the problem that needed to be solved, identify a solution that could be implemented within the time frame of one semester and with the resources available, and deliver that solution. Recently students have completed projects ranging from constructing a preliminary GIS system for an ecological activism organization to creating a promotional film for an arts education organization. We believe these accomplishments, along with the students' ability to explain their problem-solving process, are positive signs for our curriculum helping students develop independent problem solving skills.

We are also interested in whether the successes our students have while at W&J translate into successes in their later careers. The ITL department is still quite new, so there are a limited number of graduates to survey, particularly when focusing on those who have been out of school long enough to usefully reflect on the benefits and shortcomings of our curriculum and its pedagogy. We do intend, once sufficient students have graduated with an ITL major and spent at least 2-3 years in graduate school or the workplace, to enact a rigorous analysis of our program outcomes. In the short term, however, we have been able to solicit feedback from eight students who have graduated in the past few years, and maintained contact with one or more members of the department. We asked them to reflect on the program, our curriculum, and how it has prepared them for their current professional activities. These are only very preliminary results, but the commonalities in the responses strongly reflect the advantages to our approach that we had hoped to see commented on by our alumni.

Looking at their current careers, every student reported that their ITL courses were essential to their current professional activities, and students consistently identified not just the content of the classes but the problem-solving aspects of their education in general as being crucial to their current successes. One graduate from the New Media emphasis credited her introductory programming and 200-level data mining courses as giving her a solid grounding in problem-solving that she uses today in her job. Another New Media student reported:

> "I don't work directly in New Media now but I'm still required to provide a very high level of work, just like I was then. I got used to sitting in front of the computer for hours and sticking with a project until it was finished, even if it required exploring different strategies to get the desired product."

Some students talked about putting their time-management and project-management skills to use. Said one:

> "I think the most valuable skills I learned in ITL was how to manage my projects better. I always felt like each course spent a lot of time working on the baseline before tackling the main objective. In my everyday work, I spend a lot of time making sure I understand the fundamentals before trying to finish the final project."

Another student, reflecting on the curriculum as a whole, said simply: "All courses promoted independent learning, which is essential in the work world."

When asked what made the problems in their courses more than just projects, students consistently mentioned the same properties of a problem that we stressed earlier in the paper that make it more robust than simply a project: that it is a realistic problem where the student defines the scope and the outcomes for success. Every student cited an example of a "real-world" problem they were able to solve, often from their senior capstone. But some went further and discussed the way in which the project required "original thought" or "critical thinking" and some talked explicitly about the portions of the problem that involved determining what realistically could be accomplished and how:

> "We had to figure out who could do the work, when the work could be done, when we could meet as a group. […] This experience was more problem solving than working on a project because the final result was public facing, and there was a certain feeling of it being more "real world" than any other project. Also, managing people is a problem that people face every single day in the work place, and my experience with planning will help me in the future."

A few students also talked about the need for innovative work in the context of having to do research. The themes of applicability and personal connection to and ownership of the project were quite clear from all of the responses.

While these responses come from a limited sample of students, we see that when these alumni reflect on their learning, they are aware of exactly those advantages of problem-based learning that we hope to achieve. They believe that they have become good problem-solvers, and understand the role of independent learning and a focus on real-world applicability to good problem solving. They understand, in retrospect, how deep problem-solving skills have been built up as they progress through the curriculum and are aware that it is not just the subject-related prerequisites that help them succeed in their upper-level courses, it is also the general skills they are acquiring in all of their lower-level courses combined that allow them to tackle real problems at the end of their schooling and beyond. We are hopeful that future surveys will show the same positive reflections amongst our alumni as a whole.

References

Barrows, H. S. (1996). Problem-based Learning in Medicine and Beyond: A Brief Overview. In L. Wilkerson & W. Gijselaers (Eds.). *Bringing problem-based Learning to Higher Education: Theory and Practice. New Directions For Teaching and Learning Series, No. 68*. San Francisco: Jossey-Bass.

Ben-David Kolikant, Y., & Ben Ari, M. (2008). Fertile Zones of Cultural Encounter in Computer Science Education. *Journal of the Learning Science.* 17(1).

Bonwell, C. & Eison, J. (1991). *Active Learning: Creating Excitement in the Classroom*. Washington DC: ASHE-ERIC Higher Education Reports.

ACM. (2006). Computing Curricula 2005, The Overview Report. http://www.acm.org/education/education/curric_vols/ CC2005-March06Final.pdf

ACM. (2008). Computer Science Curriculum 2008. http://www.acm.org//education/curricula/ComputerScience2008.pdf

ACM. (2008). Curriculum Guidelines for Undergraduate Degree Programs in Information Technology. http://www.acm.org//education/curricula/IT2008%20Curriculum.pdf.

Belland, B. French, B. & Ertmer, P. (2009). Validity and Problem-based Learning Research: A Review of Instruments Used to Assess Intended Learning Outcomes. I*nterdisciplinary Journal of Problem-based Learning*, 3(1).

Boud, D. & Feletti, G. (1997). *The Challenge of Problem-based Learning (2nd ed.).* New York: Routledge.

Collins, A. Brown, J.S. and Holum. A. (1991). Cognitive Apprenticeship: Making Thinking Visible. *American Educator.* 12 (6).

Duch, B. Groh, S. & Allen, D. *The Power of Problem-Based Learning: A Practical "How To" for Teaching Undergraduate Courses in Any Discipline*. Sterling, VA: Stylus Publishing.

Duffy, T. & Jonnasen, D. (1992). *Constructivism and the Technology of Instruction: A Conversation*. Philadelphia, PA: Lawrence Erlbaum.

Fellows, S. & Ahmet, K. (1999). *Inspiring students: Case studies in Motivating the Learner.* London: Kogan Page.

Fincher, S. & Petre, M. (1998). Project-based learning practices in computer science education. *Proceedings of the 28th Annual Frontiers in Education – Volume 3*. Washington DC: IEEE Computer Society.

Garćıa-Famoso, M. (2006.) Problem-based learning: a case study in computer science. *Proceedings of the Third International Conference on Multimedia and Information & Communication Technologies in Education*.

Hmelo-Silver, C. E. (2004). Problem-based Learning: What and How Do Students Learn? *Educational Psychology Review.* 16(3).

Jonassen, D. (2000). Toward a Design Theory of Problem Solving. *Educational Technology Design and Development, 48(4)*.

Jonnasen, D. (2004). *Handbook of Research for Educational Communications and Technology (2nd)*. New York: Lawrence Erlbaum.

Jonnasen, D. & Hung, W. (2008). All Problems are Not Equal: Implications for Problem-Based Learning. *The Interdisciplinary Journal of Problem-based Learning.* 2(2).

Kay, J. Barg, M. Fekete, A. Greening, T. Hollands, O. Kingston, J. H. & Crawford, K. (2000). Problem-based learning for foundation computer science courses. *Computer Science Education*. 10(2).

Kirschner, P. Sweller, J. & Clark, R. (2006). 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).

Lave, J. & Wenger, E. (1991). *Situated Learning: Legitimate Peripheral Participation*. Cambridge: Cambridge University Press.

Maudsley, G. (1999). Do We All Mean the Same Thing by "Problem-based Learning"? A Review of the Concepts and a Formulation of the Ground Rules. *Academic Medicine.* 74(2).

Mergendoller, J. R., Maxwell, N. L., & Bellisimo, Y. (2006). The Effectiveness of Problem-Based Instruction: A Comparative Study of Instructional Methods and Student Characteristics. *The Interdisciplinary Journal of Problem-based Learning.* 1(2).

Pintrich, P. (1995). Understanding Self-regulated Learning. *New Directions for Teaching & Learning,* 3(10).

Resnick , L. B. (ed). (1989). *Knowing, Learning, and Instruction: Essays in Honor of Robert Glaser*. Hillsdale, NJ: Lawrence Erlbaum.

Rhem, J. (1998). Problem-Based Learning: An Introduction. *The National Learning and Teaching Forum.* 8(1).

Savery, J. (2006). Overview of Problem-based Learning: Definitions and Distinctions. *The Interdisciplinary Journal of Problem-based Learning.* 1(2).

Schwartz, P. Mennin, S. and Webb, G. (eds.) (2001). *Problem Based-Learning: Case Studies, Experience and Practice*. Routledge.

Strobel, J., & van Barneveld, A. (2009). When is PBL More Effective? A Meta-synthesis of Meta- analyses Comparing PBL to Conventional Classrooms. *Interdisciplinary Journal of Problem-based Learning,* 3(1).

Thomas, J. (2000). *A Review of Research on Problem-based Learning*. San Rafael: The Autodesk Foundation.

Uden, L. & Beaumont, C. (2005). *Technology and Problem-based Learning*. London: Information Science Publishing.

Walker, A., & Leary, H. (2009). A Problem-based Learning Meta-analysis: Differences Across Problem Types, Implementation Types, Disciplines, and Assessment Levels. *Interdisciplinary Journal of Problem-based Learning*, 3(1).

Wilkerson L. & Gijselaer, W. (1996). Bringing Problem-based Learning to Higher Education: Theory and Practice. *New Directions in Teaching and Learning.* No. 68. San Francisco: Jossey Bass.

Wilson, B. (1995). *Constructivist Learning Environments: Case Studies in Instructional Design*. Englewood Cliffs, NJ: Educational Technology Publications.