Integrating Industrial Practices in Software Development through Scenario-Based Design of PBL Activities: A Pedagogical Re-Organization Perspective

Kam Hou VAT
University of Macau, Macau SAR, China

fstkhv@umac.mo

Abstract

This paper investigates a pedagogic model appropriate to the integration of the industrial practices in software development into the learning activities of our undergraduate students, especially in the context of group-based project work. Specifically, we are interested in the potentialities of this model enhanced from the problem-based learning context, such that people collaborating in the peculiar scenario of project development, are empowered to be more sensitive and reflective of their learning experiences. Our discussion describes a practical framework of course enactment taking into account the suggestions of the latest curriculum guidelines stipulated in the final draft of the "Computing Curriculum – Software Engineering" created by the Joint Task Force on Computing Curricula of the IEEE Computer Society and the Association for Computing Machinery. Namely, software engineering education in the 21st century needs to move beyond the lecture format, and should consider the incorporation of a variety of teaching and learning approaches, one example of which includes the constructivist model of problem-based learning (PBL) considered as appropriate to supplement or even largely replace the lecture format in certain cases. In the paper, a pedagogical re-organization perspective is presented as a way to conduct teaching in the area of software engineering. In particular, the connotation of problem-based learning in the education of future software practitioners is explored from a teacher-researcher's position, through the practice of action research. The paper concludes by emphasizing the contextualized learning scenarios involved in PBL, which have been observed to enable our students to experience the real-world practice of software development, and acquire valuable learning through teamwork that should remain with their future careers.

Keywords: Collaboration, design scenarios, problem-based learning, software engineering education.

Introduction

In the traditional model of education (Vat, 2002, 2003, 2004e, 2004f), learning design proceeded in a linear fashion from defining objectives to lesson planning to course delivery. Educators first engaged in a comprehensive learning needs analysis process, often based on assessments done by others about competencies and learning objectives. Comprehensive

syllabi were developed. Finally, the course was delivered as planned. In fact, most computing courses involve setting "problems" which students are required to complete. We often refer to these problems as exercises because they are small and well defined. We used them extensively in our conventional course: there were weekly exercises, each focused on particular detailed aspects of the course, usually one that had been center-stage in the recent lectures; there were larger assignments which integrated many aspects of the course, but were still quite tightly defined, to the point where their correctness can somehow be assessed using automatic grading system. Undoubtedly, content is important, but more attention should be paid to the process. Today, a renewed mindset (Vat, 2006) is needed for education, especially when it is offered through the perspective to achieve collaborative learning, an important ingredient in the education of an engineering professional. Within the context of a computing curricula, class time should also be devoted to such generic problem-solving skills as defining a learning plan, brainstorming to get started on a problem, reflection, articulation of problems and solutions, self-assessment, practice in active listening, and other communication skills. These aspects must be assessed and contribute to the grade awarded. Besides, teaching and learning must be seen as an ongoing process rather than a program with a fixed starting and ending point. The importance of widespread participation by learners in the design of their own learning must be emphasized (Kimball, 1995). The key is to design a framework for collaborative work, which requires students to grapple with roles, protocols for working inter-dependently and mutual accountability. In this regard, problem-based learning (PBL) serves as a very good instrument to experience group learning, which is also essential to fulfill the industrial expectation in the field of software development. Our discussion in the paper elaborates on a teacher-researcher's experience in delivering a junior core course through a pedagogic model re-organized from the constructivist viewpoint of PBL. The course involves the industrial transfer of software engineering practices into classroom learning, which is fostered through a teamwork atmosphere. The model incorporates a framework of course enactment, which demonstrates the importance of contextualized learning scenarios of PBL activities, which enable students' learning through group-based project work that should remain an important asset in their future careers.

The Industrial Practices in Software Development

According to a survey done by Vaughn (2001) on the software industry's expectation of new college graduates, the following twelve characteristics were repeatedly identified: the ability to work as a member of a team, a solid work ethic, ability to work under stress, professional demeanor, communication skills, ability to follow process, technical proficiency, desire for continuous learning, project management skills, customer focus, appreciation for the dynamics of requirements, and good citizenship qualities. Tellingly, the industrial practices that have to be accommodated in any course of software engineering education, to help students acquire such qualities expected in the software industry, are many. The following items selected by Vaughn (2001) represent some typical examples: process focus, team dynamics, planning, performance evaluation, customer relationship management, delivery on time, accountability for time and billing, and product rollout.

The Focus on Process

One of the important milestones in the journey of software professionals is the encounter of the pragmatics of a process. A process characterizes the activities of people and tools, as well as the rules for organizing those activities. The software engineering community in particular has been a leader in developing new process technologies and practices for use by large, diverse organizations. These range from process improvements strategies, such as the Software Engineering Institute's Capability Maturity Model (CMM), to notations for describing software engineering processes, to tools and environments for automating software process execution, to techniques for collecting and analyzing process data. One of the lessons learned (Fuggetta &

Wolf, 1996) from experience with software process technologies and practices, is that they can be more broadly applied to other domains in which process is an important concern. For instance, software processes are currently applied to organizational contexts where there is a need to describe the complex interactions among humans, and between humans and tools. It is true that the average student has little experience in accomplishing process steps through lectures alone. Within the PBL context, it is possible for students to adopt a process for their development exercise and the team members need to learn keeping the team focused on it during the product development.

The Dynamics on Team

Working in teams is almost a fact of life in the software industry today. Most professional programming projects require individuals to be organized as members of a team or sometimes, of more than one teams. This has been the practice in contemporary software development. Still, one important factor of teamwork is whether the team has jelled. This refers to when the members of the team work so well together, they are more than the sum of their parts. In other words, their output is greater than they could have achieved individually. In practice, whether a team jells is often a function of individual attitudes, especially of egos. It is true that software developers need egos. They probably need a big one to maintain an attitude of perfection while solving difficult problems. Yet, they must find a way to submerge their egos in forming a team or the team will never jell. The dynamics of teams relative to egos is sometimes referred to as the "Bozo Effect" (Bozo was a clown character) (Tomayko & Hazzan, 2004, pp.50-51). People who disrupt teams are often called "Bozos." The Bozo effect is a function of how many Bozos can be part of the team and still have it deliver reasonably working software.

The Essence on Planning

The schedules created for a development project can make or break the project, the product, and the people involved. The attention and forethought applied to this critical activity of planning can mean the difference between: management in control and management in pain; on-time (or early) delivery and late (or no) delivery; a full-function product and a limited-function product; high employee morale (productivity) and low employee morale (productivity); customer satisfaction and customer dissatisfaction; product success and product failure. Oftentimes, the single most important plan of a project is the project schedule plan (Whitten, 1995, pp. 89-94). It defines the roadmap of activities that affect virtually every member of a project and is the keystone for communications across a project. If the schedules defined by the project schedule plan are unreasonable, then the expected progress on the project soon will become blocked. This blockage will cause project challenges to emerge, which must then be met to deal with the obstructions. It is important to realize that the failure of a major activity to be completed on schedule eventually will impact the schedules of subsequent project activities. The domino effect could continue until the project topples.

The Evaluation on Performance

In the software industry, personal contributions to the organization are kept track of, and recognized normally through an annual or semi-annual evaluation. In particular, one's individual performance is often evaluated in the context of peer evaluations, management observation, and team contribution to success. Peer evaluations are a powerful tool in determining one's contribution to the success of the organization and support to those with whom one works. In the PBL model of course enactment (Barrows, 1985, 1988; Vat, 2000, 2001, 2004f), this industrial practice of performance evaluation can be modeled by creating a peer evaluation practice which allows for the rating of each member of a PBL team by the other members on a simple 1-to-5 or 1-to-10 scale. It is also advisable to ask the rater to rate his or her own self alongside each of the

other team members. Typically, at least three evaluations are performed during a semester in order to obtain a valid sampling of true contributions. The idea of performance evaluation could also be exercised as a form of feedback mechanism. Early feedback is useful to a team member in his or her desire to succeed and affords the whole team to correct problems perceived internally among team members.

The Importance of Customer Relationships

Today, commercial and IT organizations alike realize that to be competitive and to achieve the goals of their businesses, they must drive an understanding of their customer right into the center of their development processes: how customers work, how they buy, and what they will be doing in the future. They are aware that the future success of their businesses necessitates a commitment to understanding the customers and understanding their businesses. Thereby, relationships with one's customer and the ability to communicate openly and frankly is as important to project success as any element of a software lifecycle. Yet, this is a difficult lesson to teach in a classroom environment through lectures. Engineers used to making what they are interested in often feel constrained by having to think about what the customer wants. Their design conversations barely touch on how to match the structure of user work to the structure of the system. Yet, today's challenge is in front-end design – the idea that the voice of the customer must be heard before developers start to build.

The Delivery on Time

Every software project has a date for delivery. It marks the end of a project with the completion of a number of important milestones each of which has its own target delivery date. The important thing to remember is that every team slip on a milestone delivery compresses the available delivery time for subsequent deliverables. To avoid any slip, each team needs to perform periodic reviews on the work in progress, be it done through cooperating or collaborating. The former means different portions of the work done by different members before integration occurs, whereas the latter means the same portion of work being jointly performed by two or three members. There is a need to identify the major problems that haunt the project, and work out the possible solutions as soon as necessary; thus, project tracking in order to stay in control becomes an important issue in project development.

The Rollout of Product

Once the hard and fast due date for delivery of product to the customer is set. It is incumbent upon each project team to establish a meeting with the customer on or prior to this date to present a final product demonstration and provide all necessary documentation. A formal briefing is expected during which the product is presented, features are demonstrated, requirements not incorporated are addressed, documentation is provided, and a maintenance plan is discussed. The customer is the focus of this meeting and is expected to formally accept the project. The importance of this meeting cannot be overemphasized. It brings the project to closure and provides a certain amount of jobs stress on the team in that the delivery must occur on time, be professionally conducted, be complete, and is an important record for performance evaluation.

Empowering Students in Software Engineering Education

In order to enable our students to get familiar with the industrial practices in software development, we have tried adapting the problem-based learning (PBL) experience into different scenarios of real-world problem solving (Vat, 2004b, 2005a, 2005b). These learning scenarios of PBL activities are designed to allow students to experience the team-based process of software devel-

opment in a suitable course whose context is flexible enough to accommodate the educational formalism and the practical innovations involved. It could be considered as the industrial transfer of the software engineering experience to the classroom environment. The key lies in the appropriate design of the learning scenarios to accommodate this experimental transfer. In the following discussion, the ongoing curriculum design of SFTW300 Software Psychology is delineated, in order to discuss how the industrial practices could be incorporated.

The Course Context

The teaching of *Software Psychology*, or more properly renamed as human-computer interactions (HCI) (Vat, 2000, 2001) in the undergraduate curriculum has always been a challenge as it is composed of such a mix of elements as human factors, user expectations, man-machine interfaces construction, cognitive psychology, computer science, and those latest developments on contextual design in interactive systems. In the case of the author's teaching experience, since 1998, the pedagogy adopted to deliver such a course has been shifted from a conventional instructivist approach to the constructivist method of problem-based learning (PBL) (Albanese & Mitchell 1993; Engel 1991; Greening, 2000; Ryan, 1993). Besides, with the increasingly accumulated course material to cover in a single semester, the idea of scenario-based design (Carroll, 2000) has also been incorporated in 2000 with an attempt to help undergraduate Software Engineering students deepen the idea that HCI is concerned with understanding, designing, evaluating and implementing interactive computer systems to match the needs of people. It is the author's experience that the constructivist's ideas of problem-based learning (PBL) (Barrows, 1986) revolving around a focal problem, group work, feedback, skill development and iterative reporting, with the instructor playing the coach by the side, guiding, probing, and supporting student-groups' initiatives along the way, could help students develop a unified team-based approach to better manage the underlying software requirements. Methodically, we need some working scenarios to try out some iterative process involving researchers (instructor) and practitioners (students) acting together on a particular cycle of development activities, including problem diagnosis, action intervention, and reflective learning. Particularly, the action research approach should involve evaluating how well the students playing the role of practitioners, could function as self-directed work teams (SDWTs) (Fisher, 2000) of software professionals, following the constructivist's tenets of PBL, in performing group-based software development for a specific user scenario. Against this backdrop, the use of soft systems methodology (SSM) (Checkland & Holwell, 1998; Checkland & Scholes, 1999; Vat, 2006) has demonstrated quite a promise in enhancing the student-practitioners' learning to deal with the design difficulties typified in the complex domain of ill-defined problem situations.

The Course Goals

Software Psychology is offered annually in the fall semester as a compulsory subject for Software Engineering majors affiliated with the Department of Computer and Information Science, at the Faculty of Science and Technology. The goals of the class used to include the following: (1) to help students become HCI-literate by developing fundamental understanding of HCI in relation to human factors, usability engineering, cognitive psychology, and computer science; (2) to encourage students to formulate and express their views on user interface design of interactive systems, through project development, written work, oral presentations and classroom discussions; and (3) to raise students' awareness of the HCI impact on computer industry, and the wide-spread focus of HCI from human factors, to usability engineering, to user-centered design, in constructing systems that support human activity. Besides, in view of incorporating more industrial practices into students' learning, different scenarios of team-based project development based on the PBL cycle of activities (Vat, 2000, 2001), have to be enhanced.

The Course Design of Group-work Activities

In each semester when *Software Psychology* is offered, students embark on the PBL cycle of learning through organized groups of 4-6 members (one being the team leader). Each PBL group will be given a project scenario to explore within a specified period of time. Individual PBL groups are each assigned a project client from other teams. Namely, each team, acting as developers, is to complete an interactive system design and prototype for another team. Yet, the same team is the client of another group, responsible for clarifying the project, and resolving ambiguities as they arise, typically through a liaison member in the developer's team. Meanwhile, each PBL team is required to present their work in progress, and lead class forums to elicit students' discussions. The team leader, equivalent to project leader, has to coordinate the team activities, and ensure effective team communications. And the team members have to help set the project goals, accomplish tasks assigned, meet deadlines, attend team meetings and participate in editing project documents to be combined as the final project report. At the end of the project, each member of the respective PBL teams is required to make a presentation of his or her project involvement, with a question and answer session for the whole class. The instructor, being the project sponsors for all client teams and project supervisors for all developer teams, design the necessary scenarios to guide, motivate and provide feedback to the teams. Also, the instructor has to evaluate how well students perform in the PBL groups, and how well such groups behave as self-directed work teams in managing software requirements, and provide the necessary adjustments.

The Course Scenarios for HCI

The coursework of *Software Psychology* is based on designing learning scenarios of PBL activities in group-based project work. According to Carroll (2000), scenarios are useful in coordinating the central task of system development which includes understanding people's needs, envisioning new activities and technologies, designing effective software and drawing general lessons from systems as they are developed and used. In particular, scenarios evoke task-oriented reflection in design work. They make human activity the starting point and the standard for design work. They help designers identify and develop correct problem requirements, seeing their work as artifacts-in-use, and bearing in mind the external constraints in the design process. Moreover, scenarios help designers analyze the varied possibilities through many alternative views of usage situations. It is believed that scenario-based design in HCI could be considered as a framework approach to manage the flow of design activity and information in a rubric of task-oriented abstractions. It draws on the incrementally accrued knowledge and experience of the designers, who, in our course context, happen to be our PBL students. Essentially, our learning scenario begins when the instructor, acting as the project sponsor, invites our PBL teams to embark on a journey to develop quality software that meets customers' real needs in Web-based development. Our teams need to encounter different roles (users, stakeholders, developers) and understand the diverse languages spoken in this journey. The general idea is to construct a dynamic class Web site where individual PBL groups have to participate in sharing information by maintaining their individual Web presence. Each PBL group has two roles to play: a client to express its own Web site requirements, and a developer to fulfill other group's Web site requirements. The class Web site is to be built by a joint effort of the PBL groups. It is designed that the first thing our PBL teams have to learn is a systematic approach to eliciting, organizing, and documenting the requirements of the system to be built (Leffingwell & Widrig, 2000). Also important is a process that establishes and maintains agreement between the customer and the project team on the changing requirements of the system. Individual PBL teams have to understand users' problems in their culture and their language and to build systems that meet their needs. They need to identify the services the system provides to fulfill users' needs, through different methods of software development.

The Course Review Web Sites for HCI

A very important tool for individual PBL team is the review Web site to keep clients apprised of the project progress and to keep all team members up-to-date on all aspects of the site. It is also where the PBL team will work collaboratively on the site. Through the review Web sites, our PBL teams can conduct remote reviews with their clients or stakeholders, who can view the site in progress, give feedback on a design, get in touch with the PBL team, and check the project schedule. The review Web site of each PBL team contains such information as the roles and responsibilities of the project team, contact information for all team members, the project mission, the vision document, site architecture, schematics, all design reviews, the project schedule, and a link to the staging site (site where the work-in-progress system can be demonstrated). The vision document defines the objective of the project along with a description of the audience and the key insight into their mindset. It is what the Web team uses to design and build the site. A site-architecture is a diagram showing how the Web pages link to one another, giving an overall view of the entire site's content. Schematics are associated with individual Web pages and they each show what elements of content live on each page. Overall, we have five to six Web sites for five to six PBL groups' review, plus one extra site for the whole class, created jointly by all the PBL groups through the efforts of the liaison members from each of the project groups.

The Course Assessment for Project Work

There are a number of milestones set for project teams throughout the semester. Typically, there will be a milestone for all client teams to present their systems of interest, followed by the milestones for all developer teams to fulfill the system design, prototyping, and final delivery. At the completion of each milestone, each PBL team will be assessed according to their performance, in terms of the necessary deliverables produced, and the presentation made by the whole team. Records of the team's work should also be available from the team's review Web site for evaluation purpose. There will be a group grade and an individual grade for each member of the team. The group grade is the same for all members, but the individual grade is different. The group grade is given by the instructor and by the whole class, except for the group being evaluated. The individual grade is given through peer evaluation among the team members. Typically, a peer evaluation form is created by the group, which is used by each member of the team to rate every other member in the same team. The rating is often divided into three aspects: qualitative comments of the member's work throughout the milestone, the ranking of the member among the group including the evaluator-member, using the scale of 1-to-5 if there are five members (5: highest performance; 1: lowest performance), and the bonus distribution among all the members, of a specific amount, say, how much each member gets allocated out of 1000 dollars of bonus. In the specific instance of client-developer pair, each developer-team should also be evaluated by the client-team using a more detailed format because of the direct relationship between the two PBL teams.

The Course Enactment for Problem-Based Learning

In a semester of about 15 weeks, spanning three and a half month, it is designed that a total of four milestones are planned for students' project work. Since *Software Psychology* is offered once a year in the fall of each calendar year, the general layout of the course activities, developed around the theme that HCI is concerned with understanding, designing, evaluating, and implementing interactive computer applications, are as follows:

September

PBL teams formation – an even number is needed to facilitate client-developer pairing;
Client PBL teams each have about three weeks to conceive a system of interest;
Milestone #1: Each client team is to present its idea on system of interest;

Evaluation done for each PBL client team by the remaining teams and the instructor;
Internal evaluation performed within each client team.

October

Developer teams paired with their respective client teams;
No two teams become the client and developer of each other;
Client-developer relationship building, with client's requirements gathered for developer's elaboration;
Milestone #2: Developer's presentation of systems to be developed, with client's yes to prototype.
Evaluation done for each PBL developer team by the remaining teams (based on presentation); by the client team (based on presentation, design documents and other developer-initiated contacts and follow-ups); and by the instructor
Internal evaluation performed by each PBL developer team, (based on project management, overall work done, and personal contributions).

November

Ongoing client-developer's requirements workshops for clarification with skeletal prototypes;
Memorandum developed to accommodate changes to be incorporated and not yet completed;
Milestone #3: Developer's presentation of system prototypes with client's consent to change or continue;
Evaluation done for each PBL developer team by remaining teams, by client team; and by instructor;
Internal evaluation performed by each PBL developer team.

December

Client-developer meeting to prepare for system rollout (final prototype);
Negotiation leading to agreement on project delay or closure;
Milestone #4: Developer's final presentation of system to be delivered with client's acceptance.
Evaluation done for each PBL developer team by remaining teams, by client team; and by instructor;
Internal evaluation performed by each PBL developer team.

The Course Time Investment from Instructor and Students

September

12 hours of upfront lecture-discussion sessions to get set for group-based project work in October-November-December

October

6 hours of group-based presentation of client problems (one hour per group and six groups altogether);
6 hours of individual group meetings for feedback on client problems (one hour per group);
6 hours of individual group meeting for follow-up on developer's work in gathering client's requirements for first client-developer meeting (one hour per group);
6 hours of group-based presentation of developers' first feedback meeting with clients (first formal client-developer one-hour meeting with signing of working agreement);

6 hours of feedback lectures conducted over four sessions each of 1.5 hour

November

9 hours of individual group meetings for feedback on developer's first formal session with client (1.5 hours per group);
9 hours of individual group meetings for follow-up on developer's work in progress to prepare client's first prototype (1.5 hours per group);
9 hours of group-based presentation of developer's first prototype of client's system (1.5 hours per group);
3 hours of feedback lectures conducted over three sessions each of 1 hour

December

9 hours of individual group meetings for feedback on developer's first prototype of client's system (1.5 hours per group)
9 hours of individual group meeting for follow-up on developer's work in progress to prepare client's second prototype (final delivery) (1.5 hours per group);
12 hours of group-based presentation of developer's second prototype of client's system plus client's onsite feedback (2 hours per developer-client pair);
9 hours of individual group meetings for feedback to developer and semester-score wrap-up session (1.5 hours per group).

Lessons Elaborated for Group-Based Project Work

Throughout the years of helping students in SFTW300 to get used to group-based project work, especially those related to prototyping for collaborative Web development, the following topics of interest have become the core lessons to be elaborated in every semester of *Software Psychology*.

The Project

Defining a project means understanding and communicating on paper the project's mission, objectives, risks, and requirements. Typically, writing a project mission statement is the first job to handle, which is decomposable to three essential tasks: identify the project's objectives; identify the users; and identify the project scope. Put it briefly, a mission statement describes the solution to the problem that the project is going to solve. There are three questions to answer: what are we going to do? For whom are we doing it? How do we go about it?

● Identify Objectives

Identifying objectives for a specific project is to define some attainable desired outcome at the end of the project. The important question is to establish criteria for objectives so that we know when we achieve the desired outcome. In particular, objectives for a Web project help the creative team to design the best graphics (look and feel) for the site, they help the programmers understand how the Web site might grow, and they tell the writers what tone or information is critical to the site.

● Identify Targeted Users

Identifying targeted users for a specific project, especially a Web project, is always important because the kinds of content that will be created for the Web site are often determined by such users whom we want to visit the site (Gause & Weinberg, 1989). The issue here is how we go about learning what the targeted users want to see on the Web site. The obvious answer is to ask them directly. There are many ways to gather user data (or requirements) such as interviewing,

holding requirements workshop, and inviting users to join the development team as a liaison for ongoing changes.

- **Identify Project Scope**

Identifying project scope at the beginning of the project, with proper supporting documentation and client approval, is always important. In order to nail down the scope, the project team must determine the key elements required in the project, which are often extracted from the objectives document of the project, namely, the document that states the project objectives. Starting from the project objectives, the team will need to investigate what functional and non-functional features the Web site must have to satisfy the objectives. Examples of functional features include providing information systems (IS) support for online learning, and allowing links to use broadcast services to a number of group members. Examples of non-functional features include site design, navigation architecture, copywriting, and quality assurance testing. This process often called developing the work breakdown structure (WBS) is to break down each element into tasks, analyzing the tasks and the time it will take to complete them and making some decisions about how to implement the features so as to stay on schedule. The culmination of this effort is to produce the scope document, which reiterates the mission and objectives of the Web project including a sign-off agreement for the developer team and the client to put in writing the stated features (functional and non-functional) of the project (Web site).

The Project Team

The project team is the team that will fulfill the project objectives. This team is often composed of some core members with specific roles and responsibilities. Example roles include the project coordinator, the project tracker, the team secretary, the team liaison, and other technical members such as, in the context of a Web project, the technical lead, the creative lead, the information architect, and other support roles such as the interface designers, and the Web production specialists.

- **Identify Roles and Responsibilities**

It is important to indicate the responsibilities of specific roles in a team of members in order to fulfill the project objectives, such as planning, designing, prototyping a Web site. The following describes the roles and responsibilities of team members in the SFTW300 course scenarios.
The Project Coordinator. This role is responsible for seeing and communicating the big picture to the team and the client. Within the team, the coordinator needs to work with the team members to accomplish such tasks as scoping the project work, developing the project plan, scheduling, and allocating resources, budgeting, as well as building up the team. Without the team, the coordinator needs to handle client management, keeping the client in touch with project development, and keeping track of client's emergent requirements clarification.

The Project Tracker. This role is responsible for keeping track of the work designated among the team members during a specific period (say, in between two team meetings, or two project milestones) and overseeing that they have been accomplished timely, and with the quality expected. The idea is to ensure that any work product produced should meet the criteria specified in the scope document, and be produced on schedule. The tracker also keeps a good record of the performance of individual team members for evaluation purpose.

The Team Secretary. This role is responsible for the administrative details of project work, such as scheduling meetings for team members, collecting agenda items from team members before each meeting, keeping records of meetings minutes, issuing summary of actions items to be followed up at next meeting. The secretary is keeping a good record of rationale management (Dutoit & Paech, 2001) throughout the project. Essentially, rationale includes such details at the

end of each meeting as the issues that were addressed, the alternatives that were considered, the decisions that were made to resolve the issues, the criteria that were used to guide decisions, and the debate team members went through to reach a decision.

The Team Liaison. This role is to assist the project coordinator in communicating with the client. In the SFTW300 design of inter-team communications, however, since each team is to play both the client and the developer roles, the role of the liaison becomes important in communicating with the client team, the ongoing development of their project. Meanwhile, the same liaison in each team could also serve as the spokes-person of his or her team in communicating with the developer team, the team's own project requirements. It is designed that two teams could not be the client and developer of each other.

The Technical Members. In the context of Web project, the technical members include the following roles: technical lead, creative lead, information architect, and other support staff. The technical lead oversees the project from a technical point of view. He or she assists the project coordinator in ensuring that the technical strategy is sound, and manages other technical staff such as programmers (Web or database) and systems integrators. The creative lead determines the creative concept for the Web site and is responsible for the site's design. He or she interacting with the technical lead to determine what is technically possible, often reports status to the project coordinator. The information architect designs a usable and useful interface that facilitates how the user will interact with the site and find the information they need. He or she is responsible for site architecture, navigation, search and data retrieval, as well as interaction design, such as messages to users regarding errors, service, and technical needs of the site. Other support staff includes the graphic designers and Web production specialists. The former create the look and feel of the Web site, transforming the information design into a visual design, whereas the latter transforms the artwork that the graphic designers create into Web-ready art. In practice, the Web production specialist is the person who codes the HTML pages, integrates Java or Shockwave applications, integrates images and animations, and hands the project off to the Project Tracker for quality assurance.

● Build up a Team

It certainly takes time and discipline to transform a PBL team of student members to a self-directed work team of promising software professionals. In the short span of each SFTW300's semester of about three and a half months, there are many soft skills a PBL team needs to acquire. The following serves as a useful set of selected concerns I find to be particularly important in the team buildup process.

Process Focus. The average student has little background in actually accomplishing process steps over a period of time and more often no background in doing so in a team with the added requirements of several milestones of prototyping and an inflexible delivery date, as well as a client who is generally not process oriented. During the early stages of the class, each PBL team is exposed to some typical software development processes, such as the spiral process, the unified process, and the extreme programming process. The students are asked to adopt a process for their development exercise and the team coordinator is required to attempt to keep the team focused on it during prototyping for the milestones. Students' feedback indicates that the pressure of the delivery schedule, client involvement, and prototypes development, has taught them that the chosen process has to be flexible enough to accept change but the balance between consistent application of a process and responsiveness to the client is difficult to maintain and this is impossible to learn through lectures alone. The client experience during the semester demonstrates to each team the difficulty one encounters after graduation and the lessons learned through the project tend to remain with them far longer than their readings and individual examinations.

Team Dynamics. Students embarking on SFTW300 have had one semester's PBL style of collaboration in SFTW241 Programming Language Architectures (I). However, the grouping arrangements of SFTW300 require each newly formed PBL group to discover whom they can rely on, capitalize on individual skill sets, and find a way to work together. Students still tend to be mistrusting at first and very comfortable with individual efforts but leery of having to rely on others. The early stages of class become essential occasion to conduct what will be the first of many activities that promote positive group interactions. Examples of such activities include: writing a group portfolio expressing the profiles of individual team members in terms of their individual technical expertise; engaging in mental games that require the skillful use of teamwork to complete or that make a point about the distinction between person-centered and group-centered learning/working styles. This understanding becomes instrumental when different roles are being taken by members of the group: one role taken up by one member, or one role shared by two or three members, or roles are taken by members through rotating turns. The idea is to achieve coordination to get the project work done through a suitable mix of individual work, cooperating work (different tasks done by different members so as to integrate the pieces), and collaborating work (same portions of work done jointly by different members).

Planning Concerns. Throughout the semester's work, there are several essential milestones (essential due dates) that have to be met by each PBL group. Yet, the only hard and fast date that must be met is the final delivery. Deliverables required of each PBL team include a concept of operations document, a design document, a test plan, and the final prototype comprising site architecture, schematics, and navigation guide in the specific case of a Web project. In the client-developer relationship established by the instructor, each team coordinator is permitted to make arguments for extension of any deliverable due date (except for final delivery) knowing that each extension granted added additional difficulties later in the course for an on time delivery of other documents. This procedure forced each team to evaluate their scheduling philosophy and to perform an informal risk assessment for the entire project. The policy of assigning the same project grade to each member of the team is seen as a strong motivator for each student to take seriously the project activities. Following the industrial model of shared responsibility (the team fails or the team succeeds) seems to provide a far more memorable learning experience with respect to the planning process and maintaining a schedule. Rigorous discussions have often been observed over issues of planning milestones and still leaving enough time to produce the remaining deliverables with reasonable quality and timeliness.

Meeting Routines. In a project environment, nobody likes to go to a meeting that has no agenda. These kinds of meetings seem like wastes of time to team members who have deadlines to meet. Thereby, effective meetings (Doyle & Straus, 1982) must have an agenda sent out to each participating members, so that people know what to expect. At the meetings, PBL teams have to make sure going over the schedule and determine if milestones will be hit on time. Meanwhile, asking team members for any issues or red flags that they see is important. This will help them anticipate risks as the project moves forward. Also important is to ensure that people leave with clear action items. People should know what they are supposed to be doing and what to do next either by day or by week. The main point is to keep the meeting quick and focused, and then follow up the meeting with a summary of points that were discussed. Overall, there are many steps PBL students have to learn in leading an effective team meeting.

The Project Client

Dealing with the project client is an important lesson for the PBL groups of students in SFTW300. We have a number of skills to develop in client management and these take time. So, students have to learn them incrementally throughout the semester.

● Understanding Client's Point of View

The client is the person or organization who has contracted the developer team to complete the project. In the case of SFTW300, the client team has this request to construct a staging Web site, demonstrating the team's topics of interest in interactive system design. It is important for each PBL developer team to take time to understand the client's project position; namely, what is at stake for the client. This is actually the beginning of a requirements gathering process, characterized by such activities as interviewing, storyboarding, and workshops of requirements elicitation.

Interviewing. Interviewing is one of the most important and straightforward ways to understand the client's concerns. Yet, before an interview is conducted, the developer team needs to do some hard thinking about what they need to learn. The process often involves preparing some thoughtful questions about the nature of the user's problem without any context for a potential solution. Examples include: Who is the user? Who is the client? Are their needs different? Where else can a solution to this problem be found? Namely, we are trying to find out what the users need in order to meet the objectives set forth by the stakeholders. Oftentimes, the use of a questionnaire to help strategize the interview is also necessary. Besides, it is suggested that a tape or a video recording of the interview is arranged to help capture the details of the interview, pending the approval of the client. In the case of SFTW300, each developer team is required to video-tape the interview as an important project item of all possible development records.

Storyboarding. The idea of storyboarding is to gain an early understanding from the users on the concepts proposed for the project system. Typically, with storyboarding, the user's reaction can be observed very early in the development cycle, well before concepts are committed to code and, in many cases, even before detailed specifications are developed. In fact, when the users do not know what they want, even a poor storyboard is likely to elicit a response of "No, that is not what we meant, it's more like the following," and the game is on. Basically, a storyboard can be anything the developer team wants it to be, and the team should feel free to use its imagination to think of ways to storyboard a specific application. Yet, depending on the mode of interaction with the users, storyboards can be categorized into three types: passive, active, or interactive (Leffingwell & Widrig, 2000, p.126). *Passive storyboards* include sketches, pictures, screen shots, PowerPoint presentations, or sample outputs. In a passive storyboard, the analyst plays the role of the system and simply walks the user through the storyboard, with a "When you do this, this happens" explanation. *Active storyboards* try to make the user see "a movie that has not been produced yet." They are animated or automated, perhaps by an automatically sequencing slide presentation or an animation tool or even a video. They provide an automated description of the way the system behaves in a typical usage or operational scenario. *Interactive storyboards* let the user experience the system in as realistic a manner as practical. They require participation by the user in order to execute. They can be simulations or mock-ups or can be advanced to the point of throwaway prototype.

Requirements Workshop. The requirements workshop (Leffingwell & Widrig, 2000, pp.103-111) is perhaps the most powerful technique for eliciting requirements. It is designed to encourage consensus on the requirements of the application and to gain rapid agreement on a course of action, all in a very short time frame. With this technique, key stakeholders of the project are gathered for a short, intensive period, typically no more than one or two days. The workshop is facilitated by a team member or by an experienced outside facilitator and focuses on the creation or review of the high-level features to be delivered by the new application. A properly run requirements workshop has many benefits: It assists in building an effective team, committed to one common purpose, i.e., the success of the project. All stakeholders get their say; no one is left out. It forges an agreement between the stakeholders and the developer team as to what the application must do. It can express and resolve political issues that are interfering with project

success. The output, a preliminary system definition at the features level, is available immediately.

● Setting Clear Expectations

It is important to take time to communicate the developer team's work process to the client, including setting clear expectations and providing rationale for these expectations. In particular, we need to flesh out what the client's and the developer's expectations are for the following:

Deliverables. What will the developer deliver to the client? What does the client need to deliver to the developer in order to move forward? Who in the client's team is delivering content to the developer? The developer generally does not want to be inundated with requests from all over the client's organization. If the client is delivering assets (precious items, say, master copy), it is also important to keep a good record as to what and when the item has been delivered. It is also good to specify the format in which the developer needs to obtain such items.

Approvals. How will approvals be handled? Does the developer need a 24-hour turnaround for approval? The client's approval should always be in writing, no matter how good a relationship the developer has with the client. It is better to have a policy in place to handle approvals than to loose a client as a result of a dispute about what exactly was approved. It has been observed that good relationship between developer and client teams could end abruptly over a dispute.

Policies. Once available, policies such as the change-order process or approvals process should be jointly verified by both the client and the developer. The client should be able to ask questions about the policies. It is also essential not to make exceptions in policies for a client. Such exceptions could easily create disputes later in the developer-client relationship. It is an important policy to write a contact report each time the developer has a conversation with the client in which aspects of the project have been discussed, such as issues raised, and changes negotiated. A copy of this contact report should be sent to the client. It is a good way to document the project and to keep track of things being discussed informally with your client.

● Communicating throughout the Project

As the relationship between client and developer develops, it is important that the developer remains consistent in the approach to communication so that the client's trust of the developer should become strengthened. There are often several issues to watch for in order to keep communications clear between the developer and the client.

Assumptions. PBL students are reminded that every project has assumptions. All the deliverables that the developer determines in the scope document are made with certain assumptions. It is essential to give the client the developer's assumptions early on, when the scope document is delivered. Students are advised to take time to review the document, going over each assumption carefully to make sure the client understands what the developer expects. When the client and the developer both have their own deadlines to meet, they can truly become partners of the same team, and the client could see firsthand how their participation can affect the success of the project.

Scope and Approval Document. The questions of scope, cost, and what can be done by the deadline are ongoing issues in project work. In most Web project, the developer determines the scope of a project after a discovery phase in which both the client and the developer discuss the project's objectives and requirements. Once the requirements are determined, the developer can enter the design phase with a sense of the project scope. However, the client may not be on the same page as the developer. Experience indicates that most clients can come away with very different expectations. That is why a full explanation of scope, with the time and cost considerations clearly defined, say in the approval document, is critical to getting the client on the same page as

the developer moves forward with the project. The approvals document is a document in which the client signs off on features that make up the project. In the case of a Web site, line items in the document might be navigation, visual design, forms, data schema, network configuration, site architecture, and any relevant pseudo-code. It is important to get sign-offs after each design milestone is hit.

Effective Reviews. Conducting effective reviews is a skill PBL students need to acquire through experience. Before meeting with the client, it is important to perform an internal review first, by writing down the essential points to hit, and by anticipating the client's reactions. Any point raised by the client must be responded by going back to the logic behind the decision, and the logic must also be traceable to an objective of the project. In case the client dislikes anything, the strategy is: Always be prepared to offer an alternative, and never to argue with the client. If the client approves a concept, the next step must be to get the sign-off on the design. Preparing a contact report carrying fields for signatures can help.

Enacting SSM as a Scenario-Based Learning Process

If HCI-based project development of software support is concerned with understanding, designing, evaluating and implementing interactive systems (IS) to match the needs of people (Vat, 2004a, 2004b, 2004c), a very important activity for each developer PBL team is to actively engage its client PBL team (Curtis, Krasner, & Iscoe, 1988) in ensuring the quality and timeliness of the software outcomes. Undeniably, setting up interactive IS support is a social act in itself, requiring some kind of concerted action by both the developer and the client teams (Conklin & Burgess-Yakemovic, 1991); and the operation of an IS entails such human phenomena as attributing meaning to and making judgments about what constitutes a relevant category. It is convinced (Vat, 2005a, 2005b, 2005c) that through maintaining a continuous focus on situations of and consequences for human work and activities, IS developers could become more informed of the problem domains, seeing usage situations from different perspectives, and managing trade-offs to reach usable and effective design outcomes. However, getting users' work right involves capturing and accommodating users' emergent (or subject-to-change) analytical activities, which are open-ended yet integrated and opportunistic yet coherent. IS developers must understand the intertwined regularities and idiosyncrasies of human activities and create software that support the right moves and degrees of agility at the right times and places and for specific purposes. In this regard, the problem of designing IS support for any human-centered knowledge work should never be thought of as something to be defined once and for all, and then implemented. Instead, it must be based on the observation that all real-world organizational problem situations contain people interested in trying to take purposeful action (Checkland, 1981; 1999).

And the idea of a set of activities linked together so that the whole, as an entity called the human activity systems (HAS) from the viewpoint of Soft Systems Methodology (SSM) (Checkland & Holwell, 1998; Checkland & Scholes, 1999; Wilson, 2001) could prove useful. In fact, as far as pursuing a purpose is concerned, a HAS model could be considered as a representative organizational scenario for exploring any situation-specific IS support, which is never fixed once and for all. Still, the important point is that we must be conscious of the fact that any scenario of HAS (human activity systems) models created are merely abstract logical machines for pursuing a purpose (Checkland, 1979, 1984), defined in terms of declared worldviews, which can generate insightful debate (or rational discussion) when set against actual purposeful action in the real-life situation. The implicit belief behind constructing the HAS models is that social reality – what counts as facts about the social world inside an organization, say the client PBL teams of students – is the ever changing outcome of a social process in which human beings (individual client team members) continually negotiate and re-negotiate, and so construct with others (members of the developer team) their perceptions and interpretations of the world outside themselves (expected

IS support for specific activities), and the dynamic rules for coping with it (cooperating or manipulating). In the process, the context of IS development actually becomes an organized discovery of how human agents (both the client and developer team members) make sense of their perceived worlds, and how those perceptions change over time and differ from one person or group to another.

Nevertheless, the basic shape of the scenario-based learning approach for the developer PBL team could be described as follows: Find out about the problem situation that has provoked concern; Select relevant concepts that may be integrated into different human activity systems; Create HAS models from the relevant accounts of purposeful activity; Use the models to question the real-world situation in a comparison phase. The debate initiated by the comparison normally entails the findings of accommodations between conflicting interests, that is to say, situations that may not satisfy everyone, but could still be lived with, enabling action to be taken. Oftentimes, the purpose of the debate is to collectively learn a way to possible changes (improvements) to the problem situations, by activating in the people involved (client PBL team), a learning cycle, which counts on their ability to articulate problems, to engage in collaboration, to appreciate multiple perspectives, to evaluate and to actively use their knowledge. It is worthwhile to notice that taking the purposeful action would itself change the situation, so that the whole cycle could begin again, and is in principle never ending. Likewise, through scenarios of HAS models, IS architects could provide help in articulating the requirements of specific technical support through operating the learning cycle from meanings to intentions to purposeful action among the specific group of organizational members.

Re-conceptualizing Teaching and Learning through PBL

Today, there is a growing tendency away from a conventional transmissive pedagogy in higher education, towards a pedagogy that can broadly be characterized as constructivist (Booth, 2001). The transmissive pedagogy refers to teaching based on an assumption that students receive information from the teacher and slot it straight into an empty place in their knowledge base, or at best, work on it later to make it their own. The constructivist pedagogy renders an approach to learning through a variety of knowledge building processes, and that teaching should encourage students to work actively towards understanding within a framework of personal responsibility and institutional freedom. Within the culture of transmissive teaching, what constitutes good learning has largely been based on success in examinations designed to test the quantity and the quality of what individual students have learned, in the sense of giving back, in an appropriate form, that which the teachers taught and the text-books told. The constructivist shift brings new dimensions to the notion of good learning, such as being able to find information and knowledge by oneself; of being able to look critically at what one finds; of being able to question one's teachers; of being able to collaborate with colleagues; and of being able to discuss what one knows with one's peers and with the public. Accordingly, as the need to look at the student's work as a whole is increasingly emphasized, the notion of good teaching shifts away from the role of presenter and towards the much more complex role of a guide and a coach. In this regard, problem-based learning (PBL) addresses these issues and offers an attractive alternative to traditional education by shifting the focus of education from what faculty teaches to what students learn. Content remains important, but emphasis shifts more to the process. Indeed, Greening (1998, 2000) describes PBL as a vehicle for encouraging student ownership of the learning environment. There is an emphasis on contextualization of the learning scenario, providing a basis for later transference, and learning is accomplished by reflection as an important meta-cognitive exercise. Besides, the execution of PBL, often done via group-based project work, reflects the constructivist focus on the value of negotiated meaning. More importantly, PBL being not confined by discipline boundaries encourages an integrative approach to learning, which is based on requirements of the problem as perceived by the learners themselves. Consequently, my re-

conceptualized view of teaching and learning identifies with a learner-centered perspective of PBL based on the idea of collaborative learning, where it is necessary to clarify the new roles of the teachers and the students, and the renewed pedagogical organization consistent with the philosophy of a collaborative learning environment in support of PBL.

● A New Role of the Teacher

Instead of performing as the sage on the stage transmitting knowledge to a class of innocent students, in the collaborative learning environment of PBL, teachers' roles are often defined in terms of mediating learning through dialogue and collaboration where knowledge is created in the community rather than being transferred from the individual. More specifically, the idea of mediating could include such aspects of facilitating, modeling, and coaching (Chickering & Gamson, 1987; Chung, 1991; Mayer, 1988). Facilitating involves creating rich activities for linking new information to prior knowledge, providing opportunities for cooperative work and collective problem solving, and offering students a multiplicity of authentic learning tasks. Modeling serves to share with students not only the perceived content to be learned, but also the important meta-cognitive skills of higher-order thinking, in the process of communication and collaboration. Coaching involves giving hints or cues, providing feedback, redirecting students' efforts, and helping them use a strategy. A major principle of coaching is to provide help only when students need it so that students retain as much responsibility as possible for their own learning. In fact, we need to teach students to rely less on teachers as the source of knowledge. We need to help them learn to learn as self-directed groups of active, autonomous, and responsible individuals. One of the specific goals in the PBL setting is to have students rely more heavily upon their classmates for assistance in doing a task and in evaluating a possible solution. Only after they have checked with everyone in the group should they ask their teacher for help. Operationally, it is the teacher's job to specify the instructional objectives, usually in discussion with the learning (PBL) groups; explain the cooperative goal structure; observe students' interactions in terms of the learning process, and social relationships within the group; feedback on the group-based evaluation of the learning products; and also maximize social interaction among groups through suitable design of inter-group interacting patterns, to create the expected community of learners among the students in the class.

● A New Role of the Students

In collaborative learning settings, students are expected to assume their new roles as collaborators and active participants. It may be useful to think how these new roles influence processes and activities before, during, and after learning. For example, before learning, students set goals and plan learning tasks. During learning, they work together to accomplish tasks and monitor their progress. And, after learning, they assess their performance and plan for future learning. In practice, students constantly need help from the teachers to help them fulfill such new roles. Specifically, students need to learn to share, rather than compete for, recognition, and evaluate the learning outcomes rather than hurry to finish the task. It is important to nurture a group-based atmosphere for comfortable trial and error as well as for asking questions and expressing opinions. Students must learn to become teachers of their own, and the group-based interaction should serve as the incubator for co-development of ideas. Indeed, a frequent formula (Dilworth, 1998) that action learning proposes has been quite useful in constantly reminding students of their new role in collaborative learning. Namely, L = P + Q + R, where L (learning) equals P (programmed instruction) plus Q (questioning) plus R (reflection). Here P represents the knowledge coming through textbooks, lectures, case studies, computer-based instructions, and many others. This is an important source of learning but carries with it an embedded caution flag. That is, P is all based in the past. Q means continuously seeking fresh insight into what is not yet known. This Q helps avoid the pitfall of imperfectly constructed past knowledge. By going through the Q step first, we are able to determine whether the information available is relevant and adequate to our needs. It will

point to areas that will require the creation of new P. R simply means rethinking, taking apart, putting together, making sense of facts, and attempting to understand the problem. Following the use of this formula, action steps are planned and carried out with constant feedback and reflection as the learning takes place. In short, what this formula can provide for PBL students is elevated levels of discernment and understanding through the interweaving of action and reflection.

- **A Renewed Pedagogical Organization**

It is understood that collaborative learning will not take place with students sitting in rows facing the teacher. At least, tables or desks must be pushed together in small groups to facilitate group interaction. Materials and resources should best be made readily accessible. Indeed, the basic requirements for PBL group-based activities are often beyond the capacities of many a classroom in our university environment today. Preferably, we could have access to a multi-function hall in the library (with easy access to other resources such as computers and Internet), with enough floor area to accommodate the setup of a workspace environment with re-configurable facilities such as: a number of conference desks each of which is good for at least six people; an open area with a wall-size screen for LCD-projection, for large gathering or presentation or briefing for the whole class. The idea is that sharing and working together even in a less-than-perfect environment will be louder than in an environment where isolated students work silently from textbooks. More importantly, to make collaborative learning work, there are major tasks that the instructor should address in organizing the PBL group learning activities. First, we need to clearly specify the objectives for the lesson lest the students always guess what the professor wants: academic objectives or collaborative skills objectives. Second, we need to make decisions or devise innovative strategies about placing students in learning groups before the PBL episode begins. Third, we need to carefully explain the task, goal structure, and the learning activity. Fourth, we need to monitor the effectiveness of the PBL collaborative learning groups and intervene when necessary to re-orient the learning efforts toward pinpointed tasks, or to increase students' interpersonal and group skills. Lastly, we need to evaluate student achievement and help them discuss how well they collaborated with one another. It is also important to understand that while groups work to achieve common goals, each member should fulfill a particular role or accomplish an individual task within the group project. The teacher needs to assess both the group and the individual work.

Remarks for Continuing Challenge

The software engineering workplace of this century requires professionals who not only have an extensive store of knowledge, but who also know how to keep that knowledge up-to-date, apply it to solve problems, and function as part of a team. This view of the software industry compels educators to rethink and reinvent the ways in which software professionals are prepared. In particular, schooling must extend beyond the traditional preparatory goal of establishing a knowledge base. Concomitantly, it must actively engage our students in opportunities for knowledge seeking, for problem solving, and for the collaborating necessary for effective practice. To realize such experiences, educators have looked to constructivist pedagogical designs that are based on the assumption that learning is a product of both cognitive and social interactions in problem-centered environments (Greeno, Collins, & Resnick, 1996; Savery & Duffy, 1994). Problem-based learning (PBL) is an example of such a design (Vat, 2005d, 2006). Today, what has become known as a classic version of PBL is described by Barrows (1985, 1988), in the educational context of medical students (Spaulding, 1991). This model has two key features: a rich problem is involved to afford free inquiry by students, and learning is student-centered. In this approach, a group of five to seven medical students and a facilitator meet to discuss a problem (Barrows, 1986). The facilitator provides the students with a small amount of information about a patient's case, and then the group's task is to evaluate and define different aspects of the problem and to gain insight into the underlying causes of the disease process. This is accomplished by extracting

key information from the case, generating and evaluating hypotheses, and formulating learning issues. Learning issues are topics that the group deems relevant and in need of further explication. The group members divide up the learning issues among themselves and research them. They then share their information and use it to explain the patient's disease process. At the completion of the cycle, the students reflect on what they learned from the problem. The facilitator's role is to help the students' learning processes by modeling hypothesis-driven reasoning for the students and by encouraging them to be reflective. In fact, both cognitive constructivist and socio-cultural theories provide insights into the learning mechanisms of PBL (Greeno, Collins, & Resnick, 1996). In terms of individual learning, PBL situates learning within the context of professional (medical) practice. Problems give rise to epistemic curiosity (Schmidt, 1993) that will, in turn, trigger the cognitive processes of accessing prior knowledge, establishing a problem space, searching for new information, and reconstructing information into knowledge that both fits into and shapes new mental models. Meanwhile, proceeding through the PBL process requires the learner's meta-cognitive awareness of the efficacy of the process. In this regard, PBL is inherently self-directed. However, PBL does not exist in a vacuum. Rather, it is a social system within a larger cultural context. The knowledge that the learner seeks is embedded in and derives from social sources – in Barrows' case, the world of medical practice, and in our discussion, the community of software practitioners. From this perspective, the learner is seen as both transforming and being transformed as the processes of practice and their underlying symbol systems are internalized through dialectical activity (John-Steiner & Mahn, 1996). In this sense, learning is not an accumulation of information, but a transformation of the individual who is moving toward full membership in the professional community. This identity-making is marked by observing the facility with which cultural tools, or the ways of thinking and using language, are invoked, especially in the socio-cultural context of PBL group meetings, which stimulate the social process of professional problem solving often done through the scaffolding support of PBL students themselves.

To conclude this chapter, I hereby render some of my perspectives behind adopting problem-based learning (PBL) in teaching industrial practices of software engineering. The educational literature warns against compartmentalized units of study that produce students who cannot integrate the different parts of their knowledge. Although a fully integrated degree was beyond our scope of discussion, many of our conventional courses had compartments that bore out the literature's predictions. Any new course designed must be as integrated as possible especially in the context of software development, if we want our students to bring all their knowledge to bear on solving real-world problems. In this regard, the nurture of independence and initiative becomes important. Indeed, our conventional courses were widely criticized for stifling students' independence and initiative. Yet, through PBL, the problems we offer are loosely specified, leaving plenty of scope for student initiative. Students should discover they can learn by themselves, using a range of resources. They are aided in learning to do this by the PBL cycle of collaboration, which develop in them their social and mega-cognitive skills. Consequently, students' critical thinking and problem-solving abilities are sharpened. These are crucial to effective project (software) development, especially at the higher levels of analysis and design. In the specific case of SFTW300, students have to go through the process of understanding, designing, implementing and evaluating interactive computer systems to match the needs of client. This is a teamwork development exercise requiring students to work in groups. This is important for their future careers because employers nowadays expect their fresh graduates to have the ability and experience to perform in group-based project work. SFTW300 supports groups by identifying specific roles for group members, providing class time and guidelines on group management, monitoring group planning and progress, and assigning marks for group management and reflection on group processes. Students working in a group naturally learn to communicate with one another, which is another goal highly valued by employers. In particular, at the end of each problem, PBL students

need to give a demonstration, during which each student must speak, and present a written report, followed by a session on question and answer. All these require the students to have good command of communications skills. Overall, PBL fosters such generic skills as group work, planning, problem-solving, independent learning, research skills, writing, and oral presentation. These are university goals and also highly valued by employers in the computing industry.

References

Albanese, M., & Mitchell, S. (1993). Problem-based learning: A review of literature on its outcomes and implementation issues. *Academic Medicine*, *68* (1), 52-81.

Barrows, H. S. (1985). *How to design a problem-based curriculum for the pre-clinical years.* New York: Springer.

Barrows, H. S. (1986). A taxonomy of problem-based learning methods. *Medical Education*, *20* (6), 481-486.

Barrows, H. S. (1988). *The tutorial process.* Springfield, Illinois: Southern Illinois University School of Medicine.

Booth, S. (2001). Learning computer science and engineering in context. *Computer Science Education*, *11* (3), 169-188.

Carroll, J. M. (2000). *Making use: Scenario-based design of human-computer interactions*. MIT Press.

Checkland, P. (1981). *Systems thinking, systems practice.* Chichester: Wiley.

Checkland, P. (1979). Techniques in soft systems practice, Part 2: Building conceptual models. *Journal of Applied Systems Analysis*, *6*, 41-49.

Checkland, P. (1999). Systems thinking. In W. L. Currie & B. Galliers (Eds.), *Rethinking management information systems.* Oxford University Press.

Checkland, P. & Holwell, S. (1998). *Information, systems, and information systems: Making sense of the field.* John Wiley and Sons: Chicheser.

Checkland, P. & Scholes, J. (1999). *Soft systems methodology in action.* Chichester: Wiley.

Checkland, P. (1984). Systems theory and information systems. In Th. M. A. Bemelmens (Ed.), *Beyond productivity: Information systems development for organizational effectiveness.* North- Holland, Amsterdam.

Chickering, A. W., & Gamson, A. (1987). Seven principles for good practice in undergraduate education. *AAHE Bulletin*, *39* (7), 3-7.

Chung, J. (1991). Collaborative learning strategies: The design of instructional environments for the emerging new school. *Educational Technology, 31* (12), 15-22.

Conklin, J. & Burgess-Yakemovic, K. C. (1991). A process-oriented approach to design rationale. *Human-Computer Interaction*, *6*, 357-391.

Curtis, B., Krasner, H. & Iscoe, N. (1988). A field study of the software design process for large systems. *Communications of the ACM, 31* (11), 1268-87.

Dilworth, R. L. (1998). Action learning in a nutshell. *PIQ*, *11* (1), 28-43.

Doyle, M. & Straus, D. (1982). *How to make meetings work.* New York, NY: The Berkeley Publishing Group.

Dutoit, A.H. & Paech, B. (2001). Rationale management in software engineering. In S.K. Chang (Ed.), *Handbook of software engineering and knowledge engineering* (Volume. 1). World Scientific Publishing.

Engel, J. (1991). Not just a method but a way of learning. In D. Bould & G. Felletti (Eds.), *The challenge of problem-based learning* (pp. 21-31). New York, NY: St. Martin's Press.

Fisher, K. (2000). *Leading self-directed work teams: A guide to developing new team leadership skills.* McGrawHill.

Fuggetta, A. & Wolf, A. (Eds.) (1996). *Software process.* New York: John Wiley & Sons.

Gause, D. & Weinberg, G. (1989). *Exploring requirements: Quality before design*. Dorset House Publishing.

Greening, T. (2000). Emerging constructivist forces in computer science education: Shaping a new future? In T. Greening (Ed.), *Computer science education in the 21st century* (pp. 47-80). Springer.

Greening, T. (1998). Scaffolding for success in problem-based learning. *Medical Education Online*, *3* (4): 1-15. Retrieved from http://www.utmb.edu/meo/

Greeno, J. G., Collins, A. M. & Resnick, L. (1996). Cognition and learning. In R. G. Calfee & D. C. Berliner (Eds.), *Handbook of educational psychology* (pp. 15-46). New York: Simon & Schuster Macmillan.

John-Steiner, V. & Mahn, H. (1996). Socio-cultural approaches to learning and development: A Vygotskian framework. *Educational Psychologist, 31,* 191-206.

Kimball, L. (1995). Ten ways to make online learning groups work. *Educational Leadership, 53* (2), 54-56.

Leffingwell, D. & Widrig, D. (2000). *Managing software requirements: A unified approach.* Reading, Massachusetts: Addison Wesley.

Mayer, R. (1998). Cognitive theory for education: What teachers need to know. In N. Lambert & B. McCombs (Eds.), *How students learn: Reforming schools through learner-centered education,* (pp. 353-378). Washington, DC: American Psychological Association.

Ryan, G. (1993). Student perceptions about self-directed learning in a professional course implementing problem-based learning. *Studies in Higher Education*, *18,* 53-63.

Savery, J. R. & Duffy, T. M. (1994). Problem-based learning: An instructional model and its constructivist framework. *Educational Technology, 35* (5), 31-38.

Schmidt, H. G. (1993). Foundations of problem-based learning: Some explanatory notes. *Medical Education, 27,* 422-432.

Tomayko, J. E. & Hazzan, O. (2004). *Human aspects of software engineering.* Hingham, Massachusetts: Charles River Media.

Vat, K.H. (2000). Teaching software psychology: Expanding the perspective. In *Proceedings of the Thirsty-first SIGCSE Technical Symposium on Computer Science Education*, Austin, TX, Mar. 8-12, pp. 392-396.

Vat, K.H. (2001). Teaching HCI with scenario-based design: The constructivist's synthesis. In *Proceedings of the Sixth Annual ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE2001)*, Canterbury, U.K., Jun. 25-27 pp. 9-12.

Vat, K.H. (2002). Teaching architectural approach to quality software development through problem-based learning. In *Proceedings of the 2002 Informing Science + IT Education Conference (IsITE2002)*, in Cork, Ireland: Informing Science Institute, Jun. 19-21. Also available at http://proceedings.informingscience.org/IS2002Proceedings/papers/vat043Teach.pdf

Vat, K.H. (2003). Conceiving architectural aspects for quality software education through the constructivist perspective. In T. McGill (Ed.), *Current issues in IT education* (pp.98-116). Hershey, PA: IRM Press (Idea Group Inc.).

Vat, K.H. (2004a). Conceiving a learning organization model for sustainable development: The IS Manager's perspective based on soft systems methodology. In *Proceedings of the IEEE International Engineering Management Conference 2004 (IEMC2004)* (pp. 500-504), Oct. 18-21, Singapore.

Vat, K.H. (2004b). Conceiving scenario-based IS support for knowledge synthesis: The organization architect's design challenge in systems thinking. In *Proceedings of the 10th International Conference on Information Systems Analysis and Synthesis (ISAS2004)* (pp.101-106), Orlando, Florida, USA, July 21-25.

Vat, K.H. (2004c). On the idea of soft systems methodology for IS development: A perspective based on purposeful action. In *Proceedings of the International Conference on Computing, Communications and Control Technologies (CCCT2004)* (pp. 227-232), August 14-17, Austin, Texas, USA.

Vat, K.H. (2004e). Towards a learning organization model for PBL: A virtual organizing scenario of knowledge synthesis. In *Proceedings of the Seventh Annual Conference of the Southern Association for Information Systems (SAIS2004)*, Feb. 27-28, Savannah, Georgia, USA.

Vat, K.H. (2004f). Toward a learning organization model for student empowerment: A teacher-designer's experience as a coach by the side. In *Proceedings of the 2004 IADIS International Conference on Cognition and Exploratory Learning in Digital Age (CELDA2004)* (pp. 131-140), Dec. 15-17, Lisbon, Portugal.

Vat, K.H. (2005a). Modeling human activity systems for collaborative project work: An IS development perspective. *Journal of Issues in Informing Science and Information Technology, 2*, 49-65. Available at http://2005papers.iisit.org/I05f65Vat.pdf

Vat, K.H. (2005b). On the importance of human activity systems in organization modeling for IS development. In *Proceedings of the Eighth Annual Conference of the Southern Association for Information Systems (SAIS2005)*, Feb. 25-26, Savannah, Georgia, USA.

Vat, K.H. (2005c). Systems architecting of IS support for learning organizations: The scenario-based design challenge in human activity systems. *Information Systems Education Journal (ISEDJ)*, *3* (2). Available at http://isedj.org/3/2/

Vat, K.H. (2005d). Teaching a collaborative model of IS development through problem-based learning. In the *Proceedings of the 2005 Information Systems Education Conference (ISECON2005)*, Oct. 6-9, Columbus, Ohio, USA.

Vat, K.H. (2006). Developing a learning organization model for problem-based learning: The emergent lesson of education from the IT trenches. *Journal of Cases on Information Technology*, *8* (2).

Vaughn, Jr., R. B. (2001). Teaching industrial practices in an undergraduate software engineering course. *Computer Science Education*, *11* (1), 21-32.

Whitten, N. (1995). *Managing software development projects (*2nd ed.). John Wiley & Sons.

Wilson, B. (2001). *Soft systems methodology: Conceptual model building and its contribution.* New York: John Wiley & Sons.

Biography

Kam Hou VAT is currently a lecturer in the Department of Computer and Information Science, under the Faculty of Science and Technology, at the University of Macau, Macau SAR, China. His current research interests include learner-centered design with constructivism in Software Engineering education, architected applications developments for Internet software, information systems for learning organization, information technology for knowledge synthesis, and collaborative technologies in electronic organizations.