Educating Software Engineering Students to Manage Risk

Barry Boehm
University of Southern California
Department of Computer Science
Center for Software Engineering
Los Angeles, CA 90028 USA
213-740-8163
boehm@sunset.usc.edu

Daniel Port
University of Southern California
Department of Computer Science
Center for Software Engineering
Los Angeles, CA 90028 USA
213-740-7275
dport@sunset.usc.edu

ABSTRACT
In 1996, USC switched its core two-semester software engineering course from a hypothetical-project, homework-and-exam course based on the Bloom taxonomy of educational objectives (knowledge, comprehension, application, analysis, synthesis, evaluation). The revised course is a real-client team-project course based on the CRESST model of learning objectives (content understanding, problem solving, collaboration, communication, and self-regulation). We used the CRESST cognitive demands analysis to determine the necessary student skills required for software risk management and the other major project activities, and have been refining the approach over the last four years of experience, including revised versions for one-semester undergraduate and graduate project course at Columbia.

This paper summarizes our experiences in evolving the risk management aspects of the project course. These have helped us mature more general techniques such as risk-driven specifications, domain specific simplifier and complicator lists, and the schedule as an independent variable (SSIV) process model. The largely positive results in terms of review pass/fail rates, client evaluations, product adoption rates, and hiring manager feedback are summarized as well.

Keywords
Software engineering education, project courses, risk management, process models, product models, property models, success models

1 MOTIVATION & CONTEXT

The increasing pace of change in information technology (IT) makes one-size-fits-all, cookbook solutions increasingly inadequate. Yet students are largely educated on cookbook solutions to set-piece problems (e.g., compiler design and development). Applying cookbook solution approaches to current IT applications frequently leads to:

- ?? Good solutions to the wrong problems;
- ?? Large amounts of late rework;
- ?? Overemphasized or underemphasized activities through inability to determine "how much is enough?"

A good education in risk management provides skills and methods for dealing with these problems.

- ?? Addressing the risks of building the wrong system focuses software engineers on understanding the stakeholders' objectives and context while exploring solution approaches;
- ?? Resolving risks early avoids extensive late rework;
- ?? "How much is enough" questions are best addressed by considering the risks of doing too much or too little.

Educating students in risk management is not easy. Usually risk-management skills take years to acquire. The major challenges are learning how to recognize and deal with particularly risky personal tendencies and external constraints. These include:

- ?? A desire to please, which leads to risky over commitments. For example, in Weinberg's celebrated experiment [11], five teams were asked to build a software system while optimizing a different criterion (effort, size, memory, program clarity, output clarity). Each team ranked first (or in one case, tied for first) in what they were asked to optimize.
- ?? A tendency to focus on a single criterion (budget, schedule, performance, features, correctness) at the risk of seriously underemphasizing others. For example, in the Weinberg experiment, the team asked to minimize effort did so, but in the process, the finished last in program clarity, next-to-last in size and memory, and third in output clarity.
- ?? Inappropriate solution paradigms, such as "Do the

LEAVE BLANK THE LAST 2.5 cm (1")
OF THE LEFT COLUMN ON THE FIRST PAGE
FOR THE COPYRIGHT NOTICE.
Preserve these six lines in some
cases, but make their contents
blank in your text.

0-7695-1050-7/01 $10.00 © 2001 IEEE

easiest parts first, and the hard parts will get easier." This works fine for crossword puzzles, jigsaw puzzles, and some simple computer programs. But the strategy, "Let's do the easy parts first, and add fault-tolerance and computer security later," has been a consistent failure.

?? Risk-insensitive progress metrics, such as "finish the requirements by Day X (even if they haven't been verified for feasibility);" "Drive down the number of problem reports as fast as possible (do the easiest first)."

?? The thrills of crisis management. The joys and habits established by students pulling all-nighters to finish term projects are easy to see later on in the "Wyatt Earp syndrome" of the cowboy programmer galloping in and working around the clock to save the project community from disaster.

Thus, there are significant advantages to be gained, and formidable challenges to address, in educating software engineering students to manage risk. Section 2 presents our overall course approach and set of educational strategies for addressing these challenges, including a Cognitive Demands Analysis relating project tasks to risk management skills needed. Section 3 provides details on particular course practices, organized around the tasks in the Cognitive Demands Analysis. Section 4 summarizes our results and conclusions to date.

2 COURSE APPROACH AND EDUCATIONAL STRATEGIES

We have evolved several key risk-management educational strategies over 8 years of offering a 2-semester software engineering course. A primary strategy is to involve the students in a full life cycle (through product transition) team project with real clients. This gives the students first-hand personal experience in the effects of making risky decisions on their team's performance and on the clients' satisfaction with their product.

Another primary strategy is to use a risk-driven process model for the project. We use an extension of the spiral model called Model-Based (System) Architecting and Software Engineering (MBASE) [1;4]. The biggest risk is for the team to deliver an unsatisfactory architecture package (defined later) within 12 weeks in the fall semester and to deliver and transition an unsatisfactory product within 12 weeks in the spring. In this case, the best process model is a variant of Rapid Application Development (RAD) called "schedule as independent variable (SAIV),"

Table 1. Cognitive Demands Analysis: Early Risk Management Skills

Project Tasks	Risk Management Skills - Skill-building activities
?? Select projects; form teams	?? Project risk identification ?? Staffing risk assessment and resolution - Readings, lectures, homework, case study, guidelines
?? Plan early phases	?? Schedule/budget risk assessment, planning ?? Risk–driven processes (spiral, MBASE) - Readings, lectures, homework, guidelines, planning and estimating tools
?? Achieve stakeholders' shared vision	?? Simplifier/complicator analysis ?? Prototyping as buying information to reduce risk - Readings, lectures, homework, prototype, WinWin tool
?? Formulate, validate concept of operation	?? Risk-driven level of detail - Readings, lecture, guidelines, project
?? Manage to plans	?? Risk monitoring and control - Readings, lecture, guidelines, project
?? Develop, validate LCO* package	?? Risk assessment and prioritization - Readings, lecture, guidelines, project
?? LCO Architecture Review	?? Risk-driven review process ?? Review of top-N project risks -Readings, lecture, case studies, review

* MBASE LCO (Life Cycle Objectives) package includes Operational Concept Description, Prototype, Requirements Description, Architecture Description, and Feasibility Rationale

in which the size of the prioritized feature set to be delivered becomes a dependent variable [8].

For risk management and other key skills, we use a cognitive demands analysis to determine which individual skills are important to learn, and what sequence of educational experiences (reading, lectures, homework, case studies, project guidelines, decision aids, tools, project feedback) are likely to be most effective in helping the students learn the skills. Our risk management framework is an extension of the risk assessment (identification, analysis, prioritization) and risk control (planning, resolution, and monitoring) framework [12]; supplemented by the SEI risk taxonomy [13] and the project risk techniques in [14]. The cognitive demands analysis is a key feature of the CRESST (Center for Research on Evaluation, Standards, and Student Testing, UCLA) model of learning, organized around Content Understanding, Problem Solving, Collaboration, Communication, and Self-Regulation [6; 7]. We have found that the CRESST model's emphasis on collaboration and self-regulation skills makes it a better framework of learning objectives for software engineering project courses than the classical Bloom taxonomy [5]. The CRESST approach addresses the organizational and behavioral concerns in software engineering in a similar way to the Context Understanding and Strategic Thinking approaches in [18]; as contrasted to the more traditional computer-science Knowledge Units emphasized in [19].

We use several reflection-in-action techniques to provide students with feedback and the opportunity to reflect on the risk aspects of their project actions. These include inspections, architecture review boards [50], grading criteria, monitoring of their risk management plans, student critiques of their project experiences, and client evaluations.

Cognitive Demands Analysis Overview

Every software engineering project course has a tremendous challenge to fit in all the skills that seem to be needed during the first week of the project. These include software processes, requirements, domain understanding, client interaction, team selection and teambuilding, project

Table 2. Software Risk Management Techniques

Source of Risk	Risk Management Techniques
1. Personnel shortfalls	?? Staffing with top talent; key personnel agreements; team-building; training; tailoring process to skill mix; walkthroughs.
2. Schedules, budgets, process	?? Detailed, multi-source cost and schedule estimation; design to cost; incremental development; software reuse; requirements descoping; adding more budget and schedule; outside reviews.
3. COTS, external components	?? Benchmarking; inspections; reference checking; compatibility prototyping and analysis
4. Requirements mismatch	?? Requirements scrubbing; prototyping; cost-benefit analysis; design to cost; user surveys
5. User interface mismatch	?? Prototyping; scenarios; user characterization (functionality; style, workload); identifying the real users
6. Architecture, performance, quality	?? Simulation; benchmarking; modeling; prototyping; instrumentation; tuning
7. Requirements changes	?? High change threshold: information hiding; incremental development (defer changes to later increments)
8. Legacy software	?? Reengineering; code analysis; interviewing; wrappers; incremental deconstruction
9. Externally-performed tasks	?? Pre-award audits, award-fee contracts, competitive design or prototyping
10. Straining computer science	?? Technical analysis; cost-benefit analysis; prototyping; reference checking

organization and planning, and the use of various project tools, methods, and guidelines. How can one fit risk management in as well?

We do this by using techniques that involve the students in problem-solving activities requiring combinations of needed skills. Thus, stakeholder win-win negotiations involve client interaction, teambuilding, and domain understanding; they produce spiral model objectives, constraints, and alternatives, plus the beginning of a requirement specification. And risk management can be combined with many of the required skills, as seen in Table 1, which shows the early risk management portions of our cognitive demands analysis. Drafts of the LCO package material are completed in Week 6; the LCO ARB reviews are in Week 7-8. Examples in Table 1 include combinations of risk management and project selection, staffing, planning, stakeholder negotiation, operational

Table 3: Process Model Decision Table

Objectives, Constraints			*Alternatives*		**Model**	**Example**
Growth Envelope	Understanding of Requirements	Robustness	Available Technology	Architecture Understanding		
Limited			COTS		Buy COTS	Simple Inventory Control
Limited			4GL, Transform		Transform or Evolutionary Development	Small Business-DP Application
Limited	Low	Low		Low	Evolutionary Prototype	Advanced Pattern Recognition
Limited To Large	High	High		High	Waterfall	Rebuild of old System
	Low	High			Risk Reduction Followed by Waterfall	Complex Situation Assessment
		High		Low		High-Performance Avionics
Limited to Medium	Low	Low-Medium		High	Evolutionary Development	Data Exploitation
Limited to Large			Large Reusable Components	Medium to High	Capabilities-to-Requirements	Electronic Publishing
Very Large		High			Risk Reduction & Waterfall	Air Traffic Control
Medium To Large	Low	Medium	Partial COTS	Low to Medium	Spiral	Software Support Environment

Conditions for Additional Complementary Process Model Options

Design-to-cost or Design-to-schedule	Fixed Budget or Schedule Available
Incremental Development (only one condition is sufficient)	Early Capability Needed Limited Staff or Budget Available Downstream Requirements Poorly Understood High-Risk System Nucleus Large to Very Large Application Required Phasing With System Increments

concept formulation, etc. These will be elaborated in Section 3, following the order of project tasks in Table 1.

3 COURSE PRACTICES

Project Startup

The initial team project tasks shown in Table 1 are selecting a project and forming a team. The first lecture and readings in the course include material on project and staffing risk identification.

One main topic covered by the lecture is a top-level risk identification checklist from the MBASE Guidelines (shown as Table 2). This is a 1995 update of the 1989 survey of the top-10 sources of software risk given in [12]. The top two sources of risk in 1995 were still personnel shortfalls and unrealistic schedules and budgets (unrealistic processes were added in 1995). External-component risks were #7 in 1989; with the proliferation of variable-quality commercial-off-the-shelf (COTS) products, this risk had escalated to #3 in 1995. Other increasing sources of risk in 1995 were risks associated with software architecture (#6) and legacy software (#8).

For experience in assessing project risks, a homework assignment is given to risk-analyze a case study of a failed project using Table 2. For assessing project staffing risks, additional lecture material is provided. It summarizes the experience of previous years' projects that the most significant sources of staffing risks were, in priority order: lack of commitment (most often with final-semester students), interpersonal compatibility, critical project skills (both technical and management), and communication (e.g., teams involving students from the USA, Brazil, France, India, and Korea).

The students can then use these risk-sources to guide their selection of project teammates, and subsequently reflect on how well they had been applied in their post-project critiques. The number of critiques expressing regret at not addressing these risk sources more carefully is decreasing, but has not yet reached zero.

Early Project Planning

Early project planning highlights the #2 risk in the course: an inflexible 12-week period to complete a Life Cycle Architecture (LCA) package, consisting of definitive versions of each of the artifacts contained in preliminary form in the LCO package described at the bottom of Table 1. It also highlights the fact that only 12 weeks are available in the spring semester to develop an Initial Operational Capability (IOC) and transition it to the clients. This is a particularly risky prospect, as the students generally disappear at the end of the semester, and the clients must fully assimilate the product.

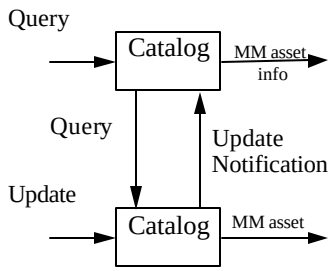
The COCOMO II model [10] is covered as an estimating tool by the COCOMO II book as textbook, along with associated lecture material and a homework exercise. Another key risk assessment and planning asset in the MBASE guidelines is the Process Model Decision Table shown in Table 3. It provides process choices that minimize the risk of a model clash between the process model selected and characteristics of the system's product models (available technology, understanding of product requirements and architecture), property models (robustness, fixed budget or schedule), or success models (growth envelope, phasing with system increments).

The fixed schedules in the fall and spring indicate in the conditions at the bottom of Table 3 that a design-to-schedule or schedule-as-independent variable (SAIV) process model will minimize the risk of a project overrun. Specific SAIV techniques included in the lectures and guidelines involve having clients prioritize features and identify a core capability buildable in an estimated 60-70% of the available schedule; architecting the software for ease of dropping or adding low-priority features; and planning an incremental development. This would establish the core capability as increment 1 and add features as appropriate with the remaining schedule.

Achieve Stakeholders' Shared Vision

The course projects currently use the Easy WinWin

Figure 1. Developer-Side Simplifiers and Complicators

Sub-Domain/Block Diagram	Simplifiers	Complicators
Multimedia Archive 	?? Use standard query languages ?? Use standard or COTS search engine ?? Uniform media formats	?? Natural language processing ?? Automated cataloging or indexing ?? Digitizing large archives ?? Digitizing complex or fragile artifacts ?? Rapid access to large Archives ?? Access to heterogeneous media collections ?? Automated annotation, description, or meanings to digital assets ?? Integration of legacy systems

collaboration tool to help negotiate and prioritize requirements and achieve a shared vision. This is a recently-developed commercial version of earlier WinWin tools [16], based on GroupSystems.com's collaboration infrastructure [17]. It helps identify risks by having stakeholders assess the relative value and difficulty of achieving a given win condition. High-risk win conditions have either uniformly high difficulty ratings or a lack of consensus on their ratings. Easy WinWin is also used to prioritize features for defining the core capability in the risk-driven SAIV process model. Prototypes are developed concurrently with EasyWinWin negotiations to reduce the risk of mis-understanding user operational requirements.

A particularly effective tool used to reduce the risk of client overexpectations is a simplifiers and complicators (S & C) list. An example is shown in Figure 1 for a particular digital library subdomain of projects: multimedia archives. Figure 1 shows a baseline architecture for multimedia archives, and lists of features that make a project more simple (low-risk) or complex (high-risk). Providing these S & C lists to project teams, along with lectures and a homework exercise, reduced the LCO review failure rate from about 25% in 1996 and 1997 to about 5% in 1998 and 1999 [2]. Figure 2 shows an example project S & C analysis for a multimedia archive of Asian films. It helped the librarian and students avoid such high-risk features such as natural language processing and over-sized film clips.

Formulate, Validate Concept of Operation

MBASE has a set of invariants such as stakeholder win-win (win-lose usually turns into lose-lose), the LCO, LCA, and IOC milestones, and a principle that the content of MBASE artifacts is risk-driven. This is the best way we have found of answering the "how much is enough?" question for prototyping, specifying, testing, configuration management, etc.

As an example, for the specification of system requirements or operational concepts, this invariant principle translates into:

If it's risky not to specify precisely, Do

(e.g., a safety-critical hardware-software interface)

If it's risky to specify precisely, Don't

(e.g., a GUI layout that can be easily evolved to match

Figure 2a. Asian Film Database Simplifiers Analysis

Specific Simplifier	Risks and Trade-offs
Uniform Media Formats All video clips are stored using an open file format for video/audio (e.g., MPEG). All film stills are stored using an open image file format (e.g., JPEG). The inverse complicator is to store film clips using streaming video technologies	This means that we may have to convert existing digital assets or digitize the original media, which may be costly. A unique file format limits the user base to those who have viewers for that particular file format The chosen file format may not be the most efficient for the various types of media (in terms of compression rates, quality, etc...)
Use Standard Query Languages Organize catalog and archive relationally so that queries will be limited to standard search formats	May not be as effective for "discovering" assets in the archive: users must know what they're looking for, in order to search for it
Use Standard COTS Use a standard Relational Database Management System (RDBMS) that supports storing multi-media asset	A Relational Database Management System may not be most suited for archival of multimedia assets, may have high initial, implementation, or administration costs

Figure 2b. Asian Film Database Complicators Analysis

Specific Complicator	Risks and Trade-offs
Natural Language Processing Store the information only in one language (e.g., English) and provide dynamic translation into Chinese, Japanese and Korean The inverse simplifier is to store the same information in 4 different languages (English, Chinese, Japanese and Korean).	The first approach is a complex, error-prone, expensive natural language processing issue The second approach will require more storage space, in addition to acquiring the translations
Digitizing Large Archives Digitizing film clips from the entire collection of films (which grows at a very fast rate of 800 films per year for Indian films alone)	If each film's clips require around 100 MB, then the rate of growth of the database will be of 80 GB a year (excluding the size of the metadata or catalog information)
Integration of "Legacy" Systems Do not require Real-Video plug-in for Web browsers to allow users to view streamed film clips, as legacy systems do not support them.	We cannot use more effective multi-media formats, which are becoming standard technologies

uncertain user needs with a GUI-builder)

We have found this risk-driven specification approach much more effective for rapid-development web-based and multimedia systems than the traditional ideal of a complete, consistent, traceable, testable requirements specification. It takes some time for the students to get used to, as they are initially concerned that anything incomplete will reduce their grade. But our grading criteria penalize over-specification as well as under-specification.

Manage to Plans

The most effective technique we have found for monitoring risk management progress is the Top-N Risk Item List. An example from one of the student projects is shown as Table 4. It provides a compact, easily updated, and highly management-relevant summary of which risk items are growing or decreasing in criticality, and which ones are more and less rapidly getting resolved. Each week, the students submit an updated Risk Item List as part of their weekly progress report.

Continuous Risk Assessment and Control

Effective risk management requires continuous feedback and control from initial project inception to construction, transition and support. With the understanding that all project tasks involve risk and require risk management skills (as indicated in Table 1), students follow MBASE guidelines in utilizing a "mini" spiral cycle [3] of risk identification, risk assessment, and risk tracking throughout the project. The cycle begins by proactively identifying possible sources of significant risks and management approaches for their project with the aid of table 2. The cycle continues either by resolving the risk or by addressing its resolution in the risk management plan. This plan is monitored and updated, with re-scoping activities undertaken when risks are too hard to resolve. An elaboration of the basic spiral cycle is:

1. Identify new risks
2. Identify affects of risks
3. Assess risk exposure; reconcile risks with project goals, constraints, objectives
4. Evaluate risk reduction alternatives and risk reduction leverage
5. Take corrective action; assess decision points to invoke contingency plans
6. Perform top-N Risk Item Tracking (See Table 4)
   a. Identify top-N Risk Items
   b. Highlight these in regular project reviews (focuses review on manager-priority items)
   c. Focus on new entries and slow-progress items
7. Reassess top-N risks

Table 4. Example top-N risk item list.

Risk Items	Weekly Ranking			Risk Resolution Progress
	Current	Previous	# Weeks	
COTS mismatch	1	5	8	Push for early installation of all COTS packages and test its functionalities
Availability of Rational Clearcase 4.0 for NT and Microsoft Access database for on time delivery	2	10	4	Contacted Rational regional representative and notified him that we are on a time-constraint schedule
Schedule—an independent variable, delivery in 12 weeks	3	2	4	Prioritize requirements and use stage delivery to avoid schedule crunch
Budget—man-hours to be put in by the project team	4	3	4	Use familiar tools, use COTS packages, and add additional team member
One team member will not be available for one week in March, 2000, and another in May 2000	5	6	4	Let other team members help in his area
Poor communication with customers	6	9	3	Schedule weekly meetings. Use teleconferences, emails to facilitate communication
Requirement mismatch	7	6	2	Provide updates to stakeholders and collect inputs during reviews

4 PROJECT RESULTS AND CONCLUSIONS

Project Results

Table 5 summarizes three years' experience to date in applying and refining MBASE on an annual selection of real-client digital library projects.

A few explanatory comments on Table 5 are in order. The USC Fall course has a much larger enrollment than the Spring course, as the former is a core course for the USC MS in computer science. In 1996-97, the subset of projects to be continued in the Spring were primarily those having students continuing from the Fall course. After we found that most of the 1996-97 products went unused, we performed a critical success factor analysis, and determined a set of Spring project selection criteria (e.g., library commitment to product use; empowered clients) which increased the project adoption rate. Even then, the inevitable changes in Library infrastructure and organizational responsibilities have caused some applications' usage to be overtaken by events.

Almost every team so far has developed an acceptable LCA or IOC package on time and passed its final review. We credit this to three major MBASE emphases. (1) The strong MBASE emphasis on risk management, with the highest risk identified as missing the LCA and IOC delivery dates. (2) The MBASE stakeholder win-win orientation, which encourages student and librarian stakeholders to cooperate on prioritized desired capabilities and agreeing on core capabilities with low risk of on-time delivery. (3) Using a SAIV process model which avoids model clashes among the project's process model, product model (feature set), and property model (top priority on schedule-to-complete).

Our annual rework of the MBASE guidelines [9] including increasing emphasis on risk management has resulted in a number of improvements. The 1998-99 reduction in teams failing their LCO reviews resulted primarily from our introduction of the S & C expectations management activity as described earlier. The reduction in size of the LCO and LCA packages between 1996-97 and 1997-98 resulted from eliminating a number of redundancies in the package guidelines. A further reduction in size for the Columbia in the Fall 99 course is attributed to the enforcement of explicit "risk based documentation guidelines" according the "do/don't" principle expressed earler. Here package scores were discounted if they included superfluous or confusing material.

The effect of the risk based documentation principle is more pronounced within the graduate F99 projects as they were developed from scratch and the teams had the opportunity to apply the principle directly as they developed the models (and they generally had some experience in determining relevance of information). Note that the undergraduate F99 LCA average package size did not significantly decrease. A explanation for this is that these undergraduate team projects are "recycled" whereby the current undergraduate projects are taken from previous graduate course project. The previous project LCA package is given to the undergraduate team as a guide for their project. In this the undergraduates are likely "risk managing" with respect to the previous graduate team LCA package by being sure to include the same level of detail (under the risky assumption that the previous team's

Table 5: USC and Columbia Project Results

Metric	USC 1996-97	USC 1997-98	USC 1998-99	Columbia U-grad. S99	Columbia Grad. S99	Columbia U-grad. F99	Columbia Grad. F99
Fall Semester: LCA Package							
Teams	15	16	20	20	13	10	7
Students	86	80	102	107	59	44	26
Applications	12	15	17	10	10	10	7
Teams failing LCO review	4	4	1	10	6	5	1
Teams failing LCA review	0	0	0	0	1	1	0
Pages, LCO package	160	103	114	124	116	107	95
Pages, LCA package	230	154	167	142	142	140	109
Client Evaluation (1-5, 5 best)	4.46	4.67	4.74	-	-	-	-
Spring Semester: IOC Package							
Teams	6	5	6	Remained the same since projects were only one semester long			
Students	28	23	28				
Applications	8	5	6				
Teams failing IOC acceptance review	0	0	0	0	0	1	0
Applications satisfying clients (*teams)	5	5	6	20*	12*	10*	7*
Applications not overtaken by events	6	4	4	10	9	10	6
Applications continued	3	3	4	2	3	1	2
Applications used	1	3	3	10	5	7	3
Client evaluation	-	4.15	4.3	4.44	4.21	3.9	4.38

success would directly translate to their project).

Involving the clients in risk management activities throughout (e.g. WinWin, S & C) clearly contributed to virtually all delivered applications being rated as satisfactory by the clients. Note that in Table 5 the USC measure for this is the actual application being satisfactory whereas at Columbia it is the number of teams that delivered a satisfactory application as it is common for multiple teams to develop the same application.

Another notable result is the number of applications that were actually used by the clients after the course ended. The undergraduate projects had a significantly higher percentage of applications used. Once again this is due to reducing risk factors as resulting from the undergraduate projects being recycled from previous graduate projects. The projects precedence reduces the risk of an undesirable outcome on many fronts. For example the project is more clearly defined and many of the design risks have already been identified and perhaps resolved. Furthermore clients for recycled projects only choose ones that they already feel are of value to them. Often it is the same client for the previous project and they know what changes need to be made for it to be used this time whereas previously the project may have been more exploratory. These factors and many more reduce the overall risk that the project will not deliver an application that will be used. In contrast, the graduate projects are typically unprecedented

Overall a particularly satisfying result of teaching risk management is the feedback we get from the students, clients, and hiring managers that have employed our students. Here are some examples:

Student: " I hate to waste time. The risk-driven specs idea helped me focus on the stuff that was really needed."

Client: "Discussing the simplifiers and complicators was an eye-opener for me. It's helped me understand what is reasonable to expect from information technology."

Hiring Manager: "It was remarkable to have summer student interns who knew how to manage risk."

The hiring manager feedback on risk management-skills has been particularly stronger since we switched in 1996 from a Bloom-taxonomy, homework-and-exam approach to risk management current CRESST-model, project oriented approach.

Conclusions

We have found that the Cognitive Demands Analysis and its associated educational activities have has helped students become effective not only in risk management, but also in such skills as process definition, client interaction, requirements negotiation, software and system architecting, project organizing and planning, and product validation and transition.

Most importantly, the students do not just learn risk management in their head with lectures, readings, simple exercises, and tests. They also learn risk management in their heart via stakeholder win-win negotiations to resolve initial risks, and via top-N risk item lists to track risk resolution progress and to apply corrective actions. And they learn risk management in their gut by overcoming their built-in desires to please, desires to do the easy things or fun things first, and desires to avoid confrontation in face-to-face discussions with clients to convince them that there are serious risks that need to be addressed. In a world where the bearers of bad tidings are often subject to the "Shoot the messenger" syndrome, it takes real gut knowledge and courage to convince reluctant clients that they will be better off acknowledging and dealing with their risks early and well.

REFERENCES

1. B. Boehm, A. Egyed, D. Port, A. Shah, J. Kwan, R. Madachy, "A Stakeholder Win-Win Approach to Software Engineering Education", Annals of Software Engineering, April 1999.
2. B. Boehm, M. Abi-Antoun, J. Kwan, A. Lynch, and D. Port, "Requirements Engineering, Expectations Management, and the Two Cultures," Proceedings, 1999 International Conference on Requirements Engineering, June 1999.
3. B. Boehm, A. Egyed, J. Kwan, D. Port, A. Shah, R. Madachy, "Using the WinWin Spiral Model: A Case Study", IEEE Computer, July 1998, pp. 33-44
4. B. Boehm, D. Port, "Escaping the Software Tar Pit: Model Clashes and How to Avoid Them ", ACM Software Engineering Notes, January 1999, pp. 36-48.
5. B. Bloom, Taxonomy of Educational Objectives: Handbook I: Cognitive Domain, David McKay, New York, 1956.
6. C. L. Baker, P. Aschbacher, D. Niemi, and E. Sato, "CRESST performance assessment models: Assessing content area explanations," Los Angeles, University of California, National Center for Research on Evaluation, Standards, and Student Testing, 1992.
7. H.F. O'Neil, Jr., K. Allred, and E. L. Baker, "Design of teacher-scored measures of workforce readiness competencies," in H.F. O'Neil, Jr. (Ed.), Workforce readiness: Competencies and assessment, Mahwah, NJ: Lawrence Erlbaum Associates, 1997.
8. B. Boehm, "Making RAD Work for Your Project," USC-CSE Technical Report 99-512. Abridged version in IEEE Computer, March 1999, pp. 113-117.
9. B. Boehm, D. Port, M. Abi-Antoun, and A. Egyed, "Guidelines for the Life Cycle Objectives (LCO) and the Life Cycle Architecture (LCA) deliverables for Model-Based Architecting and Software Engineering

(MBASE)", http://sunset.usc.edu/TechRpts/Papers/usccse98519/usccse98-519.pdf

10. B. Boehm, C. Abts, A.W. Brown, S. Chulani, B. Clark, E. Horowitz, R. Madachy, D. Reifer, and B. Steece, Software Cost Estimation with COCOMO II, Prentice Hall, 2000.
11. G. M. Weinberg, E. M. Schulman, “Goals and Performance in Computer Programming,” Human Factors, 1974, 16(1), 70-77.
12. B. Boehm, Software Risk Management, IEEE-CS Press, 1989
13. M. Carr, S. Konda, I. Monarch, F. Ulrich, and C. Walker, “Taxonomy-Based Risk Identification,” CMU/SEI-93-TR=6, Software Engineering Institute, Pittsburgh, PA 15213, 1993.
14. E. M. Hall, Managing Risk, Addison Wesley Longman, 1998.
15. AT&T, “Best Current Practices: Software Architecture Validation,” AT&T, Murry Hill, NJ, 1993.
16. B. Boehm, P. Bose, E. Horowitz, and M. J. Lee, “Software Requirements as Negotiated Win Conditions,” Proceedings, ICRE 94, April 1994, pp. 74-83
17. P. Gruenbacher, Easy Win Win Process Guide, Group Systems.com and USC-CSE, 2000
18. S. R. Faulk, "Achieving Industrial Relevance with Academic Excellence: Lessons from the Oregon Master of Software Engineering," Proceedings, ICSE 2000, June 2000, pp. 293-302.
19. W. R. Adrion, "Developing and Deploying Software Engineering Courseware in an Adaptable Curriculum Framework," Proceedings, ISCE 2000, June 2000, pp. 284-292.