Integrating Risk Management into an Undergraduate Software Engineering Course

James S. Collofello
Department of Computer Science and Engineering
Tempe, Arizona 85287-5406
collofello@asu.edu

Andrew K. Pinkerton
Excite, Inc.
drew@webcrawler.com

Abstract - *Risk management is one of the key practices of the Software Engineering Institute Capability Maturity Model. The effective management of risk is crucial to the success of software projects. Much has recently been written concerning risk management in an industrial environment. One of the most useful documents is a risk management questionnaire developed by the Software Engineering Institute. The questionnaire consists of 194 questions that a software development team can use to identify risks in their project. Unfortunately very little has been written about the risks faced by undergraduate software development teams and how they might manage them. This paper describes the introduction of risk management in an undergraduate software engineering course. The course requires students to work in teams of 5-6 persons to develop a software application in a one-semester time frame following a systematic development process. An academic version of the Software Engineering Institute risk management questionnaire suitable for undergraduate teams is described. This questionnaire addresses the real risks that an undergraduate software development team is likely to face and is based on years of our experience and that of others teaching these types of classes. The questionnaire and related risk forms and materials are described in detail as well as our experience in using these materials with 2 classes.*

Background

Much has been published in the literature concerning risk management [1-5]. Risk can be defined as exposure to adverse events, which can cause harm or loss. Risk management attempts to minimize the probability of adverse events occurring as well as their consequences.

The first step of the risk management process is identification of potential risks. Insight, experience and checklists provide the primary source of risk items. One of the best sources for potential risk items is the Software Engineering Institute risk management questionnaire. The questionnaire consists of 194 questions that a software development team can use to identify risks in their project. The questions are broad in scope and address product engineering issues, development environment concerns and program constraints [5]. Once the risks are identified, they must be prioritized based on the probability of occurrence as well as the potential consequences. High-risk items must then be managed by developing plans to minimize the probability of risk occurrence as well as the consequences should the risk occur.

Risk management activities should occur at strategic planning points in the project as well as when new risks are encountered. An effective risk management culture involves the entire team and is not just limited to management.

Integration of Risk Management into a Software Engineering Course

The undergraduate software engineering course at Arizona State University is a one-semester project course in which students work in teams of 5-6 members to develop a software application for a customer. The course project typically spans the entire semester starting with the teams defining the projects' requirements and ending with acceptance testing for the customer. The development of the course project follows a defined and documented methodology created by the instructor. The teams are organized as self-directed work teams and are responsible for planning and tracking their activities.

Risk management content is introduced in this class via 2 days of lecture followed by having each team perform a risk management exercise. The risk management content is introduced about 5 weeks into the semester project at the point where the teams have completed their requirement's documentation and are ready to plan the remainder of the project. The lecture content motivates the need for risk management and presents at a high level the content of the SEI Risk Management Process [4,5].

The risk management exercise consists of each team identifying risks relevant to their project and completing a risk management plan for the highest risk items. To facilitate this task an academic version of the SEI Risk

Management Questionnaire was developed. The academic version deleted those items that were not relevant to student teams, modified others to relate to the academic environment and added new items unique to student projects. The academic version consists of 36 questions. A condensed version of the questionnaire is contained in Appendix *A* and a complete version can be found in our web site [6].

Each team member must individually complete the questionnaire in order to identify possible risks. Each team is then required to meet in order to prioritize the risks based on probability of occurrence and consequences. Depending upon the team size, 5 or 6 of the highest risks (one per team member) are selected for further study. The risks are then divided among the team members and each team member is required to further analyze the assigned risk and develop and document a plan for managing the risk. Guidance for this task is provided with both lecture and reading material [6]. The results of the analysis and planning are documented in the form included in Appendix *B*. The team must review all of the risk management plans and commit to executing them.

Throughout the remainder of the project, teams are encouraged to periodically review the risk questionnaire to identify new risks and to follow through on their risk management plans.

Results

The results of integrating risk management into our undergraduate software engineering course have been very positive and have led to benefits both for the students and the instructor. The risk management exercise has helped teams identify risk items, prioritize them and take action to minimize the risks and their consequences. Some of the common high-risk items reported by the teams include:

1. lack of experience with development environment
2. insufficient time to perform unit testing
3. algorithms and designs difficult to implement
4. key module not complete or on schedule
5. insufficient integration time
6. design changing while coding being done

Feedback from students on successful projects suggests that the risk management exercise helped their team while unsuccessful project teams often reported in their post-mortem reports that the risk items identified were never managed.

Analysis of the risks identified by the students has also proven beneficial to the instructor. Identified risks such as: *insufficient integration time* or *lack of experience with development environment* can suggest the need for schedule revisions, exercises or additional lecture content. The value of a disciplined methodology on the project is also easier to convey to the students in the context of the risk items it addresses.

Future Work

Research is continuing in this area to develop an expert system for academic software engineering teams to use in risk management. The expert system will have an interface analogous to the academic version of the risk questionnaire. The system will both help teams to identify risks as well as suggest ways for reducing the probability of risk occurrence and minimizing the impact should the risk occur.

References

[1] Boehm, B., “Software Risk Management”, *IEEE Computer Society Press*, 1989.

[2] Charette, R., *Software Engineering Risk Analysis and Management*, 1989.

[3] Grey, S., *Practical Risk Assessment for Project Management*, 1995.

[4] Higuera, R. and Haimes, Y., “Software Risk Management”, *CMU/SEI-96-TR-012*, 1996.

[5] Carr, M., (et. al.), “Taxonomy-Based Risk Identification”, *CMU/SEI-93-TR-6*, 1993.

[6] Pinkerton, A., *Software Risk Management Web Page*, http://www.eas.asu.edu/~riskmgmt/

Appendix A

Question List for Software Risk Identification in the Classroom

Many of these questions were taken (at least in part) from the Software Engineering Institute's Taxonomy-Based Risk Identification Questionnaire. Each of these questions is from the Product Engineering section of the Taxonomy-Based Risk Identification Questionnaire. Other questions have been added or modified to account for the uniqueness of the classroom environment. The questions are further subdivided by a major category (e.g., Requirements) and a sub category (e.g., Stability).

Requirements

Stability/Completeness [assessed by evaluating the amount of information in the requirements]
[1] Are the requirements changing or yet to be determined?
[2] Does the instructor have unwritten requirements or expectations?

Clarity [assessed by evaluating your comprehension of the requirements]
[3] Are you able to understand the requirements as written?

Feasibility [assessed by evaluating the possible difficulties that might arise later in the project]
[4] Are there any requirements that are technically difficult to implement?

Tracking [assessed by evaluating the ability to keep requirements visible during the project]
[5] Do you have a plan to track the requirements throughout the design, coding and testing phases?

Design

Functionality [assessed by evaluating the feature set of and capabilities of the product]
[6] Are there any specified algorithms that may not (or only partially) satisfy the requirements?

Difficulty [assessed by evaluating the effort involved in producing the design]
[7] Does any of the design depend on unrealistic or optimistic assumptions?
[8] Are there any requirements or functions that are difficult to design?

Interfaces [assessed by evaluating the connections between components, or to the outside world]
[9] Are the internal and external interfaces well defined?

Performance and Quality [assessed by evaluating the functionality and quality of the product]
[10] Are there any problems with the expected performance, or quality, of the design?

Testability [assessed by evaluating the effort required to sufficiently test the product]
[11] Is the software going to be easy to test?

Hardware Constraints [assessed by evaluating the hardware of the target or development platform]
[12] Does the development or target hardware limit your ability to meet any requirements?

Software Reuse [assessed by evaluating the extent to which software is reused in the product]
[13] Does re-used or re-engineered software exist?

Code and Unit Test

Feasibility [assessed by evaluating the relative ease necessary to perform code and test]
[14] Are any parts of the product implementation not completely defined by the design specification?
[15] Are the selected algorithms and designs easy to implement?

Testing
[16] Is there sufficient time to perform all the unit testing that you specified?
[17] Will compromises be made regarding unit testing if there are schedule problems?

Coding/Implementation
[18] Are the design specifications in sufficient detail to write the code?
[19] Is the design changing while coding is being done?
[20] Is the language suitable for producing the software of this program?
[21] Does your team have enough experience with the development language, platform or tools?
[22] Is there a risk that a key component or module will not be complete or on schedule?
[23] Are you comfortable with your teams estimate on coding time and effort?
[24] Do you have a plan for configuration management of the code?

Integration and Test

Environment [assessed by evaluating the hardware and software support facilities and test cases]

[25] Will there be sufficient hardware to do adequate integration and testing?
[26] Is there any problem with developing realistic scenarios and test data to demonstrate any requirements?

Product [assessed by evaluating the integration and testing of groups of components]
[27] Have acceptance criteria been agreed to for all requirements?
[28] Has sufficient product integration been specified, and has adequate time been allocated for it?

System [assessed by evaluating the integration between the product and target hardware]
[29] Have sufficient system integration and system integration time been specified?

Maintainability [assessed by evaluating the effort required to locate and fix errors]
[30] Is the product design and documentation adequate for another class to maintain the code?

Specifications
[31] Are the test specifications adequate to fully test the system?

Communication, Team Compatibility and Motivation

Communication [assessed by evaluating the ability of the team to exchange information]
[32] Is there a lack of good communication amongst your team?
[33] Is there a lack of good communication with your instructor about the project?

Compatibility of Team [assessed by evaluating the ability of the team to work productively]
[34] Is your team familiar to you; have you worked together on a team project before?
[35] Are tasks delegated fairly amongst your team?

Motivation of Team [assessed by evaluating the goals of the team]
[36] Is your team motivated to create a good product?

Appendix B

Sample Risk Management Form

Date	Class	Year	Sem.	Instructor	Student Name	Team #	Project

Identification

In this project, there is a risk of:

Analysis

Probability of this risk occurring [very low/low/medium/high/very high]:

Impact if this risk occurs [negligible/marginal/critical/catastrophic] to:

Performance Effort Schedule Support

Overall risk (see Impact/Probability Matrix), include possible interrelationships to other risks:

Overall Risk

Planning

Why is this risk important?

What information is needed to track the status of this risk?

Who is responsible for this risk activity?

What resources are needed to handle this risk activity?

Possible Action Plan [mitigate the risk via an immediate response].

Reduce the probability of the risk by:

Reduce the impact of the risk by:

Possible Contingency Plan [monitor risk and invoke a predetermined response if necessary].

Metrics used during tracking phase:

Trigger Contingency Plan when:

If trigger is reached, what must be done?