

“Teaching Software Engineering with Rational Unified Process® (RUP)”

**Jan Bergandy, Computer Science Department,
College of Engineering
University of Massachusetts Dartmouth
*Electrical & Computer Engineering***

Abstract

The paper describes a transition phase in introducing a two-semester senior level capstone project in software engineering as part of the Bachelor of Science in Computer Science degree at the University of Massachusetts Dartmouth. In this phase the existing, conventional senior level Software Engineering course was delivered in a new process-based, project-driven format. Previous practice was to teach software engineering material through lectures with a major project used to provide hands-on experience for the course contents. The approach used in our transition phase was based on a well defined and supported software process (in our case RUP) used in a major team project as the driving factor for delivery of course material in the lecture portion of the course. In short, the roles of the project and lecture components were reversed. The paper presents the rationale for this new approach, and describes challenges and lessons learned during delivery of the course in the new format.

Introduction

Currently our Accreditation Board for Engineering and Technology (ABET) accredited Bachelor of Science in Computer Science program includes a required senior level standard four-credit Software Engineering course. The course includes a major software development project and has a laboratory component. As part of the curriculum improvement process a new two-semester sequence of software engineering project courses was developed to implement a capstone project. The sequence will be offered in the senior year and will replace the current software engineering course absorbing its entire contents including material covered by lectures [1]. The capstone project by its nature focuses on developing a medium size software system to meet the needs of a real customer. The new sequence must integrate providing the student with a solid understanding of software engineering concepts with opportunity to learn hands-on how to use them effectively in a close-to-reality software development process. Meeting these two objectives through separate courses is more common. Our approach attempts to integrate both in a form where software process framework (RUP) [3] [4] [5] is used to support the project software process and at the same time serves as the structure for organizing and scheduling coverage of topics of software engineering. In the past few years this “learn when you need it” approach became more dominant in software industry which in the past heavily invested in keeping its engineering task force up-to-date with a

wide spectrum of new technologies. The previous business model assumed that an in-depth knowledge of a wide spectrum of technologies will help developers make better choices and build more competitive products. This idea, in principle, is not totally abandoned but the recent shake-up in the industry forced companies to assess their training strategies from a cost-benefit perspective. The new strategy adopted by most is based on providing formal training on specific technology only when a decision has been made to use it in an approved project. A large portion of such training is done internally and frequently handled through student-mentor training where both parties are employees. In consequence, the emphasis on “classroom” type training has greatly diminished. One of the practical reasons of the new business model of training employees is in the number, complexity, and the rapid change in the domain of software technologies. Companies found out that most of their training budget dollars were spent on technologies that were newer used in context of developing their products. The question is, how does this issue relate to the way we teach software engineering in our programs? One of the major responsibilities of a computer science program is to prepare students for the reality of working in software industry. This responsibility should be reflected by program objectives and outcomes properly mapped into program contents and structure. The capstone project is the most critical component in fulfilling this responsibility by bridging the reality of software development on an industrial scale with the virtual reality of college education. The capstone project fulfills this role by providing students with the opportunity for integration and reflection on value of knowledge and skills acquired in previous courses in the context of a large-scale problem from conception to implementation of the solution. The setting for this experience must emulate the industrial reality as close as possible by using real customers, processes, tools, and time/quality criteria. The setting must also reflect the reality of group/individual responsibility, compromise between time and quality, and the need for adapting and dealing with unexpected. Learning on the job and making critical decisions on partial knowledge are also part of this reality.

Rational Unified Process (RUP) as a framework of choice

The Process-centric approach in a software engineering course obviously impacts the of order material covered but more profoundly increases the significance of the process framework selection. The criteria used in selecting RUP as our process framework of choice including the rationale is provided below.

- The RUP framework is straightforward, well-defined, and very well documented and supported. These characteristics are critical for our approach where students with minimum guidance have to acquire working knowledge of the key elements of the software process and become productive performing assigned roles in the project.
- RUP promotes using use cases to capture software requirements. Students are intimately familiar with Unified Modeling Language (UML) and use cases through other courses.

- RUP as one of its “best practices” embraces component-based development, another concept very familiar to our students through prerequisite courses.
- RUP is rapidly becoming the industry standard.
- RUP structures development into four phases each done in one or more iterations allowing risk mitigation early and throughout the entire process. Emphasis on risk mitigation in projects with student teams “learning on the job” in attempt to produce a product in a scope of three months is a must.
- RUP is configurable and can be structured from low to high ceremony. This flexibility is critical in tuning the software process to reflect the lessons learned from the student projects conducted in previous semesters. Adjustments to the software process and supporting tools used in the projects is as important to the curriculum improvement process as the changes to course content and methodology of course delivery. Obviously such adjustments have to be made based on data provided by a well defined assessment.

Rational Unified Process (RUP) lifecycle

RUP consists of four phases (Inception, Elaboration, Construction, and Transition) and each phase involves one to several iterations. Each phase commences with a milestone. Completion of each phase is marked by a specific milestone (Figure 1).

Inception Phase:

- Objectives: determine system’s scope and vision, identify critical functionality, develop initial estimates of cost, timeline, and risk factors, identify at least one viable architecture.
- Lifecycle Objective Milestone: Customer and developer agree on estimates of scope (including requirements), cost/timeline of the project, and initial plan for mitigating risks.

Elaboration Phase:

- Objectives: construct detailed requirements model, design and validate architecture, baseline architecture, mitigate risks, refine cost/timeline estimates, define and implement the development environment, develop executable architecture and test it to mitigate all major risk factors.
- Lifecycle Architecture Milestone: Stability of the requirements and the architecture has been achieved, major risks have been addressed through testing and prototyping, the iterations for the Construction phase are developed with acceptable estimates, and the resources are in place.

Construction Phase:

- Objectives: conduct detailed design, implementation, and testing of the components, integration and testing of the system, and preparation for the transition of the system to the customer.

- Initial Operational Capability Milestone: Product is stable and both the developer and customer are ready for its deployment.

Transition Phase:

- Objectives: prepare resources and processes for product deployment, conduct beta-testing and possibly training of users for transition of the product to the customer
- Product Release Milestone: All conditions are met and the product is in the customer's hands.

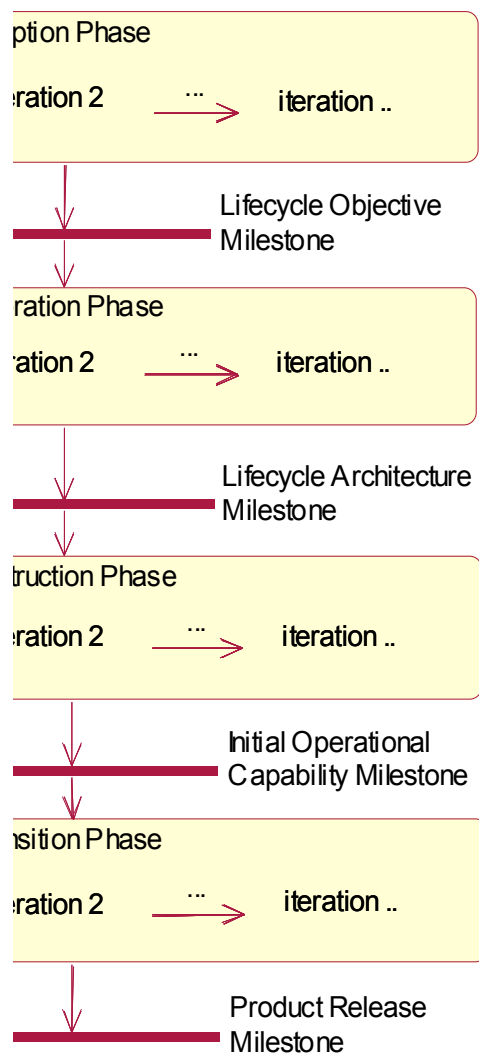


Figure 1. Rational Unified Process (RUP) Lifecycle

Using RUP framework in Software Engineering course

Rational Unified Process uses four basic concepts:

- Role (who?) – to define behavior and set of responsibilities of a person or team.
- Activity (how?)– to represent a specific task performed in the process.
- Artifact (what?)– to represent an outcome of activities (for example: document, model, software module, etc.).
- Workflow (when?) – to define order and relationships between activities of the process that must be completed in order to achieve a specific objective/produce an artifact.

In our approach students were provided with a very well documented RUP framework as a tool for defining their own process as opposed to defining a specific process based on RUP to be followed by all teams. Individual students were able to view the documentation for their assigned roles in order to understand their responsibilities in the development process, to identify activities they should perform in specific phases of the process, and the artifacts they should produce. Teams were able to use the framework's documentation to understand the project workflows within the phases of the development (Figure 2).

The decision of giving students the framework instead of a defined process was based on a premise that students, after being provided with the “grant tour” of the RUP framework, are able to make reasonable choices in selecting the right process elements for delivering the product (working software).

We believed that more can be learned by observing what they choose to use as oppose to asking them in the “postmortem” to identify what was missing.

Meaningful retrospection on the absence of process elements requires an in-depth understanding of the domain of (software) process engineering.

Course Organization

The teaching team consisted of an instructor and a teaching assistant. The instructor had an extensive experience with teaching a wide range of software engineering topics on the undergraduate and graduate level but never taught this required senior level software engineering course in the past. The teaching assistant was a graduate student who recently received a Bachelor Degree in Computer Science in our program and took this course taught in its original version one year ago.

The role of the instructor was to provide and implement the new vision for the course (project centric). The instructor was responsible for selection of RUP for the project process framework and for organization and delivery of the course material through lectures. The instructor, with the assistance of the teaching assistant, performed the job of the program manager (for all projects) and served as a liaison/vendor in interaction with all customers. The teaching assistant worked closely with the teams

as an observer/consultant in laboratories and during his office hours. His observations were communicated to the instructor on a daily basis.

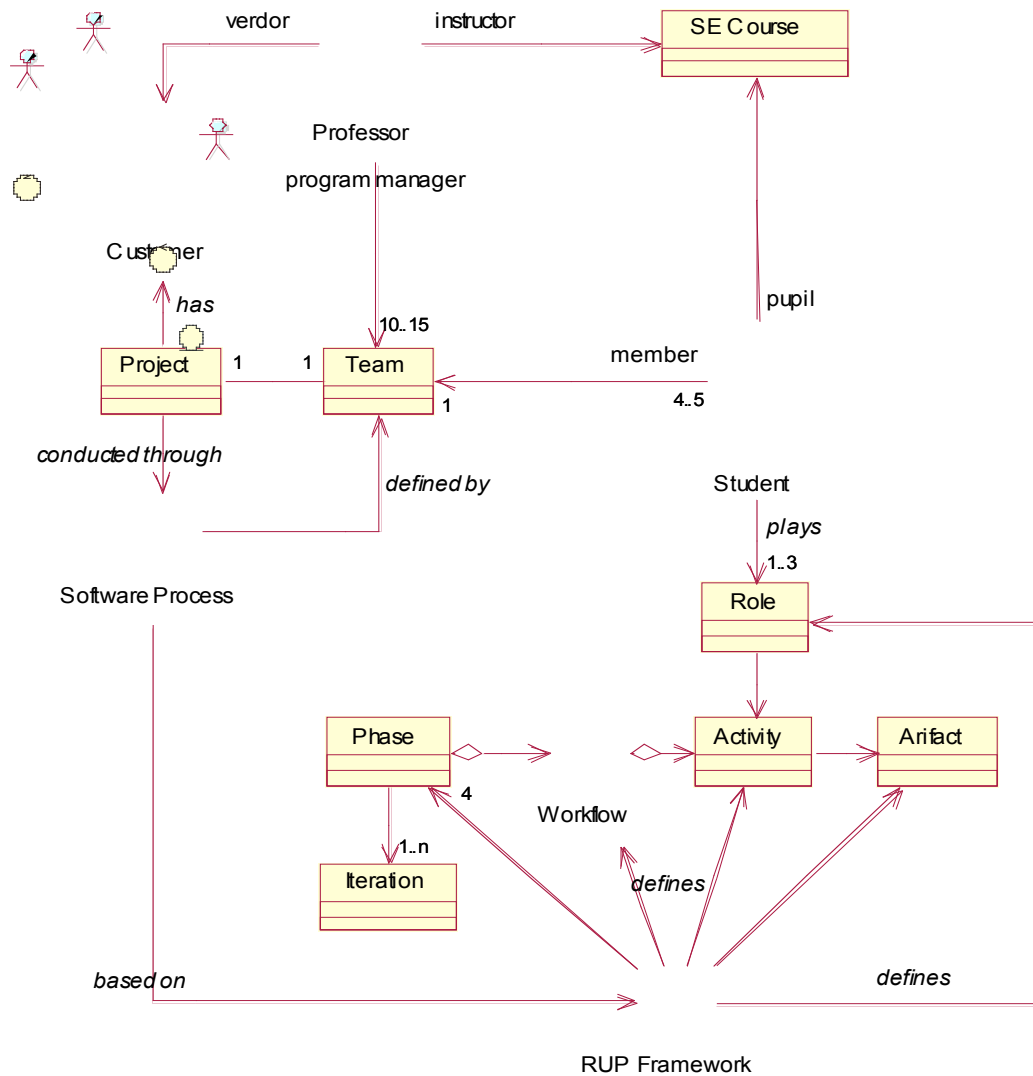


Figure 2. RUP Framework in SE Course

The learning portal was used as a vehicle for posting all course and project materials (including RUP framework documentation) as well as submission of all artifacts by students, and student teams. The portal was also used for communication between the teaching team, student teams, and individual students.

The student body consisted of Computer Science and Computer Engineering seniors. For both groups of students, this was a required course.

Projects were solicited in advance from on-campus and off-campus customers. In the first week of the semester all students submitted their resumes. The teaching team selected team leads for specific projects based on the project's characteristics, student's experience (resume), and academic standing (transcript). Team leads were notified about the nomination and provided with a brief description of the project previously obtained from customers. Over one weekend the team leads reviewed the project description, determined the set of skills needed for their project, and reviewed the resumes of all non-team lead students. They were advised to prioritize their choices of team members. The selection of team members by the leads took place using round-robin method and the teams started their projects in the second week of the semester.

Teams were given freedom on deciding which RUP roles were critical to the team and project and on assignment of selected roles to team members (with exception of the lead). This was done based on the RUP framework customized (by the instructor) for a small-team development. During the entire project teams interacted with the customer directly. The communication of the teaching team with projects' customers was minimal and it was restricted to problem resolution. Teams and individual students used RUP framework to define their process, understand the responsibilities of their assigned roles, research the workflows, and use the framework provided templates for the process artifacts.

Project Evaluation Process

Project evaluation process had three major objectives. The first objective was to evaluate the performance of each team in meeting the goals of the project and delivering the product on time. This obviously was important in grading students. The second objective was to evaluate the impact of using the RUP framework in students' projects. This objective played a key role in verification and validation of the project-centric approach to teaching software engineering using RUP as process framework. The third objective was to evaluate the use of Computer Aided Software Engineering (CASE) and other tools as well as use of methodologies in the development process. This objective was important in assessing students' ability to choose the right tool for the right task, their proficiency in using specific tools, as well as their overall ability to pull together what they learned in previous courses and use it effectively in their projects.

In the evaluation process students were required to address the following issues:

- Team dynamics: members and their roles, work distribution and organization, communication.
- Requirements engineering process (tools & methodologies used).
- Design and Architecture (how and what architecture was selected, design tools and methodologies used).

- Development (tools, languages).
- When, how, and what risks were identified? How did the team mitigate the identified risks?
- RUP – Did the RUP process used in this course help you or detract you from delivering the product on time? How?
- Issues and challenges faced by the team when conducting the project. How did the team deal with challenges?
- Lessons learned. What would you do differently and what would you do the same if involved in this type of project again?

The evaluation was conducted through weekly monitoring of student teams in laboratories, meeting with the teams outside of the laboratories, and through examination of artifacts produced by the teams. The complete list of elements used to conduct the evaluation is as follows:

- Weekly written progress reports were submitted through the portal together with updated schedules (Gant Chart Diagram). Weekly reports included goals for following week, accomplishments for the previous week, and tracking of project issues and their resolution.
- Teams working on the projects were monitored weekly during scheduled lab hours. The teaching Assistant worked as an observer/consultant with the teams engaging in project activities.
- Two meetings with each team were held during the semester to assess their progress. The meetings took place at the beginning of the Elaboration Phase and at the beginning of the Construction Phase of the project.
- Each team gave an in-class presentation addressing all of the issues listed above.
- The instructor met (30-45 minutes) with each team to ask questions about the team's handling of the project and following the software process by the team members. Again, all the issues listed above were addressed during the meeting with each team.
- A self evaluation form was designed and filled by each student at the end of the semester commenting on his/her project participation. Students had to address the RUP roles they played, the tasks and activities performed, and the artifacts produced. Each student had to estimate the time spent on each activity. The last portion of self evaluation asked for an assessment on how useful the RUP framework (and its documentation) was in helping the student to understand his/her role, tasks, activities, and artifacts that he/she was responsible for.
- An evaluation of other team members was conducted at the end of the semester providing the student with an opportunity to comment on the roles and effort of their teammates in the project.
- The instructor met with the customers at the end of the semester to obtain customers' perspective on the projects and the conduct of the teams.

How did this work?

- The project-centric approach required introducing RUP - a specific software process framework as the first topic in the course. The software process in its wider context was covered towards the end of semester. This approach was selected for two reasons:
 - Pragmatic: There was a vast body of topics immediately applicable to the projects that required coverage early-on in the semester. There was no adequate time for comprehensive coverage of the software process.
 - Retrospective: We believed that students will be able to better understand the software process concepts and appreciate their significance by relating the discussed material to their experience with a particular process implementation (their own project).

This approach worked very well. At the end of the semester, when the software process topic was formally covered, students showed a great appreciation and understanding of not only the software process issues but also software process improvement approaches as comprehensive as Capability Maturity Model ® Integration (CMMI).

- The impact of project-centric approach on course content and especially organization was not as severe as initially anticipated. The pace of material coverage in a typical course tends to be moderate at the beginning of the semester, increases as the semester progresses, and sometimes (if we do not plan right) reaches frenzy at the very end in a desperate attempt to cover it all. The experience with this course was somewhat opposite. With all the planning and preparation, the frenzy happened at the beginning of the semester. This was anticipated. The coverage of the material in lectures progressed in the last two thirds of the semester at a comfortable pace. We believe that extending the software engineering project into two semesters will eliminate this problem.
- Another minor challenge of the new approach was lack of synchronization between the order of material coverage in the course and its presentation order in a typical software engineering textbook. This did not have any detrimental effects in a senior level course.
- Communication between all stake holders was a key issue. Student teams were allowed to select their own method of communication between team members and between the team and the customer. Significant problems in this area impacted only one of thirteen projects and were related to inability of the customer to interact with the team in a timely fashion. Significance of establishing and maintaining effective communication between stakeholders throughout the project can not be underestimated especially in short projects

that allow little or no time for recovery. Using the learning portal for communication between the teaching team and the students worked very well.

- The RUP process consists of four phases, where each phase incorporates one to several iterations. Most teams, due to the short (three months) period to complete the project and due to lack of experience, were able to perform only a single iteration per phase. Where for Inception Phase and Transition Phase a single iteration would be sufficient, the remaining phases (Elaboration and Constriction) should include multiple iterations. Extending the software engineering project into two semesters will eliminate this problem.
- Development based on incremental releases considered as one of the “best practices” of RUP was not feasible for a project of such short duration. This issue will be addressed in the two-semester software engineering project in the near future.
- Configuration management was left to the discretion of student teams. Students were instructed about the need for version control. Half of the teams took the advice seriously and used CVS or Subversion to manage configuration in their projects. The remaining teams used informal methods of version control. All of these teams in “postmortem” analysis linked many encountered problems to lack of proper configuration management. In the future projects teams will be required to use a configuration management tool.
- Teams were encouraged to use “rolling wave” strategy for project planning [2]. The notion of “plan in detail short-term, plan in general long-term” was followed by most student teams consistently as observed through the examination of the weekly reports. Issues of scaling planning effort to risk, complexity, and newness of the project were discussed in class but were not incorporated into the projects. For obvious reasons, the two-semester project will require stronger emphasis on planning.
- Critical role of risk management was addressed very early in the semester. Teams were required to define and prioritize risks and develop a plan for risk mitigation. The teams performed risk analysis as one of the first tasks. Each team involved its customer in this process and used the teaching team members as consultants. From our observations, because of such short project duration, risk management was the most critical factor in determining the team’s success or failure.

Conclusion

The RUP framework has provided students with a baseline for validation of tasks performed, responsibilities to be fulfilled, and artifacts to be produced. It gave them a “solid ground” to stand on when deciding what steps to take in their projects.

The outstanding RUP documentation provided students with a “self-guided tour” of the process elements allowing the teaching team to focus on supplying students in a timely fashion with information essential to the conducted projects. Using RUP has made a significant difference in students’ introspection on the role of process and process improvement in software development.

Bibliography & References

- [1] J. Bergandy, 2005, *Object Paradigm in Software Engineering Curriculum*, Proceedings of the ASEE New England Section 2005 Annual Conference.
- [2] G. Githens, 1998, *Rolling Wave Project Planning*, Proceedings of the 29th Annual Project Management Institute 1998 Symposium , Long Beach, California, October 1998.
- [3] I. Jacobson, G. Booch, J. Rumbauch, 1999, *The Unified Software Development Process*, Addison-Wesley.
- [4] P. Kruchten, 2000, *The Rational Unified Process- an Introduction*, Addison-Wesley.
- [5] P.Kroll, P. Kruchen, 2003, *The Rational Unified Process Made Easy*, Addison-Wesley.

Dr. Jan Bergandy is a professor of Computer Science and the Director of Computer Science Graduate Program in College of Engineering at University of Massachusetts Dartmouth. He was involved in introducing object-paradigm based software engineering component in to the undergraduate degree program. For the past ten years he has been promoting object technology through academia and industry. Currently he is involved in design and implementation of a software engineering capstone project to become part of the BS in Computer Science program.

Contact information: Telephone 508-999-8293, E-mail jbergandy@umassd.edu