Software Engineering (CS 673)

Last Updated: Thursday April 15 @ 22:19

Announcements | Syllabus | Links | Documents | Final Exam

Course Syllabus

Instructor	Neel Pandeya 617-two-five-eight-1963 npandeya at bu dot edu • You can call me 24/7. If you leave a voicemail, be sure to leave the course number and your full name and phone number. • When sending me email, please always put the course number "CS 673" in the subject line, no matter what the email is about.
Date & Time	• Thursdays, from January 14 through April 29, from 6pm to 9pm • Class will be canceled on Thursday January 28 due to instructor travel • We will have a make-up class on Thursday March 11, which is during the BU Spring Break • There will be no class on Thursday April 22, per the BU academic calendar • See this BU Academic Calendar and this BU Academic Calendar .
Location	BU MET North Campus 100 Apollo Drive, Chelmsford MA 01824 Classroom 3
Website	http://cs673.neelpandeya.com/
WiFi Access	For information about on-campus WiFi access via your laptop computer, click here and here. .
Required Textbook	Software Engineering: Modern Approaches by Eric J. Braude, Michael E. Bernstein

	Hardcover: 800 pages Publisher: Wiley; 2 edition (April 5, 2010) ISBN-10: 0471692085 ISBN-13: 978-0471692089
Optional Textbooks	1. Software Engineering: A Practitioner's Approach by Roger S. Pressman Hardcover: 928 pages Publisher: McGraw-Hill; 7 edition (January 20, 2009) ISBN-10: 0073375977 ISBN-13: 9780073375977 2. A Discipline for Software Engineering by Watts S. Humphrey Hardcover: 816 pages Publisher: Addison-Wesley; 1st edition (December 31, 1994) ISBN: 0201546108 3. Introduction to the Team Software Process by Watts S. Humphrey Hardcover: 496 pages Publisher: Addison-Wesley; 1st edition (August 24, 1999) ISBN: 020147719X 4. UML Distilled: A Brief Guide to the Standard Object Modeling Language by Martin Fowler Publisher: Addison-Wesley Professional; 3 edition (September 19, 2003) ISBN: 0321193687 5. Learning UML 2.0 by Russ Miles, Kim Hamilton Paperback: 269 pages Publisher: O'Reilly Media (June 1, 2006) ISBN: 0596009828 6. UML 2.0 in a Nutshell by Dan Pilone, Neil Pitman Paperback: 216 pages Publisher: O'Reilly Media; 2 edition (June 1, 2005) ISBN: 0596007957 7. Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development by Craig Larman Hardcover: 736 pages Publisher: Prentice Hall Ptr; 3 edition (October 20, 2004) ISBN: 0131489062 8. Agile and Iterative Development: A Manager's Guide by Craig Larman Paperback: 368 pages

	Publisher: Addison-Wesley Professional; 1ST edition (August 15, 2003) ISBN: 0131111558 9. Design Patterns: Elements of Reusable Object-Oriented Software by Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides Hardcover: 395 pages Publisher: Addison-Wesley Professional; 1st edition (January 15, 1995) ISBN: 0201633612 10. Head First Design Patterns by Elisabeth Freeman, Eric Freeman, Bert Bates, Kathy Sierra Paperback: 638 pages Publisher: O'Reilly Media (October 25, 2004) ISBN: 0596007124 11. Head First Object-Oriented Analysis and Design by Brett D. McLaughlin, Gary Pollice, Dave West Paperback: 600 pages Publisher: O'Reilly Media (November 1, 2006) ISBN: 0596008678 12. Systems Analysis and Design with UML Version 2.0: An Object-Oriented Approach by Alan Dennis, Barbara Haley Wixom, David Tegarden Hardcover: 544 pages Publisher: Wiley; 2nd edition (August 10, 2004) ISBN: 0471348066 13. Software Architecture: Perspectives on an Emerging Discipline by Mary Shaw, David Garlan Paperback: 242 pages Publisher: Prentice Hall; 1 edition (April 2, 1996) ISBN: 0131829572 14. Code Complete by Steve McConnell Paperback: 960 pages Publisher: Microsoft Press; 2 edition (June 2004) ISBN: 0735619670
Attendance	Students are expected to attend all classes, and attendance will be taken. Please let the instructor know ahead of time if a class will be missed. More than three unexcused absences are grounds for failing the course. Excused absences may be due to personal illness, urgent family business, or work-related issues. The instructor reserves the right to require a reason or documentation as the basis for granting an excused absence. Class attendance and participation are expected and required.

Prerequisites	• Students must have a strong knowledge of at least one major programming language, such as C, C++, C#, Java, Python, or Perl. • Students should have completed a course in Data Structures and have completed at least one 500-level programming course. • Students must have a strong knowledge of either the Microsoft Windows XP/Vista/7, Linux, or Mac OS X operating systems. • Students must bring their laptop computers into class each week. • Students must be able to work in groups outside of class.
Description	In this course, students will learn the techniques for the construction of reliable, efficient, and cost-effective software. Topics covered include requirements analysis, software design, programming methodologies, testing procedures, software development tools, and management issues. Students will work in groups to plan a software application project, gather requirements, create an architecture, create a design, implement the code, test the final product, and finally present it to the entire class and to the instructor at the end of the semester.
Group Project	Students will design, implement, test, and demonstrate a group project, and present it to the class and to the instructor at the end of the semester. Some example projects include: a basic tax-preparation software like TurboTax; an online two-player chess game; an online DVD rental management system, such as used by Netflix; an airline reservation system such www.AA.com; an online car-rental reservation system such as Hertz.com or Avis.com; an online book purchasing system such as Amazon.com; a book/DVD management system for a public library. It is strongly suggested that the project contain either a GUI or web interface. Students are free to choose any programming language and operating system platform, such as C, C++, C#, Java, Python, or PHP under Windows XP, Windows Vista, Windows 7, Linux, or Mac OS X. Students are responsible for all hardware and software infrastructure needed by the project, and there is no financial support available to purchase any hardware, software, tools, or training. It would be very helpful if students could bring laptops into class for group work. Groups should be 3 or 4 or 5 people in size. Teams will select leaders. Specialization within groups is permitted, but all members must know all parts.
Project Assignments	**Guidelines for all the assignment documents:** • Be sure to always include page numbers. • It should look professional (presentation and format count), be logical in its organization, use correct spelling and grammar, cite all sources used, and have a cover page. • The cover page should contain only the course number (CS 673),

the assignment number, the assignment due date, the full names of all group members, and the name of the group.

- Each page should have the assignment number and the assignment title/name in the header.
- Each page should have the page (in the form "Page N of M") in the footer.
- Be sure to include a revision history at the beginning of your document. Keep track of the major changes you make to the document, incrementing the version number as people make changes. Start the version number at 0.1, and use 1.0 for the version that you submit.
- Have a sign-off section on the cover page requiring every team member to physically sign off on the document before submission.
- The document should be broken up into sections and sub-sections with section numbers, and it should contain a *Table of Contents* .
- Always submit only one monolithic PDF file; insert any and all figures, diagrams, drawings, etc. into your document; never submit more than one file for an assignment.
- Submit only by email. Do not submit a hard-copy in class.
- Submit assignments using only Adobe PDF (.pdf) format. Your documents may be developed using any format you wish, such as Microsoft Word 2000/XP/2003 (.doc), Microsoft Word 2007 (.docx), OpenOffice Writer (.odt), HTML/CSS (.html/.css), and LaTeX (.tex) formats, but you must use PDF format when submitting assignments.

Assignment 1: Software Team Communication Document (STCD) [Due Thursday January 21]

- Assign a name to your team.
- List, in detail, the technical backgrounds/skillsets of each team member.
- Describe what each team member wants to learn from participating in the team project, what the technical interests are for each team member. Also describe what are your most important concerns, as software engineers. Be as specific as possible.
- Describe in concrete terms how the team intends to communicate throughout the semester. When will team meetings be held?? What tools will you use to communicate?? (email, cell phone, IM, blog, Yahoo Groups, Google Groups, etc.)
- Describe, in detail, when, where, how often, and for how long

your team will meet. Specify a start time **and** an end time. You should meet at least once per week for one hour. You may find you need to meet for more time than that, or have more than one meeting per week.
- List the full name, primary email address, cell phone number, home location, and work location for each team members.
- Decide who your team leader will be. Being a team leader provides you with valuable experience, but puts additional demands on your time. Explain why this person was elected to the role. The team leader will: (1) create the agenda for the team meetings; (2) merge the team documents into final versions; (3) handle the submission of assignments to the instructor; (4) issue the tie-breaking vote if the team comes to an impasse.
- List all the tools and technologies that your team agrees to use to communicate and conduct work in. For example, does your team agree to use Microsoft Word for making the documents, or will you use OpenOffice, plain text files, HTML files, PDF files, LaTeX, etc.?? What about diagrams?? Will you use Microsoft Visio, or Graphviz, or Dia?? Explain why the team makes the choices it makes.

Assignment 2: Software Prototype/Proposal Document (SPD) [Due Thursday February 4]

- Write a proposal for your project, describing the project purpose, scope, major supported operations, input and output data, and implementation language and platform.
- Produce a simple prototype for your project that concentrates on addressing the risks that can be retired within three weeks.
- Describe what value, if any, the prototype provided in terms of these risks.
- Include screenshots and/or diagrams in your document.
- You are not required to submit any code.
- Estimate the total number of hours your team will spend on the project, starting with the team meeting in the first class, through the project presentation in the last class. Include a table like the one below that will be updated as the course progresses. **Submit a table like this with each homework.**

Deliverable	Estimated Hours	Actual Hours
STCD		
SPD		

SCMP		
SPMP		
SRS		
SDD		
Group Project Presentation		
Individual Project Report		

- Evaluate how much time (number of hours) that the team spent on this assignment. List the contributions of each team member and time spent on different parts of the homework. The following table is an example of what you should submit, but your particular list of tasks in the first column may be different. **Submit a table like this with each homework**, with the first column reflecting the actual parts of that homework.

Item	Person 1	Person 2	Person 3	Person 4	Total Hrs.
Meetings					
Communications Plan					
Application Investigation					
Prototype (coding)					
Prototype (drawings)					
Project Proposal					
Document Editing					
Document Review					
Total Hours					

-

Assignment 3: Software Configuration Management Plan (SCMP) [Due Thursday February 11]

- Produce a software configuration management plan for your project. You may use the SCMP template posted in the *Documents* section of the course website. It is an overview of the IEE 828-1998 specification and written by Professor Braude. Your document should be much more specific than the template. Alternatively, you may write your own template as

long as it covers all the areas covered by the IEEE specification.

- Avoid bottlenecks and unnecessary procedures.
- Explain the various CM processes your team will use.
- Explain the various CM roles. In this class project, there will probably be only one role, that of the CM manager. Explain the role of the CM manager in your project, and what responsibilities he/she will have. For which activities does a team member need the approval of the CM manager??
- Before you begin the review, estimate the number of substantive defects per document page that the team thinks it will discover, and make a note of this. Although there is currently no historical information, use the personal history of each team member and be as realistic as you can. Compare your estimate with the actual number of defects found per page after the document is finished and has been reviewed by the entire group.
- Define what your team will mean by "substantive defects per document page". Make this a practical, do-able, consistent measure, as you will be measuring defects found for subsequent homework assignments and you want the measures to be comparable across the project phases.
- Include a Post-Mortem at the end of your document. Use the template in the *Documents* section of the course website. Assess your team's effectiveness in each stage (e.g. team meeting, preparation, initial document, inspection, etc.) on a scale of 0 to 10. Summarize using the numerical results, and state how the team's process could have been improved. Be as specific as possible in describing process improvements.
- Evaluate the amount of time that the team spent on this stage of the project. List the contributions of each team member and time spent on different parts of the homework. (same as for Assignment 2; see the table above)
- Don't assign individual names to various roles in the SCMP. This should be done in the SPMP so that this information is in one location only.
- The SCMP should contain information relevant to configuration only. In particular, it should not stray into material that belongs in the SPMP.
- Evaluation criteria: (a) Practicality: How well does the plan ensure that documents and their versions will be secure, coordinated, and available? (b) Specifics: How specific is the plan in terms of suitably naming places and participants? (c) Process assessment and improvement: To what degree did the team understand the strengths and weaknesses of its process,

and how specific are its plans for improvement?
- Choose the license under which you will release your project's code upon the end of the semester. Is it purely proprietary? GPL? LGPL? BSD?

Assignment 4: Software Project Management Plan (SPMP) [Due Thursday February 25]

- Please use the SPMP template posted in the *Documents* section of the course website. You do not have to follow it exactly; it is just there as a guide/outline to help you.
- Develop a Software Project Management Plan for your project. Assume only one iteration of the Unified Process in the project schedule. Include somewhere a rough estimate of the size of the product in lines of code. Estimate the number of person-hours or person-days that will be required to complete the entire job. Explain how you arrived at this estimate. Take into account the approximate timing of various class assignments.
- Use SLOC, Functions Points, simple COCOMO, and intermediate COCOMO to create the estimates for your project. Explain your SLOC guess-estimates in detail. Explain how you decided the values for the inputs to calculate the function points. Compare the results from the simple COCOMO and the intermediate COCOMO.
- Be sure to include an accurate and detailed Gantt Chart with resolution of single days. The Gantt chart should begin at the start of the course and go to the project presentation on December 2. It should show all project activities and milestones, including testing.
- Include a section on testing which discusses your group's approach to testing and explains how, and by whom, each part of the software will be tested. Be sure to include testing in your group's Gantt Chart.
- You do not need to specify a separate person as the Configuration Manager; the Group Leader may serve in this position since we only have 12 weeks in this course. Whomever you chose, be sure to clearly explain your group's policies on using the tree and doing check-ins and check-outs.
- Be sure to include the **Team Contributions Table** (as we started doing in Assignment 2) for this assignment, as shown below. Evaluate how much time (number of hours to nearest 1/2 hour) that the team spent on this assignment. List the contributions of each team member and time spent on different parts of the homework. The following table is an example of

what you should submit, but your particular list of tasks in the first column may be different. **Be sure to submit a Team Contributions Table with each homework.**

Item	Person 1	Person 2	Person 3	Person 4	Total Hrs.
Meetings					
Communications Plan					
Application Investigation					
Prototype (coding)					
Prototype (drawings)					
Project Proposal					
Document Editing					
Document Review					
Total Hours					

- Be sure to include and update the **Effort Table** (as we started doing in Assignment 2) for this assignment, as shown below. **Be sure to submit the Effort Table with each assignment.**

Deliverable	Original Estimated Hours	Revised Estimated Hours	Actual Hours
STCD			
SPD			
SCMP			
SPMP			
SRS			
SDD			
Group Project Report			
Individual Project Report			

- Be sure to include a **detailed** Gantt Chart. The time resolution of the chart should be a single day, starting from the due date of this document, and running through the last day of the course. Tasks should be broken down from top-level/high-level tasks to specific low-level tasks, as appropriate. Assign specific

people to each specific task.

- Be sure to have a section that discusses risks and risk reduction in detail. Give specific target dates for risk reduction actions. Making everyone responsible for retiring a risk is not effective, and often means that no one is responsible. Someone specific should be named to follow and retire each risk. It may well be that this will involve everyone, but a single person is still required to see to it that the retirement process takes place and is completed.
- Provide a rough estimate of code size as requested. How many lines of code are projected for each section of the project, and for the total?
- You do not need to talk about Configuration Management in this assignment - ignore this section in the template. The previous assignment already talked about this topic.
- Explain how you arrived at your code size and time estimates based on function point analysis and COCOMO.
- In your project schedule, be sure to plan for the availability of the team through the semester. Allocate days for people who have upcoming vacations, weddings, business trips, personal issues, etc. List any time abscences for any team members.
- If you are not sure about something but can make an estimate, supply a range (e.g., "between April 3 and April 8" rather than just writing "TBD".
- Be sure to include the location (the URL) of your Google Code repository, or list the IP address of the computer with the code repository. Make sure it is is publically readable so I am able to take a look at your tree.

Assignment 5: Software Requirements Specification (SRS) [Due Thursday March 18]

- You may use the SRS template posted in the *Documents* section of the course website.
- Write the SRS for your project. Your document should include: (1) a thorough list of both functional and non-functional requirements (often in the form of "the software shall..." statements); (2) GUI sketches or screenshots (where appropriate); (3) four to six complete use-cases; (4) any applicable activity diagrams; (5) a complete, thorough, detailed class diagram; (6) any applicable state diagrams; and (7) any applicable sequence diagrams. Use proper UML notation, at least as was done in the course slides.
- Note that CRC cards and object diagrams are not necessary.

- Be sure to uniquely number each use-case, class diagram, functional and non-functional requirement, GUI screenshot, etc. Create your own numbering scheme and explain it in the document.
- Be sure to include a Post-Mortem at the end of your document, as we have done in previous homeworks. Use the template in the *Documents* section of the course website. Assess your team's effectiveness in each stage (e.g. team meeting, preparation, initial document, inspection, etc.) on a scale of 0 to 10. Summarize using the numerical results, and state how the team's process could have been improved. Be as specific as possible in describing process improvements.
- Do not forget to update your Gantt chart. Be sure to include an up-to-date Gantt chart in this document.
- Be sure to include the Team Contributions Table, the Effort Table, the Post-Mortem, and the up-to-date Gantt Chart in an appendix or final section.

Assignment 6: Software Design Document (SDD) [Due Thursday April 8]

- Write the SDD for your project. You may use a format of your choice, or the IEEE standard, which you may modify if you wish. Please also submit an updated SRS as well, if it has changed. This is required so that I can check your design against the requirements.
- Thoroughly explain the entire operation of your software in a clear, well-organized manner. Explain every aspect of the software, such as concurrency (multi-process and multi-thread), interfaces to databases, etc. Include activity diagrams, sequence diagrams, and architecture diagrams. Include state diagrams where appropriate/needed.
- Include a complete class diagram, showing all classes, methods, and attributes for your software. The diagram may span multiple pages if necessary. It should match your actual code exactly. Also include a table which alphabetically lists all your classes and provides a throrough description of each one.
- Include pseudocode where nevessary, or where it makes sense. Use pseudocode to explain, in clear English, complicated/intricate sections of code, or to explain the operation of an algorithm.
- Include a detailed list (or even a table) of all the dependencies (the OS itself, any frameworks used (Java, .Net, QT), third-party libraries used, and any client-side requirements such as

specific versions of IE/Firefox) that are used to develop, test, and run your software. Specify the dependency, its source (website, company name, etc.), the version number, and the license it is issued under. For example, do not just say you're "using Java"; explain that you're using "Sun Java JDK version 6, update 16, from http://java.sun.com/, issued under the GPL version 2 license." And same for the OS: do not just say you're developing and testing under "Windows XP"; explain that it is "Microsoft Windows XP, Professional Edition, Service Pack 3, 32-bit", or "Red Hat Enterprise (RHEL) Linux 5.4, 64-bit". If your software requires any specific patches/hotfixes to run, then specify those as well, and also explain why those patches are necessary.

- Include a list (or even a table) of what development tools you're using (Visual Studio, Eclipse, gcc/g++,), and their version numbers, and be specific. For example: Visual Studio 2008, Standard Edition, Service Pack 1.
- Evaluation criteria: clarity; completeness, sufficient to cover your requirements; thoroughness and technical detail and technical accuracy; elegance, expandability, scalability, maintainability; and meaningfulness of process improvement.
- Be sure to include a Post-Mortem at the end of your document, as we have done in previous homeworks. Use the template in the *Documents* section of the course website. Assess your team's effectiveness in each stage (e.g. team meeting, preparation, initial document, inspection, etc.) on a scale of 0 to 10. Summarize using the numerical results, and state how the team's process could have been improved. Be as specific as possible in describing process improvements.
- Do not forget to update your Gantt chart. Be sure to include an up-to-date Gantt chart in this document. Make sure that it is large enough to read and to see.
- Be sure to include the Team Contributions Table, the Effort Table, the Post-Mortem, and the up-to-date Gantt Chart in an appendix or final section.

Assignment 7: Code Reviews [Conducted Thursday April 15 / Write-Up Due Thursday April 29]

- Each group will: (1) perform a code review of another group's code segment; and (2) have a segment of their own code reviewed by another group.
- On April 8, each group will be paired with one other group.
- On April 9, each group will send the group that they are paired

with the code segments that they desire to be reviewed, along with the group's SRS and SDD documents. Make sure your SRS and SDD documents are up-to-date. The code segment you send to the other group should be enough for the entire group to review during the week. Use your judgement. Not too much, not too little. Your group should receive similar information from the group you are paired with. Review their code, and compile a detailed list of defects in preparation for your in-class code review meeting.

- In addition, each group should also peer-review their own code segments, and create a list of issues to bring up and discuss for their own code segments. In other words, the group you are paired with will review your code, but also include comments and defects raised by members of your own team.
- Every group member should serve as a reviewer/inspector for both their own group's code segments as well as for the other group's code segments. All group members from both teams are expected to participate in both groups' code review.
- Each group member should come to the meeting prepared. Read/Study the other group's code segments ahead of time, and make your personal list of issues to be discussed in the meeting.
- The code segments that are chosen for review may be from one author, or may be from multiple authors. It may be from one file or from one class, or it may be from multiple files, or from multiple classes. Use your judgement in choosing the appropriate code segments to be reviewed. Pick interesting code segments, not simple code segments that may not need to be reviewed as much as more complicated/intricate segments.
- Each code review should take about 60 minutes. Therefore, it should take 120 minutes for the two paired groups to complete the code reviews for each other.
- Each group leader will serve as the code review facilitator and the scribe (note-taker). This person will capture all the issues raised in the review of their group's code segments. They will compile this list into the write-up for this assignment. The list should contain the following items for each issue raised: the unique issue identification number; the person who raised the issue (either from your team or the other team); the owner of the issue; the severity of the issue (from 1 to 10, or high-medium-low); the category of the issue (explain what each category means); a detailed description of the issue; the file(s) and line number(s) and class(es) affected; the course of action to be taken for the issue; and the due date for it to be resolved

by. This detailed list is this assignment.
- The code reviews are conducted on April 15, and the write-up (Assignment 7) is due on the following class, April 29, which is the same day as your final project presentations.

Assignment 8: Group Project Presentation [Due Thursday April 29]

- Present your group project to the class and to the instructor. Presentations should be about 45 minutes long, including 5 to 10 minutes for questions.
- The presentation should consist of slides in either Microsoft PowerPoint or Adobe PDF format. The presentation should also include a working demonstration of your software. The presentation should be conducted via your laptop. A projector will be provided in the room.
- Each team member should present something, and each team member should speak at least once.
- Each team member should be prepared to answer any questions.
- The instructor should not be the only person in the room asking questions after each group's presentation; everyone in the audience is expected to participate.
- Be sure to present the final Team Contributions Table and the Effort Table.
- The presentation should include a running demonstration of your project, a discussion of the requirements, a demonstration of each major requirement from your SRS, a discussion of the design and the implementation, and the testing process and testing infrastructure.
- Discuss how the original Gantt chart and your original project estimates for task times, code size, and code quality compare with the final results. Discuss your final Gantt chart.
- Show the structure of your Google Code or Subversion tree. Show the trace-ability from your design and requirements documentation. Explain how you decided on each of the major requirements. How did it impact your design? Why did you make certain design choices?
- It should be straightforward to trace from your requirements to your implementation.
- Discuss what you learned from the code inspection process.
- Explain what worked well and what didn't work during the project (team communications, with the requirements, the design, the implementation, and/or the process), and what

specifically you would do differently, both individually and as a team.

- Example outline:
  - Introduction (3 min)
    - Names and backgrounds of all team members
    - Name of team
  - Project overview (5 min)
    - Describe problem description, describe major requirements from SRS, describe what your project does in detail
    - Why do this project?? Why is it interesting?? What do you do differently??
    - Solution description (overview of what you implemented)
    - Why and How did you chose your platform, your environment, your language(s), your tools, etc.??
  - Demonstration (8 min)
    - Real-time walkthrough of application using laptop and projector
    - Demonstration of all GUI features and major requirements
  - Design (8 min)
    - Present overview of architecture and design
    - Describe why you chose the design, language, environment, etc. that you used
    - Provide enough detail for others to understand, but don't get too detailed
    - Describe the issues you ran across, and the changes you made along the way
    - Overview of your implementation standards and your Google Code/Subversion tree structure
  - Group Process (6 min)
    - Describe overall group project process: your development process; your configuration management; meeting logistics; etc.
    - Present project schedule (Gantt chart) (Be sure to discuss your original estimates and the final outcome)
  - Project Post-Mortem (7 min)
    - Project successes (what are you proud of??)
    - Specific process problems, and specific quantifiable improvements
    - Anything else that didn't go well in the project
    - What would you differently next time??
    - What, in particular, did you learn from the code

inspection??
        - What, in particular, did you learn on this project??
    - Question and Answer (5-10 min)
        - Questions from class members and from the instructor

Assignment 9: Individual Project Report [Due Thursday April 29]

- The goal of the individual project assessments is to ensure that everyone in your group has made a significant and relatively equal contribution, and if not, that I am made aware of this. I don't expect the contributions to be exactly equal, but each member should contribute in a meaningful way to the project's success.
- This assignment is to be done by each person individually, not as a group. Each individual will submit their own document for this assignment. I do not share this assignment with your group; this assignment is private between the student and the instructor.
- Give your background - describe your level of software engineering sophistication upon entering the course. Explain the major points/lessons learned; state the most important specific issues in software engineering that you learned from participating in your group project. You should list at least three issues. Each should be backed up by specific examples from your experience. Experiences should include positive and negative ones. Please avoid repeating points in the textbook or lecture unless you can back them up with specific or unique experience in the team project. Back up your assertions with specifics and quantitative data. Please do not include aspects that you already understood upon entering the course. Discuss the positive things as well as the negative things. Be clear, concise, and specifc in your writing.
- Include your team peer review for each of your fellow teammates. Do you think everyone in your group made a significant contribution and deserves an equal grade? If not, which member(s) do you think were deficient, and why? Which member(s) did significantly more, and why?
- Provide a table with your team peer review that includes the names of all your teammates, including yourself. Allocate ((number_of_teammates) * 10) points over all the people in your team, including yourself. Be fair, honest, and un-biased when grading yourself.

<table>
<tr>
<td>Final Exam</td>
<td>

This course will have one comprehensive exam
The exam will be a take-home exam which will be posted on the course website on Thursday April 29 at 11pm, and will be due by Sunday May 2 at 6pm.
Students may use any textbook, the course lecture notes, course slides, course assignments, and any printed materials.
Students must work alone and are not allowed to communicate with anyone.
The exam must be submitted in one PDF file. Any figures, diagrams, etc. must be incorporated into the final PDF file. Follow all the homework guidelines (use a title page, use the correct file name, etc.)

The following is a list of the types of questions on the exam:

writing use-cases, class diagrams, sequence diagrams for a specific case study
evaluating a design and/or code fragments, relating to a specific case study
questions about software process applied to various situations
questions about inspections and developing/evaluating project schedules and costs
creating, evaluating, scheduling, and costing a project for a specific case study
Function points and COCOMO for project estimation
Testing, equivalence classes, black-box and white-box testing
GoF design patterns (Singleton and Observer patterns)

</td>
</tr>
<tr>
<td>Grading</td>
<td>
<table>
<tr><td>Assignment 1 - Software Team Communication Document (STCD)</td><td>5%</td></tr>
<tr><td>Assignment 2 - Software Prototype Document (SPD)</td><td>5%</td></tr>
<tr><td>Assignment 3 - Software Configuration Management Plan (SCMP)</td><td>5%</td></tr>
<tr><td>Assignment 4 - Software Project Management Plan (SPMP)</td><td>5%</td></tr>
<tr><td>Assignment 5 - Software Requirements Specification (SRS)</td><td>10%</td></tr>
<tr><td>Assignment 6 - Software Design Document (SDD)</td><td>10%</td></tr>
</table>
</td>
</tr>
</table>

	<table><tr><td>Assignment 7 - Code Reviews</td><td>10%</td></tr><tr><td>Assignment 8 - Group Project Presentation</td><td>20%</td></tr><tr><td>Assignment 9 - Individual Project Report</td><td>5%</td></tr><tr><td>Final Exam</td><td>25%</td></tr></table> The course consists of one in-class final exam as well as nine assignments which represent various steps in the software development process of the group project. All members of the group will generally receive the same grade for each assignment, but one scaled by peer evaluations within the group. In peer evaluations, each group member assesses the relative contribution to the group's final grade made by each of the other team members. Be honest, fair, and rational. If your group has N participants, then apportion (10 * (N-1)) points among the team members, excluding yourself. For example, the members of a team consisting of A, B, C, and D could be evaluated by member B as A=8, C=12, and D=10. The extent of this factoring will be decided by the instructor near the end of the course. Each team will also perform a 30-minute code inspection with the instructor on April 19. Late submissions will not be accepted without an appropriate explanation or prior arrangement. One project grade is given to all members of a group. However, individual project grades can be adjusted by the instructor based on the peer evaluations (Assignment 9). As part of each project submission, group members will email the instructor with a relative contribution for each group member. The extent of factoring will be decided by the instructor near the end of the course.
Outline: Thursday January 14	• Create Project Groups • Version Control and Code Repositories (SourceForge, Google Code, etc.)
Outline: Thursday January 21	• Lecture 1 (Introduction) • Lecture 2 (Software Process)

Outline: Thursday January 28	• Class canceled today due to instructor travel
Outline: Thursday February 4	• Assignment 1 (STCD) Due • Assignment 2 (SPD) Due • Finish Lecture 2 (Software Process) • The Capability Maturity Model (CMM) • Finalize groups
Outline: Thursday February 11	• Assignment 3 (SCMP) Due • Software development risk and risk management • Lecture 3 (Project Management)
Outline: Thursday February 18	• Show examples of COCOMO and Function Points • Revision Control Software (CVS, Subversion, Git, etc.) • Software Licenses (GPLv2, GPLv3, LGPL, BSD, MIT, Apache, etc.) • Start Lecture 4 (Requirements Analysis 1)
Outline: Thursday February 25	• Assignment 4 (SPMP) Due • Finish Lecture 4 (Requirements Analysis 1) • Lecture 5 (Requirements Analysis 2)
Outline: Thursday March 4	• Start Lecture 6 (Software Design)
Outline: Thursday March 11	• Make-Up Class (Regardless of Spring Break) • Finish Lecture 6 (Software Design)
Outline: Thursday March 18	• Assignment 5 (SRS) Due • Lecture 7 (Implementation)
Outline: Thursday March 25	• Start Lecture 8 (Software Testing)
Outline: Thursday April 1	• Finish Lecture 8 (Software Testing)
Outline: Thursday April 8	• Assignment 6 (SDD) Due • Lecture 9 (Maintenance) • Assign team pairs for code review

	• Teams exchange code, SRS document, SDD document for code review
Outline: Thursday April 15	• Each team brings two lists of issues (for their own team's code and for the other team's code) into class • Assignment 7 (Code Inspection) Conducted in-class
Outline: Thursday April 22	• No class today, per BU academic calendar
Outline: Thursday April 29	• Order Pizza in Class • Assignment 7 (Code Inspection) Write-Up Due • Assignment 8 (Group Project Presentations) Conducted in-class • Assignment 9 (Individual Project Report) Due • Final Exam Posted on-line at 11pm
Outline: Sunday May 2	• Final Exam Due by 6pm