Computer Science B665
Software Engineering Management

Section No	8262	*Instructor:*	Edward Robertson
Room:	LH 401D	*Office:*	Lindley Hall 401D
Time:	13:30 TR	*Phone:*	855-4954
		e-mail:	edrbtsn@cs.indiana.edu

Course Description

B665 Software Engineering Management (3 cr.) P: P465, P565, or consent of instructor Seminar in software engineering and management. Readings and presentations of classic works and current papers in the field. Topics include: the programming activity, program products, software life cycle, programming teams and supervising of programming projects. Actual supervision of one or more programming teams producing software products.

Texts

Roger Pressman, *Software Engineering: A Practitioner's Approach* (recent edition). McGraw–Hill.

Course Responsibilities

There will be three categories of course responsibilities:

a. Oral presentations and class participation.
b. Writing a paper.
c. Project supervision.

Oral presentations and class participation

During the semester each participant is expected to present at least one paper and contribute to the discussion of other papers. The presentation of a paper will require careful reading of the paper and involve three major activities:

1 choosing, for the preceding class, the article and posting on the class news group, significant questions concerning the article at least one day in advance of the presentation,
2 initiating class discussion, usually by giving a synopsis of the significant points and a critiquing its content and presentation, and
3 leading the subsequent discussion on the article.

In preparing the critique you might consider what ideas in the article are

a. significant and well-presented,
b. possibly significant but obscured by poor presentation,
c. valid in some contexts but not in others (especially where the paper does not adequately explore the context issue),
d. contain the germ of something significant but require further thought, or
e. simply bogus.

In particular, you should ask "what could be done to improve the article."

Students who are not presenting the day's paper are expected to read that paper, although perhaps not in as much detail as the presenter, to read and consider the questions on the news file, and participate in the class discussion.

Writing a Paper

You are required to write an annotated bibliography of some topic in software engineering. Under special circumstances, another form of paper can be negotiated. Your paper must be well-written and meet standard stylistic criteria for a good technical paper. To achieve this goal, your writing task will consist of three phases:

1. An outline or single page description of the content and format of the paper.
2. A first draft, which provides feedback without grading.
3. A final draft for grading.

An annotated bibliography is more than just a list of a few articles with a sentence or two about each. It is, on the other hand, less than a survey article of an area. It is first a discussion of a subject area, classifying the major issues, conceptual approaches, and methodologies. Next, the articles of the bibliography are categorized according to the preceeding classification and their major results and conclusions are described. Fianlly, it *may* be possible to discern holes in the coverage of the area and thus suggest questions that deserve further investigation.

A bibliography naturally begins with a literature search. The literature search should be conducted by more than just thumbing through a few recent journals. In particular, use the *Computer Science Citation Index* or the online versions `webofscience.com` and `citeseer.com` to explore the subject area.

Thus the goals of this assignment are

A. Learn how to navigate through the literture of a subject area and
B. Learn how to analyze a subject area for dominant themes and open questions.

A *very rough* outline is: Introduction. In addition to a "10,000 foot overview" of the subejct area, devote aparagraph to describing how any why you selected the papers covered (that is, the search process, not the results). Characteristics of the Subject area: Try to identify major indpendent "dimensions" which may be used to classify papers in the bibliography. One expects that a well-written paper will hold some dimensions constant and explore variations in others. For example, dimensions related to testing might be goals (functionality, robustness, interface & usability, $\cdots$), phase (unit, integration, $\cdots$), design mode (glass box, black box, $\cdots$), process & implementation (planning, infrastructure (scaffolding, stubs), test evaluation, *cdots*)

Project Supervision

Each student in this course will be assigned the role of supervisor for one or more teams of students from the Information Systems class and, later, the role of Quality Assurance Monitor for other teams.

As Supervisor, you will give guidance and advice to the teams and facilitate communication between the teams and the instructors. Following sections of this syllabus, Team Management and Milestone Responsibilities, discuss Project Supervision in greater

detail. The first covers personnel issues. The second gives a short guide to the specific responsibilities associated with each milestone.

As Quality Assurance Monitor (QAM), your primary responsibility is to ensure that your team designs an adequate process and follows that process. In particular, you will evaluate the Test Plan. A short section on the QAM role, which is most relevant during the second semester (B666), also follows.

Grading

The grading of these three categories of responsibilities will be weighted approximately as follows:

Part	*Approximate Weight* ($\pm 10\%$)
Paper	25%
Presentation and class participation	25%
Group supervision	50%

Team Management

Your team management responsibilities are both formal and informal. The formal side involves a process of meeting with the teams on a weekly basis and tracking their progress through their weekly Team Log submissions. On the informal side, you will give guidance and advice, review outlines and drafts, cajole and occasionally threaten team members, and generally have a very rewarding experience.

Formal Management: the Team Log

Since the Team Log is so essential to the management process, the instructions given in the Information Systems *Information Packet* concerning the log are repeated here:

> The team must maintain a Team Log for the life of the project. The Team Log must record all important team decisions, task assignments (such as 'Joe agreed to provide a debugged copy of module A by Feb 15'), task completions and other task status facts, and design or process changes. All this information must be recorded in a timely manner. The Log should also record role assignments, discussed below, and may also include notes from team meetings.
>
> The Team Log must be a "write-once" document. Originally the Team Log was a permanently bound volume (hence we still refer to the "Log Book"), but now it is more common to use electronic media. But even though the log is kept in a computer file, it is important that it retain its "write-once" nature. When the log was kept on paper, the "write-once" characteristic was enforced simply by keeping the log in ink, not allowing erasures. In a computer file, it can be enforced by using a file that is append-only. The reason for these constraints is that we wish to require that changes in the log be explicit. Just as a bank reflects an error in a deposit by adding a new journal entry rather than changing the original deposit, a correction or change in a log requires noting the incorrect statement and presenting the correct one.
>
> In order to insure the "write once" characteristic, each team is required to email their log entry to their Supervisor each week. It is required that the team have one formal meeting a week during which process matters are dealt with – in particular, tasks are assigned and task status is reviewed. These process matters are the heart of the log entry.

You are asked to use a template for your log entry. This is easier for you because it indicates exactly what is required. This is easier for the Supervisors and Instructors because we know exactly what to look for. Templates are available in the files `team-log-template.*` in the course's NFS repository. The team secretary should simply edit this file and submit it by email each week. In fact, it is a bit easier to start with a copy of last week's log entry, since much of it carries forward.

Most of the fields in the template are self-evident, but some deserve a few words of explanation:

- `Recording Date` The date on which the secretary edited the log entry.
- `Due Date` The date given in this packet on which the milestone is due (see the Project Schedule on the following page).
- `Group Target Date` The team should set their own internal due date a few days before the milestone is to be turned in. Record that date here.
- `Deliverables Due` A list of those items to be turned in.

In certain circumstances (*e.g.* when working on the Prototype and Preliminary Design), you may need to have two "Current Milestone" entries.

At each meeting when new task assignments are made, those assignments should be logged where indicated. The next week, these assignments should be move down to the next area, as carry-forward assignments. Tasks remain as carry-forward until after they have been marked "completed" in the status field. A completed task is then removed from the next log entry.

To reiterate, the purpose of the log is to record in a consistent manner and place: what tasks must be done, when they must be begun and finished, who is responsible for accomplishing them, and where they stand with respect to their completion.

There are many reasons to keep a Team Log. Most of all, it allows you to really keep track of what is going on in the team. It also provides a entry for intervention if necessary. Finally, it is valuable when evaluating team members at the end of the semester.

There may be resistance from the team in reporting information to be logged, but you should tell them this information is required. More importantly, the team needs the discipline of task assignment and monitoring independently of our need to track the project's progress.

Informal Management

In the informal areas of helping your teams, your own common sense and your experiences in P565/6 (and team professional experiences) are more important than any guidelines. If you had a great supervisor, you should try to emulate what that person did well. If you had a poor supervisor, you should try do improve on those areas where your supervisor performed poorly. P465[†] and P466 are very difficult courses - mostly because they are crowded with work and have expectations that are hard to meet. We are asking students to do well things they have not heard of prior to P465. It is your job to make things smoother and easier for the teams. You should keep in touch with them and watch actively for difficulties. You should not do the work for the team - *i.e.* actual programming or design. That is not your function, although you may criticize, consult,

[†] Here and subsequently, "P465" will refer to both P465 and P565; and "P466" will similarly refer to both P466 and P566

and provide general direction. Your job is to give the team every chance to succeed on its own efforts.

With these general principles in mind, there are some suggestions (mainly procedural ones) which are likely to facilitate working with your team. A key factor in several of these is the *delegation* of certain organizational responsibilities. The person to whom each of these responsibilities is delegated may be chosen by you, elected by the team, or other means as appropriate.

- Meet with your team on a **regular** basis. Once a week is often enough, but the meeting should be at a fixed time and place.
- Establish and use e-mail communication.
- Have the team select or you appoint a *convener*, who will be responsible for communication with other team members. such as notification of extraordinary meetings.
- Select/appoint a *recorder* or *secretary* who will keep track of internal team communications. The recorder should keep an archive of e-mail, maintain shared file systems, *etc.*
- Select/appoint a *client contact*, who is the primary spokesperson for the team when dealing with the client. The client contact should of course have good rapport with the client, but it is equally important that the contact have the respect of and respect for the rest of the team, since any commitments by the contact must be carried out by the entire group. It is very important that the team speaks "with one voice" to avoid confusing the client, and this is insured if indeed one physical voice does the speaking for the group.

Team Evaluation

At the end of the semester, you must evaluate the members of the team (or teams) that you supervise. Most importantly, you should write a description of the performance of each team as a whole and describe the participation and contributions of each team member. In order to help you in this evaluation, you will be given a form which structures the analysis of the contribution of each team member.

The Team Log, if properly kept, should be a great assistance in doing the team evaluations.

Personnel Issues/Team Discipline

Each year there are a few Information Systems students who require some additional prodding to encourage their team participation. Each of these cases is of course unique and requires individual handling, but following steps are a commonly used gradation of severity in disciplining a recalcitrant team member. They represent the "bureaucratic" side of the process and are usually best coupled with face-to-face discussions.

1. Email messages to the student from you.
2. Email messages to the student cc'ed to the P465 instructor. The instructor will not act further on these messages without your request.
3. Requesting that the P465 instructor step in with email to the student. This may be coupled with a face-to-face meeting.

4. A warning letter to the student, delivered on department letterhead. This describes the result of being dropped from the project team.
5. Actual removal from the project team. The P465 Syllabus does describe this as a possible consequence. However, this action has rarely been necessary.

Each of the above steps takes time. If you wait to long before taking action, the team as a whole may be in trouble because of missed deadlines. Therefore it is essential to keep on top of team performance and to address problems as soon as they are apparent.

Quality Assurance Monitor

The QAM is supposed to ensure the quality of both the *product* and the *process*. The product/process distinction is in general hard for some students to understand, but it seems particularly difficult where quality issues are concerned. Perhaps this is due to the fact that, while students have had experience designing and writing programs, they have had little or no experience with rigorous testing, change control, *etc*.

The primary activity of the QAM is to evaluate the Test Plan (the testing aspects of the Requirements Analysis and the Designs) and ensure that the Test Plan is followed during the Testing phase. Some teams go through this quite easily; others require substantial assistance with the overall test process. In the later case, try to keep your help focused on the process rather than helping them test. However, you should try a few of the tests in their test suite; do this twice, just after they logged a few tests completed and at the very end.

The secondary activity of the QAM is to monitor other process components, particularly those such as the change log that are part of a "quality process". The real reason for doing this is that the teams will pay attention to these matters only if we, the "project management", are seen to be paying attention too. Thus, you should invest only as much of your own or the teams' time on monitoring as is needed to achieve a reasonable, not a perfect, process. Note that this kind of monitoring reflects an important general management skill: often a manager only sits back and observes when things are going well, calls attention to problems as they arise, and becomes actively involved only when things need fixing.

Milestone Responsibilities

The purpose of this section is to give you some explicit advice about what a supervisor must do concerning project milestones. It also gives you some helpful hints.

Milestone 0: Project Proposal

During this first round, you get a bye. Since projects have not been assigned (they don't even exist), you do not have any direct responsibilities. However, you may give help if asked by students who know you are a supervisor.

Milestone 1: Feasibility Study

The purpose of this document is to evaluate (1) whether the project is of the right size and focus for the course and (2) whether the environment has any pitfalls which would impede the project. The judging focus is of course easy - an information system - while judging the size takes substantial experience. The judgment on the environment has many facets, which are itemized in *Running the River*, the text for P465/6. What the team must do is to provide sufficient information for making these judgments.

In order to provide the information, it is necessary for the students to discover what the client "wants." Of course, it is highly unlikely that the client knows this. So from here all through the design process (and unfortunately through the development phases), you must press your teams to carefully consider what they hear from the client. Point out ways to discover more information. Draw into question any parts of the document that seem not well founded or are suspect on the basis of your own experience.

You should from the beginning try to evaluate the project yourself. Is it a good project for P465/6? Is it too large? Too small? Does the client have the resources (and the will) to back the team? Is the team able to focus on the requirements for the milestone? Be alert – the sooner you spot trouble the less it costs. This is one of the most important lessons of software engineering: the later we discover a serious problem, the less our chances of recovery.

Draft E-R Model

The E-R Model is so critical to the project specification that we require an extra iteration in its development. Each team is required to hand in a draft E-R model in conjunction with assignment 3. We review these drafts as rapidly as possible in order that the team may incorporate our suggestions into the next milestone document.

It is important to remember that these are *draft* E-R diagrams. If the team has questions about the model, those should be noted on the draft. If there are issues about how to model some aspect of the clients "world," the team could submit two (or more) possible alternative diagrams.

The supervisors may not need do much at this stage, other than to encourage their teams to approach E-R modeling from the right perspective. It is particularly important that teams do modeling, not implementation design.

Milestone 2: Requirements Specification

This is a crucial software engineering document – probably the second most important. If we do not understand the problem fully, then our chances of a complete and satisfactory solution are slim. The basic problem for you here is that the students do not understand this document. It justifies the rest of the work they do by giving a reason and target for the design elements. It also sets operational standards to be met and tested for.

This document once again (now more fully) explains what the user wants. Its main purpose is communication between the team and client. In effect it is a contract about the project. The specification should be written so that the client is the audience. It should be very, very clear and accessible to the client. In other words, it should be written in the language of the client instead of computer jargon. Try to also get the team to have the client REALLY review and approve the document.

As with all the documents, be sure that what is handed in matches the requirements in the P465 student's information packet. Save the P465 AIs some trouble by reviewing and rejecting grossly inadequate attempts. Also this is our last chance probably to spot killer problems, for example, to detect a client without the proper equipment or software.

Try to get the team to think about the *Preliminary Project Plan* and write something on it. This is the dim reflection in P465 of the most critical software engineering document, the Project Management Plan. If it is well thought out and executed then the project can weather most storms. In the past, this part of the document has been short-changed. That is a great shame.

The Quality Assurance Plan (QAP) is required at this stage mostly for pedagogic reasons. Therefore the students are provided with a "boilerplate" draft QAP which requires minimal modifications. Help your teams understand that this is not a mere formality – it is very much "real world" in better programming shops.

Milestone 3: Prototype

There are three distinct reasons for prototyping at this stage in the course: (1) it should give the team experience with the software tools they will use to develop the project, (2) it allows the team to experiment with various design issues, and (3) it helps users focus on form and function of the system while there is still time to modify the specifications. Unfortunately the first reason may not apply to teams who do not yet have access to the software platform the will finally use in the project. But every effort should be made to have the teams use the actual platform of their project – they need that experience.

Supervisors should encourage students to develop enough of the prototype to get a true picture of both the function of the proposed design and the potential problems of implementing that design. Also, supervisors make sure that teams give their clients a complete demonstration of the prototype.

Milestone 4: Preliminary Design

This is the culminating document for P465. Its purpose is to give the architecture for the system addressing the client's needs. It must match the needs that appeared in the Requirements Specification. You should ensure that all the significant requirements of the project are covered by the Preliminary Design. We strongly recommend that each team hold a semi-formal review of the Preliminary Design draft. Go through both the Requirements Specification and the draft, looking for holes or mismatches.

Check to see that operational requirements are met. Review carefully the design of the user interface. Make sure it makes sense and is well layed out. Watch for lack of uniformity and silly glitches. Something just thrown together will be seen for what it is right away.

Once again this document must be understandable by the client. It shows what the team intends as a solution to the client's problem. The client should be able to see the connections to the requirements and see that there is a solution. If you can't see it, then it is not very likely the client will either. Once again, try to get the team to get the client to read and review the document.

Concerning the test plan: it should allow for a reasonable schedule of tests. Each of the operational requirements should be tested so they should be clearly represented in the test plan. Good requirements are testable. However, emphasize that this is the plan and

not the actual tests. The team should learn, at this point, that they must contribute to the process definition, not merely follow a pre-defined recipe.

Semester Presentation

Help them make a clear and concise report. The goal here is to have it be as professional as possible. Have the team imagine that they are working at some big company and that their presentation is in front of their management. Management personnel are sharp, but not fully knowledgeable about the team's work (just as the class will be). The goal is to make us understand the problem and the preliminary design solution. Good graphics, organization and simple clear speech are a must. It is probably best to avoid weirdness – the team should want to attract attention to itself by the excellence of the job, not by tricks.

Milestone 5: Detailed Design

This is a second chance to get the design right. It is also the last chance!

Milestone 6: Implementation

If everything has gone as hoped, your role at this point should be one of monitoring and giving advice. If there are problems, however, these problems may be agonizing.

Milestone 7: Testing

This is the Milestone where the Quality Assurance Monitor becomes a critical player. It is not expected that you, as QAM, review all tests in details. You should spot check the tests; if the cases you appear adequate, you may assume they are representative of the entire test plan.

Milestone 8: Installation

There are two distinct verifications of the final system. One is a final demonstration of the system to the supervisor, QAM, and instructors; it is desirable but not required that this occur on the final platform. The second is a sign-off by the client acknowledging that the system (software, documentation, *etc.*) has been delivered and installed.

General Advice

A few "rules of thumb" from previous years:

1. Help teams schedule (and work on!) tasks in parallel. They are likely to try to do one task at a time and miss the lead time (or "gestation time"). Perhaps the most significant place to watch for this is between the Requirements Specification and the Prototype.
2. If you find bugs as QAM, the most significant question for you is not why the system failed but why the testing failed.

   If the team claims failure due to a flaw in a tool, ask whether they have sent a bug report to the tool's vendor.

Project Schedule

The following project schedule table and paragraph describing deadline conditions are quoted directly from the P465/6 Information Packet.

	Date Due	*Supporting Reading*	*Description Goals and Activities*
Milestone 0	Sept 16	R I-IV; My 1; St 10	Project Proposal. Life cycle, program products, reliability. Interviewing.
Milestone 1	Sept 30	R V-VI; BH 1,2, Appx A; St 12; Pr 11.2	Feasibility Study. Systems analysis. Software management, Benefit/cost analysis. Team organization.
Requirements Reviews	Oct 28-Oct 31	RVG; BCN; Ram 2	Walkthrough of Requirements Specification. Draft E-R model. Peer feedback on requirements.
Milestone 2	Oct 31†, Nov 6‡	R VII-VIII; My 2; RVG; KS 2; Ram 15; St 11, 13	Requirements Specification, Project Plan, Quality Assurance Plan. Data flow diagrams, E-R model. Project scheduling and management.
Milestone 3	Nov 18	My 3; Sv 6; St 9	Prototype. User interfaces. Software evaluation. Tool experience.
Milestone 4	Nov 25†, Dec 4‡	R IX; BH 5: My 3, 4, 7, 8.2; St 15	Preliminary Design and Coding Standards. File organization. Pseudo-code, HIPO, Warnier-Orr diagrams. The user interface, report design. Module, interface, and documentation standards.
Presentations	Dec 4-Dec 11		Project Presentations. Project goals and general approach toward achieving these goals. Specifications, ER model, and user interface. *Client oriented.*
Design Reviews	Jan 20-Jan 30	Sv 14; R X	Design Walkthroughs. Peer feedback on designs.
Milestone 5	Feb 3†, Feb 5‡	My 5; R XI	Detailed Design and Test Plans. Software specification and design. System flowcharts. Relational design issues.
Milestone 6	Mar 3	My 6; Pr 12	Implementation. Programming style, readability. Documentation. Code walkthroughs. Unit testing.
Milestone 7	Mar 31	R XI;My 7; BH 10,11	Testing. Integration testing. Debugging.
Milestone 8	Apr 14†, Apr 21‡	R XII	Installation and Documentation. Post mortem.

† – draft due to supervisor. ‡ – final copy due to instructors.