DESCRIPTION: This course explores classical and emerging techniques for managing software development projects. Empirical and theoretical foundations for development lifecycle model selection, project estimation, risk management will be treated through textbook and research papers. Lifecycle models include classical waterfall, the spiral model, iterative/incremental approaches like extreme programming and agile software development. Students will investigate modern challenges of outsourcing, open software, and geo-distributed teams. The student will learn to apply technical management methods and tools for project estimation, risk management, project control and status reporting, with a particular emphasis on the use of quantitative methods. Instructional methods include case studies, review of classical and current literature, exposure to representative industry practices, and group assignments culminating in group discussions and presentation.

TEXTS:

REFERENCES:

RESEARCH PAPERS: A reading list will be provided.

OBJECTIVES: The purpose of this course is six-fold: (1) to expose students to the breadth of the subject; (2) to give students experiences performing a variety of software management tasks; (3) to give students experience using automated project management tools; (4) to introduce students to current trends and research in software process and project management; and (5) to expose students to industry practice and career opportunities in project management; (6) to give students experience using software management tools.

- To understand the technical activities inherent in a software development project.
- To gain a historical perspective on the evolution of software development process models
- To develop skills for selecting a management strategy that best fits the project, project staff, and customer.
- To provide experience using automated tools for project management activities.
- To expose the student to classical and current literature in project management.
- To provide the student opportunities to conduct research in empirical software engineering.
TEACHING APPROACH: The following types of activities are used.

- **Lectures** taken primarily from the reference text books, and from research papers.
- **Research papers.** Assigned research papers/projects with in-class discussions.
- **Individual projects.** Short term assignments – homework, reflections/reviews on (of) readings.
- **Group assignments.** Students will work on teams of 3-5 to perform tasks such as the following:
  - *Project Estimation* – use software cost models to develop labor, staffing and cost estimates for a development project.
  - *Project Planning* – develop a project plan containing: project requirements, work breakdown structure, staffing, task list, activity network (Gantt chart), milestone list, burn-rate charts (planned spending).
  - *Risk Analysis* – perform a risk analysis of a case study or an assigned project: prioritized list of risks; risk management plan, etc.
  - *Management Review* – prepare project status reports (with respect to project plan) and conduct a project status review.
- **Student presentations.** A portion of each unit is used for technical/management presentations by student teams. Presentations may focus on case studies, papers, or other group tasks assigned by the instructor.
- **Case studies.** Students will use published case studies to understand and apply management techniques.
- **Examinations.** A mid-term and final exam will be given. One or more may be take-home or computer based.
- **Reflection.** A course lessons learned experience report or research paper.

GRADING:

<table>
<thead>
<tr>
<th>Percentages:</th>
<th>30-50% Exams (2)</th>
<th>30-50% Projects</th>
<th>10-20% Research</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scale:</td>
<td>A = 90-100</td>
<td>B = 80-89</td>
<td>C = 70-79</td>
</tr>
</tbody>
</table>

Conceptual Objectives: The student successfully completing this course the student shall understand and be able to

- Explain and differentiate the various software development management models.
- Articulate the factors that contribute to software productivity.
- Be able to perform project management tasks such as project estimation, developing a project plan.
- Be able to explain the role of risk and discuss strategies for risk identification, assessment and mitigation/
- Explain techniques for continuous process improvement.
- Discuss the special issues that arise when managing software development projects that involve outsourcing, geo-distributed software development teams, and open-source software.
- Identify the subject areas in the body of knowledge for the Project Management Professional certification.

Performance Objectives: After successful completion of this course, the student shall be able to do the following.

- Given a broad statement of work, estimate the software size and project staffing and duration.
- Develop a project schedule based on tasks and staffing profiles.
- Evaluate a project proposal with respect to technical completeness (given a statement of need).
- Perform a systematic evaluation of a legacy system to identify basic requirements.
- Select the most appropriate management model for a given development project.
- Perform a project audit to determine the “health” of the project and to identify evidence of uncontrolled risk.
- Perform project management tasks using automated project management tools (like COCOMO II, MS Project, etc.).
- Use popular project management software such as MS-Project.