برنامه پیشنهادی برای کارشناسی مهندسی نرمافزار

همانطور که مطلع هستید، برنامه جامعی را برای رشته مهندسی نرمافزار در مقطع کارشناسی پیشنهاد کرده‌اند که مبنای تعیین این رشته در دانشگاه‌های متعدد بوده است (این برنامه در این آدرس در دسترس است: (http://sites.computer.org/ccse/SE2004Volume.pdf).

دروس اصلی مهندسی نرمافزار

طبق پیشنهاد ACM و IEEE، برنامه درسی رشته‌های مهندسی نرمافزار در مقطع کارشناسی بايد حداکثر شامل ۲۱ واحد تربیتی-Sh. برنامه پیشنهادی ACM و IEEE شامل دو مجموعه شش تایی از دروس اصلی مهندسی نرمافزار است که دانشگاه محل ارائه می‌تواند از بین این دو مجموعه پیشنهادی یکی را انتخاب و بیان‌سازی نماید. این دو مجموعه شامل: نرمافزار، سیستم‌های آزاد. که عیناً از برنامه پیشنهادی ACM و IEEE بفرستد. شداند. در صفحات بعد آورده شده‌دند (معمولاً های فارسی تا حذی مورد مناسب‌سازی گرفته‌اند). اگر قری به اتفاق دروس مجموعه دوم هم اکنون در دانشگاه‌های ما ارائه می‌شوند (در قالب دروس کارشناسی یا ارزش); به همین دلیل به نظر می‌رسد که مجموعه دوم برای ایران مناسب‌تر باشد.

<table>
<thead>
<tr>
<th>مجموعه دوم (برگرفته از برنامه ACM/IEEE، صفحه ۵۵)</th>
<th>مجموعه اول (برگرفته از برنامه ACM/IEEE، صفحه ۵۵)</th>
</tr>
</thead>
<tbody>
<tr>
<td>۱ - مقدمه‌ای بر مهندسی نرم‌افزار ۲-۱۱۱</td>
<td>۱ - مقدمه‌ای بر مهندسی نرم‌افزار ۲-۱۱۱</td>
</tr>
<tr>
<td>۲-تولید کامپیوتر در مهندسی نرم‌افزار (۲-۱۱۱(B))</td>
<td>۲ - ساخت نرم‌افزار (۲-۱۱۱(A))</td>
</tr>
<tr>
<td>۳-جایگاه و مجازی سامانه‌های نرم‌افزار برگز (۲-۱۱۱(A))</td>
<td>۲-تولید سخت‌افزار (۲-۱۱۱(B))</td>
</tr>
<tr>
<td>۴-آزمون نرم‌افزار (۲-۱۱۱(C))</td>
<td>۳-در حال و مجازی نرم‌افزار (۲-۱۱۱(D))</td>
</tr>
<tr>
<td>۵-طراحی تفصیلی نرم‌افزار (۲-۱۱۱(D))</td>
<td>۴-در حال و مجازی نرم‌افزار (۲-۱۱۱(E))</td>
</tr>
<tr>
<td>۶-روش‌های عرضه در مهندسی نرم‌افزار (۲-۱۱۱(F))</td>
<td>۵-تجزیه و تحلیل خواص‌های نرم‌افزار (۲-۱۱۱(E))</td>
</tr>
<tr>
<td>۷-فرآیند و مدیریت انجام نرم‌افزار (۲-۱۱۱(G))</td>
<td>۶-آزمون نرم‌افزار (۲-۱۱۱(F))</td>
</tr>
<tr>
<td>۸-پروژه نهایی مهندسی نرم‌افزار: این پروژه، دو مرحله‌ای است و در قالب دو درن سه واحدی در سال آخر تحلیل</td>
<td>۷-پروژه نهایی مهندسی نرم‌افزار: این پروژه، دو مرحله‌ای است و در قالب دو درن سه واحدی در سال آخر تحلیل</td>
</tr>
<tr>
<td>در دو نیمسال متوالی - ارتقاء می‌شود ۴۰۰</td>
<td>در دو نیمسال متوالی - ارتقاء می‌شود ۴۰۰</td>
</tr>
</tbody>
</table>
چاپگاه دروس اصلی مهندسی نرم‌افزار در یک برنامه نوعی

نمونه‌های متعددی از برنامه‌های کارشناسی مهندسی نرم‌افزار در برنامه پیشنهادی مشخص شده‌اند: IEEE و ACM نسخه‌ای از صفحه ۴ برنامه پیشنهادی (ACM/IEEE) شناختمی‌که نوعاً چگونه می‌توان یک رشته کارشناسی مهندسی نرم‌افزار را در یک دانشکده مهندسی ارائه نمود.

جدول ۱ - نمونه دوره کارشناسی مهندسی نرم‌افزار در یک دانشکده مهندسی

<table>
<thead>
<tr>
<th>Year 1</th>
<th>Year 2</th>
<th>Year 3</th>
<th>Year 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sem 1A</td>
<td>Sem 1B</td>
<td>Sem 2A</td>
<td>Sem 2B</td>
</tr>
<tr>
<td>CS101</td>
<td>CS102</td>
<td>CS103</td>
<td>CS220</td>
</tr>
<tr>
<td>Calc 1</td>
<td>Calc 2</td>
<td>Calc 106</td>
<td>SE A</td>
</tr>
<tr>
<td>MA271</td>
<td>MA271</td>
<td>SE D</td>
<td>SE F</td>
</tr>
<tr>
<td>NT181</td>
<td>NT181</td>
<td>SE221</td>
<td>Tech elect</td>
</tr>
<tr>
<td>Physics 1</td>
<td>Physics 2</td>
<td>NT272</td>
<td>Lin Alg</td>
</tr>
<tr>
<td>Tech elect</td>
<td>Tech elect</td>
<td>Tech elect</td>
<td>Tech elect</td>
</tr>
<tr>
<td>Chemistry</td>
<td>Engineering</td>
<td>Calc 3</td>
<td>Gen Alg</td>
</tr>
<tr>
<td>Gen ed</td>
<td>Gen ed</td>
<td>Gen ed</td>
<td>Gen ed</td>
</tr>
<tr>
<td>Gen ed</td>
<td>Gen ed</td>
<td>Gen ed</td>
<td>Gen ed</td>
</tr>
<tr>
<td>Gen ed</td>
<td>Gen ed</td>
<td>Gen ed</td>
<td>Gen ed</td>
</tr>
<tr>
<td>Gen ed</td>
<td>Gen ed</td>
<td>Gen ed</td>
<td>Gen ed</td>
</tr>
</tbody>
</table>

برنامه نشان داده شده در جدول ۱ غیر از دروس مهندسی نرم‌افزار (با پیشوند SE)، که در صفحه قبل فهرست شده‌اند، شامل دروس علمی کامپیوتر زیر هست (با پیشوند CS):

- CS101 Programming Fundamentals
- CS102 Object-Oriented Paradigm (OO Programming)
- CS103 Data Structures and Algorithms
- CS105 Discrete Structures I
- CS106 Discrete Structures II
- CS220 Computer Architecture
- CS226 Operating Systems and Networking
- CS270T Databases

سایر دروس این جدول، از انواع دروس عمومی، پایه‌ای یا اختیاری هستند که از توضیح آنها صرف نظر کرده‌ایم.

پیشنهادی برای ارائه این دوره (به صورت رشته مستقل با گرایش) در دانشکده خودمان به نظر بنده، به درس مقدماتی بر مهندسی نرم‌افزار ۲۰۱۹ طراحی و معماری سامانه‌های نرم‌افزاری بزرگ - SE-213(A) و طراحی نرم‌افزار (D) SE-312(B) برای می‌توان با تغییراتی در قالب سه درس موجود (تحلیل و طراحی سیستم‌ها، مهندسی نرم‌افزار، طراحی شبکه) ارائه کرد. از طرف دیگر، پروژه دوم و رحلتهای مهندسی نرم‌افزار ۴۰۰SE-313(C) با دو درس موجود آن مهندسی نرم‌افزار و پروژه کارشناسی قابل تطیف است.

چهار درس باقی‌مانده عبارتند از: تعامل انسان و کامپیوتر در مهندسی نرم‌افزار (SE-212(B) فرانریز و مدیریت ایجاد نرم‌افزار (SE-313(F) و روش‌های رسمی در مهندسی نرم‌افزار (SE-211(E) آزمون نرم‌افزار (C) SE-324 ضرورت در مهندسی نرم‌افزار در حال حاضر در نقطه کارشناسی در دانشکده ارائه می‌شود: دو درس اول در مقطع کارشناسی در دانشکده ارائه می‌شود، اما هر دو درس دیگر در مقطع ارشد ارائه می‌شود. اما به توجه می‌توان نسبت‌های مناسب دوره کارشناسی برای آن‌ها طراحی نمود (نمونه‌هایی از این دروس در دانشگاه‌های کانادا.}

صفحه ۲۲ از
وجود دارند). برای پیشنهاد چارت جدید، این چهار درس باید جایگزین چهار درس موجود در برنامه فعلی کارشناسی نرمافزار شوند؛ مقایسه با برنامه کارشناسی مهندسی نرمافزار در دانشگاه واترلو نشان می‌دهد که دروس زیر کاندیداهای مناسبی برای اختیاری شدن هستند. تا بدين ترتيب جا برای ارائه چهار درس جديد باژ شود:

1- بازيابي پيشرفته اطلاعات
2- هوش مصنوعي
3- محاسبات عددی
4- طراحی سيستم های ديجيتال + آز

با عرض احترام

رامان رامسين
SE-201 Introduction to Software Engineering

Course description
Main Topics:

1. Principles of software engineering: Requirements, design and testing.
2. Review of principles of object orientation.
3. Object oriented analysis using UML.
4. Frameworks and APIs.
5. Introduction to the client-server architecture.
6. Analysis, design and programming of simple servers and clients.
7. Introduction to user interface technology.

Learning objectives
Upon completion of this course, students will have the ability to:

- Develop clear, concise, and sufficiently formal requirements for extensions to an existing system, based on the true needs of users and other stakeholders
- Apply design principles and patterns while designing and implementing simple distributed systems-based on reusable technology
- Create UML class diagrams which model aspects of the domain and the software architecture
- Create UML sequence diagrams and state machines that correctly model system behavior
- Implement a simple graphical user interfaces for a system
- Apply simple measurement techniques to software
- Demonstrate an appreciation for the breadth of software engineering

Suggested sequence of teaching modules

1. Software engineering and its place as an engineering discipline
2. Review of the principles of object orientation
3. Reusable technologies as a basis for software engineering: Frameworks and APIs.
4. Introduction to client-server computing
5. Requirements analysis
6. UML class diagrams and object-oriented analysis; introduction to formal modeling using OCL. Examples of building class diagrams to model various domains
7. Design patterns (abstraction-occurrence, composite, player-role, singleton, observer, etc.)
8. Use cases and user-centered design
9. Representing software behavior: Sequence diagrams, state machines, activity diagrams
10. General software design principles: Decomposition, decoupling, cohesion, reuse, reusability, portability, testability, flexibility, etc.
11. Software architecture: Distributed architectures, pipe-and-filter, model-view-controller, etc.
12. Introduction to testing and project management
Sample labs and assignments

- Evaluating the performance of various simple software designs
- Adding features to an existing system
- Testing a system to verify conformance to test cases
- Building a GUI for an application
- Numerous exercises building models in UML, particularly class diagrams and state machines
- Developing a simple set of requirements (to be done as a team) for some innovative client-server application of very small size
- Implementing the above, using reusable technology to the greatest extent possible
SE-211 Software Construction

Course Description
Main Topics:

1- General principles and techniques for disciplined low-level software design.
2- BNF and basic theory of grammars and parsing.
3- Use of parser generators.
4- Basics of language and protocol design.
5- Formal languages.
6- State-transition and table-based software design.
7- Formal methods for software construction.
8- Techniques for handling concurrency and inter-process communication.
9- Techniques for designing numerical software.
10- Tools for model-driven construction.
11- Introduction to Middleware.
12- Hot-spot analysis and performance tuning.

Prerequisite: (SE201 or SE200), CS103 and CS105.

Learning objectives
Upon completion of this course, students will have the ability to:

- Apply a wide variety of software construction techniques and tools, including state-based and table-driven approaches to low-level design of software
- Design simple languages and protocols suitable for a variety of applications
- Generate code for simple languages and protocols using suitable tools
- Create simple formal specifications of low-level software modules, check the validity of these specifications, and generate code from the specifications using appropriate tools
- Design simple concurrent software
- Analyze software to improve its efficiency, reliability, and maintainability

Suggested sequence of teaching modules

1- Basics of formal languages; syntax and semantics; grammars; Backus Naur Form. Parsing; regular expressions and their relationship to state diagrams
2- Lexical Analysis; tokens; more regular expressions and transition networks; principles of scanners
3- Using tools to generate scanners; applications of scanners. Relation of scanners and compilers
4- Parsing concepts; parse trees; context free grammars, LL Parsing
5- Overview of principles of programming languages. Criteria for selecting programming languages and platforms
6- Tools for automating software design and construction. Modeling system behavior with extended finite state machines
7- SDL
8- Representing concurrency, and analyzing concurrent designs

Sample labs and assignments

- Use of software engineering tools to create designs
- Use of parser generators to generate languages

تا جایی که من اطلاع دارم، این درس معادل تزدلیکی در بین دروس فضای مهندسی نرمافزار دانشگاه‌های ایران ندارد.
SE-212  Software Engineering Approach to HCI

Course Description
Main Topics:

1. Psychological principles of human-computer interaction.
2. Evaluation of user interfaces.
3. Usability engineering.
4. Task analysis, user-centered design, and prototyping.
5. Conceptual models and metaphors.
6. Software design rationale.
7. Design of windows, menus, and commands.
8. Voice and natural language I/O.
9. Response time and feedback.
10. Color, icons, and sound.
11. Internationalization and localization.
12. User interface architectures and APIs.
13. Case studies and project.

Prerequisite: SE201 or SE200

Learning objectives
Upon completion of this course, students will have the ability to:

- Evaluate software user interfaces using heuristic evaluation and user observation techniques
- Conduct simple formal experiments to evaluate usability hypotheses.
- Apply user centered design and usability engineering principles as they design a wide variety of software user interfaces

Suggested sequence of teaching modules

1. Background to human-computer interaction. Underpinnings from psychology and cognitive science
3. More evaluation techniques: Videotaped user testing; cognitive walkthroughs
4. Task analysis. User-centered design
5. Usability engineering processes; conducting experiments
6. Conceptual models and metaphors
7. Designing interfaces: Coding techniques using color, fonts, sound, animation, etc.
8. Designing interfaces: Screen layout, response time, feedback, error messages, etc.
9. Designing interfaces for special devices. Use of voice I/O
10. Designing interfaces: Internationalization, help systems, etc. User interface software architectures
11. Expressing design rationale for user interface design

Sample labs and assignments

- Evaluation of user interfaces using heuristic evaluation
- Evaluation of user interfaces using videotaped observation of users
- Paper prototyping of user interfaces, then discussing design options in order to arrive at a consensus design
- Writers-workshop for style critiquing of prototypes presented by others
- Implementation of a system with a significant user interface component using a rapid prototyping environment
SE-213  Design and Architecture of Large Software Systems

Course Description
Main Topics:

1. Modeling and design of flexible software at the architectural level.
2. Basics of model-driven architecture.
3. Architectural styles and patterns.
4. Middleware and application frameworks.
5. Configurations and configuration management.
6. Product lines.
7. Design using Commercial Off-The-Shelf (COTS) software.

Prerequisites: SE201 or SE200, and CS103

Learning objectives
Upon completion of this course, students will have the ability to:

- Take requirements for simple systems and develop software architectures and high-level designs
- Use configuration management tools effectively, and apply change management processes properly
- Design simple distributed software
- Design software using COTS components
- Apply a wide variety of frameworks and architectures in designing a wide variety of software
- Design and implement software using several different middleware technologies
SE-221  Software Testing

Course Description
Main Topics:

1. Testing techniques and principles: Defects vs. failures, equivalence classes, boundary testing.
2. Types of defects.
3. Black-box vs. Structural testing.
4. Testing strategies: Unit testing, integration testing, profiling, test driven development.
5. State based testing; configuration testing; compatibility testing; web site testing.
6. Alpha, beta, and acceptance testing.
7. Coverage criteria.
8. Test instrumentation and tools.
9. Developing test plans.
10. Managing the testing process.
11. Problem reporting, tracking, and analysis.

Prerequisites: SE201 or SE200

Learning objectives
Upon completion of this course, students will have the ability to:

- Analyze requirements to determine appropriate testing strategies.
- Design and implement comprehensive test plans
- Apply a wide variety of testing techniques in an effective and efficient manner
- Compute test coverage and yield according to a variety of criteria
- Use statistical techniques to evaluate the defect density and the likelihood of faults.
- Conduct reviews and inspections.

این درس تا حد زیادی با درس کارشناسی ارشد "آزمون نرمافزار" قابل تطیب است.
Course Description

Main Topics:

1. An in-depth look at software design.
2. Continuation of the study of design patterns, frameworks, and architectures.
3. Survey of current middleware architectures.
4. Design of distributed systems using middleware.
5. Component-based design.
6. Measurement theory and appropriate use of metrics in design.
7. Designing for qualities such as performance, safety, security, reusability, reliability, etc.
8. Measuring internal qualities and complexity of software.

Prerequisites: SE211

Learning objectives

Upon completion of this course, students will have the ability to:

- Apply a wide variety of design patterns, frameworks, and architectures in designing a wide variety of software
- Design and implement software using several different middleware technologies
- Use sound quality metrics as objectives for designs, and then measure and assess designs to ensure the objectives have been met
- Modify designs using sound change control approaches
- Use reverse engineering techniques to recapture the design of software
SE-312  Low-Level Design of Software

Course Description
Main Topics:

1. Detailed software design and construction in depth.
2. In-depth coverage of design patterns and refactoring.
3. Introduction to formal approaches to design.
4. Analysis of designs based on internal quality criteria.
5. Performance and maintainability improvement.
6. Reverse engineering.
7. Disciplined approaches to design change.

Prerequisite: SE213

Learning objectives
Upon completion of this course, students will have the ability to:

- Apply a wide variety of software construction techniques and tools, including state-based and table-driven approaches to low-level design of software
- Use a wide variety of design patterns in the design of software
- Perform object-oriented design and programming with a high level of proficiency
- Analyze software in order to improve its efficiency, reliability, and maintainability.
- Modify designs using sound change control approaches
- Use reverse engineering techniques to recapture the design of software
SE-313  Formal Methods in Software Engineering

Course Description
Main Topics:

1. Review of mathematical foundations for formal methods.
2. Formal languages and techniques for specification and design, including specifying syntax using grammars and finite state machines.
3. Analysis and verification of specifications and designs.
4. Use of assertions and proofs.
5. Automated program and design transformation.

Prerequisite: SE312.

Learning objectives
Upon completion of this course, students will have the ability to:

- Create mathematically precise specifications and designs using languages such as OCL, Z, etc.
- Analyze the properties of formal specifications and designs
- Use tools to transform specifications and designs

این درس تا حد زیادی با درس کارشناسی ارشد "توصیف و وارسی برنامه‌ها" قابل تطبیق است.
SE-321 Software Quality Assurance and Testing

Course Description
Main Topics:

1- Quality: how to assure it and verify it, and the need for a culture of quality.
2- Avoidance of errors and other quality problems.
3- Inspections and reviews.
4- Testing, verification and validation techniques.
5- Process assurance vs. Product assurance.
6- Quality process standards.
7- Problem analysis and reporting.
8- Statistical approaches to quality control.

Prerequisite: SE201 or SE200

Learning objectives
Upon completion of this course, students will have the ability to:

- Conduct effective and efficient inspections
- Design and implement comprehensive test plans
- Apply a wide variety of testing techniques in an effective and efficient manner
- Compute test coverage and yield, according to a variety of criteria
- Use statistical techniques to evaluate the defect density and the likelihood of faults
- Assess a software process to evaluate how effective it is at promoting quality

Suggested sequence of teaching modules

1- Introduction to software quality assurance
2- Inspections and reviews
3- Principles of software validation
4- Software verification
5- Software testing
6- Specification based test construction techniques
7- White-box and grey-box testing
8- Control flow oriented test construction techniques
9- Data flow oriented test construction techniques
10- Cleanroom approach to quality assurance
11- Software process certification
Sample labs and assignments

- Use of automated testing tools
- Testing of a wide variety of software
- Application of a wide variety of testing techniques
- Inspecting of software in teams; comparison and analysis of results
SE322  Software Requirements Analysis

Course Description
Main Topics:

1- Domain engineering.
2- Techniques for discovering and eliciting requirements.
3- Languages and models for representing requirements.
4- Analysis and validation techniques, including need, goal, and use case analysis.
5- Requirements in the context of system engineering.
6- Specifying and measuring external qualities: performance, reliability, availability, safety, security, etc.
7- Specifying and analyzing requirements for various types of systems: embedded systems, consumer systems, web-based systems, business systems, systems for scientists and other engineers.
8- Resolving feature interactions.
9- Requirements documentation standards.
10- Traceability.
11- Human factors.
12- Requirements in the context of agile processes.
13- Requirements management: Handling requirements changes.

Prerequisites: SE201 or SE200.

Learning objectives
Upon completion of this course, students will have the ability to:

- Discover or elicit requirements using a variety of techniques
- Organize and prioritize requirements
- Apply analysis techniques such as needs analysis, goal analysis, and use case analysis
- Validate requirements according to criteria such as feasibility, clarity, freedom from ambiguity, etc.
- Represent functional and non-functional requirements for different types of systems using formal and informal techniques
- Specify and measure quality attributes
- Negotiate among different stakeholders in order to agree on a set of requirements
- Detect and resolve feature interactions
Suggested sequence of teaching modules

1- Basics of software requirements engineering
2- Requirements engineering process: requirements elicitation, specification, analysis, and management
3- Types of requirements: functional, non-functional, quality attributes
4- Requirements elicitation: identifying needs, goals, and requirements. Customers and other stakeholders. Interviews and observations
5- Requirements specification: textual and graphical notations and languages (UML, User Requirements notation). Techniques to write high-quality requirements. Documentation standards
6- Requirements analysis: inspection, validation, completeness, detection of conflicts and inconsistencies. Feature interaction analysis and resolution
7- Goal- and use-case-oriented modeling, prototyping, and analysis techniques
8- Requirements for typical systems: embedded systems, consumer systems, web-based systems, business systems, systems for scientists and other engineers
9- Requirements management: traceability, priorities, changes, baselines, and tool support
10- Requirements negotiation and risk management
11- Integrating requirements analysis and software processes (including agile ones)

Sample labs and assignments

- Writing good requirements.
- Analysis of a wide variety of existing software systems: Measuring qualities, and reverse engineering requirements.
- Interviewing users, and translating the results into prototypes iteratively
- Use of tools for managing requirements.
- Modeling, prototyping, and analyzing requirements with UML/URN tools
- Resolving feature interactions

تا جایی که من اطلاع دارم، معادل این درس تنها در دانشگاه امیرکبیر ارائه می‌شود.
SE323  Software Project Management

Course Description
Main Topics:

1- Project planning, cost estimation, and scheduling.
2- Project management tools.
3- Factors influencing productivity and success.
4- Productivity metrics.
5- Analysis of options and risks.
6- Planning for change.
7- Management of expectations.
8- Release and configuration management.
9- Software process standards and process implementation.
10- Software contracts and intellectual property.
11- Approaches to maintenance and long-term software development.
12- Case studies of real industrial projects.

Prerequisites: SE321 and SE322

Learning objectives
Upon completion of this course, students will have the ability to:

- Develop a comprehensive project plan for a significant development effort
- Apply management techniques to projects that follow agile methodologies, as well as methodologies involve larger-scale iterations or releases
- Effectively estimate costs for a project using several different techniques.
- Apply function point measurement techniques
- Measure project progress, productivity and other aspects of the software process
- Apply earned-value analysis techniques
- Perform risk management, dynamically adjusting project plans
- Use configuration management tools effectively, and apply change management processes properly
- Draft and evaluate basic software licenses, contracts, and intellectual property agreements, while recognizing the necessity of involving legal expertise
- Use standards in project management, including ISO 10006 (project management quality) and ISO 12207 (software development process) along with the SEI’s CMM model
Suggested sequence of teaching modules

1- Basic concepts of project management
2- Managing requirements
3- Software lifecycles
4- Software estimation
5- The project plan
6- Monitoring the project
7- Risk analysis
8- Managing quality
9- People problems

Sample labs and assignments

- Use a commercial project management tool to assist with all aspects of software project management. This includes creating Gantt, PERT, and Earned Value charts
- Make cost estimates for a small system using a variety of techniques
- Developing a project plan for a significant system
- Writing a configuration management plan
- Using change control and configuration management tools
- Evaluating a software contract or license

این درس تا حد زیادی با درس کارشناسی "مدیریت پروژه" هموارهی دارد.
SE-324 Software Process and Management

Course Description
Main Topics:

2. Project management with a focus on requirements management and long-term evolution: Eliciting and prioritizing requirements, cost estimation, planning and tracking projects, risk analysis, project control, change management.

Prerequisites: SE201 or SE200, plus at least two additional software engineering courses at the 2 level or higher.

Learning objectives
Upon completion of this course, students will have the ability to:

- Elicit requirements using a variety of techniques
- Organize and prioritize requirements
- Design processes suitable for different types of project
- Assess a software process, to evaluate how effective it is at promoting quality
- Develop a comprehensive project plan for a significant development effort
- Measure project progress, productivity and other aspects of the software process
- Effectively estimate costs for development and evolution of a system using several different techniques
- Perform risk management, dynamically adjusting project plans
- Use standards for quality, process and project management
- Perform root cause analysis, and work towards continual improvement of process
SE-400  Software Engineering Capstone Project

Course Description
This is a two-semester senior course focusing on development of a significant software system, employing knowledge gained from courses throughout the program. Includes development of requirements, design, implementation, and quality assurance. Students may follow any suitable process model, must pay attention to quality issues, and must manage the project themselves, following all appropriate project management techniques. Success of the project is determined in large part by whether students have adequately solved their customer’s problem.

Sample deliverables
Students should be expected to deliver one or several iterations of a software system, along with all artifacts appropriate to the process model they are using. These would likely include a project plan (perhaps updated regularly, and containing cost estimations, risk analysis, division of the work into tasks, etc.), requirements (including use cases), architectural and design documents, test plans, source code, and installable system.