Description of Courses

SEP503 Computer Science for Software Engineers
This course offers various background topics needed for software engineers to start a graduate level study of software engineering. The main topics of the course are 'Advanced Discrete Mathematics, UML 2.0 and Design Tool, Advanced Data Structure and Algorithms, Program Execution, Object-Oriented Method.' In addition, time permitting, the following topics that draw much attention today in software engineering field may also be covered: Middleware, Design Patterns, Component Technology.

SEP537 Models of Software Systems
Computer Scientists have long investigated the problem of how to automate software development from its specification to its program. So far the efforts were not fully successful but much of the results can be fruitfully applied to development of small programs and critical small portions of large programs. In this course, we learn how to formally write requirements, how to formally model specifications and how to rigorously verify that the models have the required properties.

SEP539 Methods: Deciding What to Design
Practical development of software requires an understanding of successful methods for bridging the gap between a problem to be solved and a working software system. In this course you will study a variety of ways to understand the problem you're solving, the various factors that constrain the possible solutions, and approaches to deciding among alternatives.

SEP591 Managing Software Development
Large scale software development requires the ability to manage resources - both human and computational - through control of the development process. This course is a breadth oriented course, designed to help technically-trained software engineers to acquire the knowledge and skills necessary to lead a project team, understand the relationship of software development to overall project engineering, estimate time and costs, and understand the software process. The nature of software development is sufficiently unique to require specialized management techniques, especially in the areas of the estimating and scheduling.

SEP601, 602, 603 Software Development Studio I, II, III
Software Development Studio provides students with a laboratory for direct application of concepts learned in core courses. Teamwork, organization, and the use of disciplined software processes are stressed in studio project work. Software Development Studio spans the entire MSE program over an academic year, consisting of Software Development Studio I, II, and III for the Fall, Spring, and Summer semesters respectively. Software Development Studio acts a laboratory where students apply knowledge gained from core and elective courses in realistic, yet monitored, environments. Former and or practicing professionals are selected to mentor each Studio project, providing team as well ad individual guidance. With their significant industrial experience, mentors bring their experience to bear in guiding students in their application of methods, techniques, and technologies learned in the classroom to real-world problems encountered in Studio.

SEP604 CMU Software Studio
This course is required to be registered with SEP603 Software Development Studio III. In this course, CMU mentors give guidance to the studio team members by having regular meetings either via online or by visiting the students.

Analysis of Software Artifact
Analysis is the systematic examination of an artifact to determine its properties. This course will focus on analysis of software artifacts--primarily code, but also including analysis of designs, architectures, and test suites. We will focus on functional properties, but also cover non-functional properties like performance and security. In order to illustrate core analysis concepts in some depth, the course will center on static program analysis; however, the course will also include a breadth of techniques such as testing, model checking, theorem proving, dynamic analysis, and type systems. The course will balance theoretical discussions with lab exercises in which students will apply the ideas they are learning to real artifacts.

Architecture for Software System

Successful design of complex software systems requires the ability to describe, evaluate, and create systems at an architectural level of abstraction. This course introduces architectural design of complex software systems. The course considers commonly-used software system structures, techniques for designing and implementing these structures, models and formal notations for characterizing and reasoning about architectures, tools for generating specific instances of an architecture, and case studies of actual system architectures. It teaches the skills and background students need to evaluate the architectures of existing systems and to design new systems in principled ways using well-founded architectural paradigms.

Practicum Proposal

In this course, students propose a project to work on during their practicum courses. This will take the form of a formal document and will be considered the first deliverable of the practicum. The proposal must include an executive summary, a definition of the work to be completed, including deliverables, timelines, reviews, and final report.

Practicum I, II

Practicum is the capstone demonstration by the student of their abilities as a software engineer. The purpose of the practicum is for the student to demonstrate command of the material learned in the core and electives courses. Students will do so by solving a substantial practical problem in a realistic setting with a focus on understanding major aspects of the software development life cycle in detail. Practicum is intended for individual students or small teams of three or four students.

* The descriptions of elective courses, mandatory general courses and research courses is the same as them of the Software Graduate Program's courses.