

CERTIFICATE IN SOFTWARE ENGINEERING

Academic Year

2014-2015

Department

Software Engineering

Program Director

Djaffer Ibaroudene, Ph.D. dibaroudene@stmarytx.edu

About the Certificate

The certificate program in Software Engineering provides students with the knowledge and the skills necessary to apply sound engineering theory, principles, tools and processes, as well as principles of computer science and mathematics, to the evaluation, design, development, testing and maintenance of complex software systems. This program is designed to serve the needs of individuals who would like to increase their knowledge and skill levels in the field of software engineering in a short time frame.

It is important to note that this certificate program does not focus on providing students with a large repertoire of programming skills that emphasize particular language syntax. Coding is only one phase of the software life cycle. Instead, the program gives students the knowledge and skills necessary to develop and manage large-scale software systems with a primary focus on quality, reliability, and maintainability. It emphasizes the organizational structures, development processes, analysis and modeling techniques, design methodologies, documentation standards, and quality controls. The student will learn all aspects of software development; namely, software project management, system specifications, requirement analysis, design techniques, documentation, coding, testing, maintenance, quality assurance, and configuration management.

Students who successfully complete the certificate program are highly encouraged to continue and earn their Master of Science degree in Software Engineering by taking six additional courses and complete a capstone project or five additional courses and a thesis.

Admission Requirements

Applicants must have a Bachelor of Science degree in Software Engineering, Computer Science, Computer Engineering or a closely related discipline.

Applicants whose Bachelor of Science degree is not in Software Engineering, Computer Science or Computer Engineering are required to demonstrate proficiency in the following subjects or take the corresponding prerequisite courses at St. Mary's University or other institutions:

Prerequisite (one of the following languages)	Corresponding Course at St. Mary's
C/C++ Programming	EG1302 – C Programming for Engineers
Java	EG3392 – Java and Applications
Other programming languages may be acceptable	CS1310 – Programming 1
Data Structure and Algorithms	E2342 – Data Structures and Algorithms Or CS1311 – Programming 11 and CS2315 – Algorithms

Certificate Requirements

The certificate program in Software Engineering is a four-course (12 hour) program. Students will choose four courses from this list of three-hour courses:

CERTIFICATE IN SOFTWARE ENGINEERING

- EG6306 Software Project Planning and Management
- EG7304 Requirement Engineering
- EG7312 Software Design and Architecture
- EG6334 Software Quality Assurance
- EG7308 Software Validation and Verification
- EG7310 Software Maintenance Evolution and Reengineering

Department Courses and Descriptions

EG 6306 **Software Project Planning and Management** (3)

Planning and control of software project. Cost factors and cost estimation. Project scheduling, staffing, setting milestones. Role of project manager and organization of project team. Project management tools. Factors influencing productivity and success. Productivity metrics. Software project economics. Metrics for software quality, schedule, budget, and progress. Analysis of options and risks. Planning for change. Management of expectations. Release and configuration management. Software process standards and process implementation. Software contracts and intellectual property. Approaches to maintenance and long-term software development. Case studies of real industrial projects. CASE tools for project planning, cost estimation, and project management

EG 6328 **Software Engineering** (3)

This course surveys the entire software engineering field. It presents the management and technical aspects of the software development process. Software architectures, paradigms, and life-cycles are briefly discussed and compared. It covers topics in software management, problem specification and analysis, system design techniques, documentation, system testing and performance evaluation, and system maintenance. The technical aspects include software requirement analysis, design methodologies, system implementation, and testing techniques. Software verification and validation, quality assurance, and configuration management are also introduced.

EG 6334 **Software Quality Assurance** (3)

Quality: How to assure it and verify it? Avoidance of errors and other quality problems. Inspections and formal technical reviews. Testing, verification, and validation techniques. Process assurance versus product assurance. Quality work product attributes. Software quality measurements and metrics. Quality process standards and formal approaches to SQA. Product and process assurance. Problem analysis and reporting. Statistical approach to quality control. Software configuration management: baselines, version control, change control, configuration audits, and SCM standards. CASE tools for SQA.

EG 7155, 7255, 7355 **Internship** 1, 2, 3 semester hour(s)

An experimental approach to advanced engineering topics through work in a company or organization. Industry/University cooperation is required. Topics vary depending on the needs of the sponsoring company or organization and the academic needs of the student. Students may start an internship projects anytime after enrolling in any Engineering program. A minimum of three credit hours is required. Credit hours may be completed in increments of 1, 2, or 3 credit hours per semester. Prerequisite: Consent of the Graduate Program Director.

EG 7304 **Requirement Engineering** (3)

Domain engineering. Techniques for discovering and eliciting requirements. Languages and models for representing requirements. Analysis and validation techniques, including need, goal, and use case analysis. Requirements in the context of system engineering. Specifying and measuring external qualities: performance, reliability, availability, safety, security, etc. Specifying and analyzing requirements for various types of systems: embedded systems, consumer systems, web-based systems, business systems, systems for scientists and other engineers. Resolving feature interactions. Requirements documentation standards. Requirement traceability. Human factors. Requirements in the context of agile processes. Requirements management: Handling requirements changes. CASE tools for requirement engineering.

EG 7305 **Object-Oriented Analysis and Design Methodologies** (3)

Review of object oriented concepts: objects, classes, instances, inheritance, and entity relationship diagrams. Object-oriented analysis methodologies and their role in the software development process. Object-oriented modeling and prototyping using

CERTIFICATE IN SOFTWARE ENGINEERING

UML. Software reuse. Design patterns, frameworks, architectures. Component design. Measures of design attributes. Component and system interface design.

EG 7308 **Software Verification and Validation** (3)

Testing techniques and principles: defects versus failures, equivalence classes, boundary testing. Types of defects. Black-box versus structural testing. Testing categories: Unit testing, integration testing, profiling, test driven development. State-based testing, configuration testing, compatibility testing. Website testing. Alpha, beta, and acceptance testing. Coverage criteria. Test instrumentation and tools. Developing a test plan. Managing the test process. Problem reporting, tracking, and analysis. Testing metrics. Software safety. Debugging and fault isolation techniques. Defect analysis.

EG 7309 **Formal Methods in Software Engineering** (3)

Review of mathematical foundation for formal methods. Formal languages and techniques for specification and design, including specifying syntax using grammars and finite state machines. Analysis and verification of specification and designs. Use of assertion and proofs. Automated program and design transformation.

EG 7310 **Software Maintenance, Evolution, and Reengineering** (3)

Introduction to software maintenance, defect management, corrective, adaptive and perfective maintenance. Evolution of legacy software systems. Program comprehension techniques, reverse engineering, restructuring, refactoring of software systems. Software re-engineering, data reverse engineering. Software reuse. Impact analysis, regression testing.

EG 7311 **User Interface Design** (3)

Psychological principles of human-computer interaction. Evaluation of user interfaces. Usability engineering. Task analysis, user-centered design, and prototyping. Conceptual models and metaphors. Software design rationale. Design of windows, menus, and commands. Voice and natural language I/O. Response time and feedback. Color, icons, and sound. Internationalization and localization. User interface architectures and APIs. Case studies and project.

EG 7312 **Software Design and Architecture** (3)

Modeling and design of flexible software at the architectural level. Basics of model-driven architecture. Architectural styles and patterns. Middleware and application frameworks. An in-depth look at software design. Survey of current middleware architectures. Design of distributed systems using middleware. Component based design. Measurement theory and appropriate use of metrics in design. Designing for qualities such as performance, safety, security, reusability, reliability, etc. Measuring internal qualities and complexity of software. Evaluation and evolution of designs.

EG 7313 **Web Engineering** (3)

Concepts, principles, techniques, and methods of Web engineering. Topics include requirement engineering for Web applications, modeling Web applications, Web application architectures, Web application design, technologies for Web applications, testing Web applications, operation and maintenance of Web applications, web project management, web application development process, usability of Web applications, performance of Web applications, and security of Web applications. Quality characteristic and attributes for websites.

EG 7314 **Software Security** (3)

Theory and practice of software security. Identification of potential threats and vulnerabilities early in the design cycle. Methodologies and tools for identifying and eliminating security vulnerabilities. Techniques to prove the absence of vulnerabilities and ways to avoid security holes in new software. Essential guidelines for building secure software: how to design software with security in mind from the ground up and to integrate analysis and risk management throughout the software life cycle.

EG 7314 **Software Security** (3)

Theory and practice of software security. Identification of potential threats and vulnerabilities early in the design cycle. Methodologies and tools for identifying and eliminating security vulnerabilities. Techniques to prove the absence of vulnerabilities and ways to avoid security holes in new software. Essential guidelines for building secure software: how to design software with security in mind from the ground up and to integrate analysis and risk management throughout the software life cycle.

CERTIFICATE IN SOFTWARE ENGINEERING

Department Faculty

[Certificate in Software Engineering Faculty Website](#)

Department Website

[Certificate in Software Engineering Website](#)