GEORGE MASON UNIVERSITY | The Volgenau School of Engineering

APPLIED COMPUTER SCIENCE, B.S.

Concentration in Software Engineering

2017-18

The Bachelor of Science degree in Applied Computer Science (BS ACS) has been created for students who want and need the knowledge and expertise of computer science to work in one of the many disciplines that require advanced computing techniques. These fields do not merely "use" computing but create new and interesting problems for computer scientists. One such field is Software Engineering.

The objectives of the BS ACS concentration in Software Engineering are to provide students with the following:

1. Fundamental knowledge regarding theory, methods and applications of Computer Science.
2. Foundational knowledge in engineering principles as applied to producing high quality software.
3. An understanding of how to integrate Computer Science and Software Engineering to produce software that is usable, reliable, maintainable, secure, scalable and efficient.
4. Preparation for employment as a software engineer in the software industry.
5. Preparation for graduate studies in fields such as Software Engineering and Computer Science.

Application Area

Software Engineering is one of the largest global industries today. Jobs are plentiful and salaries are high. Whereas in past decades, the success of software was due to efficiency, algorithms and time-to-market; 21st century software must be usable, reliable, maintainable, secure, scalable and efficient. Creating high quality software requires teams of people with highly developed, diverse, skills and knowledge of cutting-edge technologies. This program is ideal for students who want careers designing, building, and evaluating high quality software products, either as part of a unified team or in leadership roles.

Therefore, this program emphasizes formal education in software usability, applied object-oriented theory, software requirements and design, software testing and software maintenance. Since workplace communication and interaction are crucial to successful software projects, coursework in technical writing and organizational interactions are included. A major component of this program is a term of practical training in the form of an internship at a software company to cement the connection between academic lessons and real-world challenges.

Many industries desperately need teamwork-oriented engineering knowledge for software. These industries include web application companies, aerospace, financial, government, military, and energy.

Degree Requirements

The BS ACS in Software Engineering program can be successfully completed within the normal 120 semester hours at GMU. In addition to Mason Core requirements, including humanities, and social science, the BS ACS Software Engineering concentration requires foundation, core, and elective courses as described in this brochure.

These course requirements provide the student with expertise in programming, computer systems, software requirements and modeling, formal methods and analysis of algorithms. At least 45 semester hours of the degree requirements must be at the 300 level or above.

ACS Foundation Courses: CS 110, 112, 211, MATH 113, 114, 125, 203
ACS Core:
CS 262, 310, 321, 330, 367, 471, 483
One CS course numbered above 400

All BS ACS majors must complete at least 37 additional credits to meet the course requirements of the Software Engineering concentration. These credits will include CS 306 (Synthesis of Ethics and Law for the Computing Professional) and STAT 344 (Probability and Statistics for Engineers and Scientists I).

Software Engineering concentration

Course requirements provide students with expertise in usability, programming, design, testing, maintenance and cutting edge technologies.

Foundation: CS 306; STAT 344.

Core: SWE 205, 301/401, 332, 437

SWE Related: 15 hours chosen from
CS 450, CS 455, CS 463, CS 465, CS 468, CS 475, CS 491; SWE 432, SWE 443

Cross Disciplinary: ENGH 388 and one of
PSYC 333, COMM 320, COMM 335

GEORGE MASON UNIVERSITY | The Volgenau School of Engineering

Sample Schedule

FIRST SEMESTER (14 CREDITS)	
CS 110 Essentials of Computer Science	3
CS 112 Introduction to Programming	4
MATH 113 Analytical Geometry & Calculus	4
Mason Core (ENGH 101)	3

SECOND SEMESTER (16 CREDITS)	
CS 211 Object-Oriented Programming	3
MATH 114 Analytical Geometry & Calculus II	4
SWE 205 Software Usability Design & Analysis	3
COMM 100 Public Speaking	3
Mason Core	3

THIRD SEMESTER (14 CREDITS)	
CS 262 Low-Level Programming	3
CS 310 Data Structures	3
MATH 125 Discrete Mathematics	3
Natural Science Elective	4
Elective	1

FOURTH SEMESTER (16 CREDITS)	
CS 330 Formal Methods & Models	3
CS 367 Computer Systems and Programming	4
MATH 203 Linear Algebra	3
Mason Core	3
Natural Science Elective	3

FIFTH SEMESTER (15 CREDITS)	
CS/SWE 332 OO Software Design & Implementation	3
STAT 344 Prob/Stat for Engrs & Scientists	3
SWE Cross Disciplinary Elective	3
ENGH 302 Advanced Composition	3
Mason Core	3

SIXTH SEMESTER (15 CREDITS)	
CS/SWE 321Software Engineering	3
SWE 437 Software Testing and Maintenance	3
SWE 301 Internship Preparation	0
SWE Related elective	3
Mason Core	3
Mason Core	3

SEVENTH SEMESTER (15 CREDITS)	
SWE Related Elective	3
SWE Related Elective	3
CS 471 Operating Systems	3
ENGH 388 Professional / Technical Writing	3
SWE 401 Internship Reflection	1
Elective	2

EIGHTH SEMESTER (15 CREDITS)	
SWE Related Elective	3
SWE Related Elective	3
CS 306 Synth Ethics & Law for Computing Professional	3
CS 483 Analysis of Algorithms	3
CS Senior Elective	3

A Closer Look at SWE Courses

The following courses have been created specifically for this program.

SWE 205 Software Usability Analysis and Design (Spring)

Prerequisite: ENGH 101. Principles of user interface design. Concepts for objectively and quantitatively assessing the usability of software user interfaces. Outcomes include knowledge of quantitative engineering principles for designing usable software interfaces and an understanding that usability is more important than efficiency for almost all modern software projects, and often the primary factor that leads to product success. Major topics include cognitive models for human perceptions and needs, which are used as a basis for analytical and critical thinking about user interfaces; specific engineering principles for designing usable menus, forms, command languages, web sites, graphical user interfaces and web-based user interfaces. Assessments will include written analytical evaluations of existing user interfaces, exams, and HTML-based design projects.

SWE 301 Internship Preparation
SWE 401 Internship Reflection

Preparation for, then reflection on, the Internship Educational Experience.

SWE 332 Object-Oriented Software Design & Implementation

Prerequisite: C or better in CS 310. In-depth study of software design and implementation using a modern, object-oriented language with support for graphical user interfaces and complex data structures. Topics covered are specifications, design patterns, and abstraction techniques, including typing, access control, inheritance, and polymorphism. Students will learn the proper engineering use of techniques such as information hiding, classes, objects, inheritance, exception handling, event-based systems, and concurrency.

SWE 432 Design and Implementation of Software for the Web (Fall)

*Prerequisite: C or better in MATH 125 and CS 321. T*eaches how to develop software for web applications. Covers client-server computing, theories of usable graphical user interfaces, and models for web-based information retrieval and processing. Goals are to understand how to design usable software interfaces and implement them on web, learn how to build software that accepts information from users across web and returns data to user, and understand how to interact with database engines to store and retrieve information. Specific topics are HTML, CGI programming, Java, Java applets, Javascripts, and Java servlets.

SWE 437 Software Testing and Maintenance (Spring)

Prerequisite: C or better in MATH 125 and CS 310. Concepts and techniques for testing and modifying software in evolving environments. Topics include software testing at the unit, module, subsystem, and system levels; developer testing; automatic and manual techniques for generating test data; testing concurrent and distributed software; designing and implementing software to increase maintainability and reuse; evaluating software for change; and validating software changes.

SWE 443 Software Architectures (Spring)

Prerequisite: C or better in CS/SWE 321. Teaches how to design, understand, and evaluate software systems at an architectural level of abstraction. By end of course, students will be able to recognize major architectural styles in existing software systems, describe a system's architecture accurately, generate architectural alternatives to address a problem and choose from among them, design a medium-size software system that satisfies a specification of requirements, use existing tools to expedite software design, and evaluate the suitability of a given architecture in meeting a set of system requirements.