



APPLIED COMPUTER SCIENCE, B.S.

Concentration in Software Engineering

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 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 General Education (GE) requirements, including humanities, and social science, the BS ACS Software Engineering concentration requires foundation, core, and elective courses as described in this brochure.

Course requirements provide students with expertise in usability, programming, design, testing, maintenance and cutting edge technologies. At least 45 semester hours of the degree requirements must be at the 300 level or above.

ACS Foundation Courses: CS 101, 105, 112, 211, MATH 113, 114, 125, 203

ACS Core:

ECE 301, CS 262, 310, 330, 367, 421, 465, 483

One CS course numbered above 400

Software Engineering concentration

Foundation: STAT 250, CS 306

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

SWE Related: 15 hours chosen from

CS 363, CS 455, CS 450, CS 468, CS 471, CS 475, SWE 432, SWE 443,

Cross Disciplinary: ENGL 410 and one of PSYC 333, COMM 320, COMM 335

Sample Schedule

FIRST SEMESTER (16 CREDITS)

CS 101 Preview of Computer Science	2
CS 112 Introduction to Programming	4
MATH 113 Analytical Geometry & Calculus	4
ENGL 101 Composition	3
Western Civilization Elective	3



APPLIED COMPUTER SCIENCE, B.S.

SECOND SEMESTER (14 CREDITS)

CS 211 Object-Oriented Programming	3
MATH 114 Analytical Geometry & Calculus II	4
CS 105 Computer Ethics and Society	1
COMM 100 Public Speaking	3
Literature Elective	3

THIRD SEMESTER (14 CREDITS)

CS 262 Low-Level Programming	1
CS 310 Data Structures	3
ECE 301 Digital Electronics	3
MATH 125 Discrete Mathematics	3
Natural Science Elective	4

FOURTH SEMESTER (15 CREDITS)

CS 330 Formal Methods & Models	3
CS 367 Computer Systems and Programming	3
SWE 205 Software Usability Design & Analysis	3
MATH 203 Linear Algebra	3
Natural Science Elective	3

FIFTH SEMESTER (15 CREDITS)

CS/SWE 332 OO Software Design & Implementation	3
CS/SWE 421 Software Requirements & Design Modeling	3
SWE Cross Disciplinary Elective	3
ENGL 302 Advanced Composition	3
Social and Behavioral Science Elective	3

SIXTH SEMESTER (15 CREDITS)

SWE 437 Software Testing and Maintenance	3
STAT 250 Introductory Statistics I	3
SWE 301 Internship Preparation	0
SWE Related elective	3
Fine Arts Elective	3
Global Understanding Elective	3

SEVENTH SEMESTER (16 CREDITS)

SWE Related Elective	3
SWE Related Elective	3
CS 465 Computer Systems Architecture	3
ENGL 410 Professional / Technical Writing	3
SWE 401 Internship Reflection	1
Elective	3

EIGHTH SEMESTER (15 CREDITS)

SWE Related Elective	3
SWE Related Elective	3
CS 483 Analysis of Algorithms	3
CS 306 Synth Ethics & Law for Computing Professional	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

Explores concepts for objectively and quantitatively assessing the usability of software user interfaces.

SWE 301 Internship Preparation

SWE 401 Internship Reflection

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

SWE 332 Object-Oriented Software Design & Implementation

In-depth study of software design and implementation using a modern, object-oriented language with support for graphical user interfaces and complex data structures.

SWE 432 Design and Implementation of Software for the Web

This course teaches students how to develop software for web applications.

SWE 437 Software Testing and Maintenance

Concepts and techniques for testing and modifying software in evolving environments.

SWE 443 Software Architectures

This course teaches how to design, understand, and evaluate software systems at an architectural level of abstraction.