Computer Science Assessment Data

This section of the report deals with student educational outcomes and provides evidence in the forms of assessment measures and feedback from key stakeholders. In addition to these key measures, this section discusses innovative teaching practices within the department as well as other factors in student learning.

Assessment Measures of the CS Degree Program

MFAT Test Scores

The ETS® Major Field Achievement Test (MFAT) provides the best evidence for the educational outcomes of the computer science department students. The MFAT is a test developed and administered by the Educational Testing Service (ETS). According to ETS, the MFAT test is "a comprehensive undergraduate outcomes assessments designed to measure the critical knowledge and understanding obtained by students in a major field of study."[16]

The computer science department has administered this test to our junior and senior students every year since 1988. With the exception of 1994, the results of this assessment have been above the national norm as evidence by Figure 3 below.

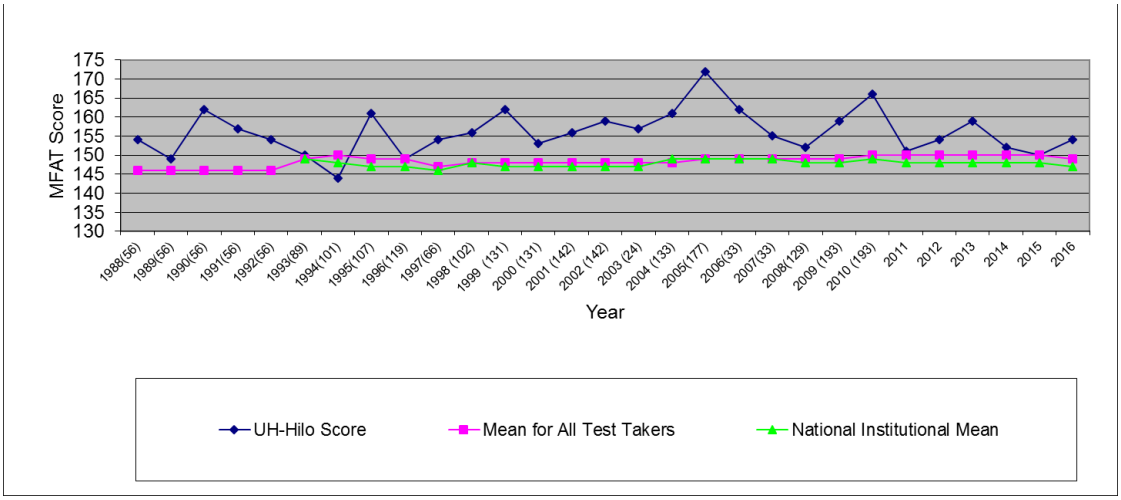

Figure 3 – Major Field Achievement Test Results by Year

This activity of ongoing assessment promotes a culture of educational effectiveness and student success within the department. The computer science department is pleased with its students and their long-term performance on this important assessment measure.

The results from 2011-2016 show that UH-Hilo placed above the 50th percentile each year. The stabilizing trend toward the mean is likely due to the department having a more consistent number of upper division students for the past six years. While graduation numbers have held steady at a rate of 9 per year over the past two program review periods, the standard deviation for the number of graduates has decreased from 5 to 2.5 meaning that a more consistent number of students are taking the test each year.

[16] Educational Testing Service. About Major Field Achievement Tests. https://www.ets.org/mft/about

Learning Outcomes and Core CS Objectives

Matrix of Program Outcomes and Courses

Degree or Program Name: Computer Science (BS)

Program / Department Chair: Dr. H. Keith Edwards hedwards@hawaii.edu

Revision Date: February 19, 2017

Outcome 1: Understand classical algorithmic processes and data structures and be able to perform simple analysis of algorithms

Outcome 2: Be proficient in one high-level programming language and have basic skill levels in a variety of programming languages

Outcome 3: Understand the basics of logic design and computer organization and be aware of multiple architecture approaches and numerical limitations

Outcome 4: Be competent in techniques of discrete mathematics and understand the theoretical foundations of computing

Outcome 5: Understand the steps of the software development process and the activities/products appropriate to each

Outcome 6: Know the major issues in the design and implementation of major computing artifacts such as operating systems, programming languages, graphical user interfaces or systems programming, and databases, networks, or compilers

Outcome 7: Be able to adapt to changing development platforms and design/implementation tools

Outcome 8: Communicate effectively on technical matters in both oral and written forms, work well within a team, and understand the social/ethical issues of computing

Courses for Majors	Require Elective	Outcome 1	Outcome 2	Outcome 3	Outcome 4	Outcome 5	Outcome 6	Outcome 7	Outcome 8
MATH 205	**R**	**P**			**P**				
MATH 206	**R**				**P**				
MATH 311	**R**			**P**					
PHYS 170 -170L	**R**			**P**					
PHYS 171 -171L	**R**			**P**					
CS 141 – Discrete Mathematics for Computer Science 1	**R**	**I**		**I**	**I**				**I**
CS 150 - Introduction to Computer Science	**R**	**I**	**I**			**I**		**I**	
CS 151 - Introduction to Software Development	**R**	**D**	**D**			**D**			
CS 241 – Discrete mathematics for CS II	**R**				**D**				

Courses for Majors	Require Elective	Outcome 1	Outcome 2	Outcome 3	Outcome 4	Outcome 5	Outcome 6	Outcome 7	Outcome 8
CS 266 - Computer Organization and Assembly Language	**R**			**D**				**I**	
CS 321 - Data Structures	**R**	**M**	**M**						**D**
CS 340 – GUI	**One R**						**D**	**D**	
CS 350 – Systems Programming			**D**				**D**	**D**	
CS 407 – Introduction to Numerical Anaylsis I	**R**		**D**	**M**				**D**	
CS 410 - Elements of Computer Architecture	**R**			**M**				**D**	**D**
CS 420 - File Management	**R**						**M**	**D**	
CS 430 - Operating Systems	**R**		**D**				**M**	**M**	
CS 450 - Organization of Programming Languages	**R**		**M**				**M**	**M**	
CS 460 - Software Engineering I	**R**		**M**			**M**		**M**	**M**
CS 461 - Software Engineering II	**R**		**M**			**M**		**M**	**M**
CS 470 - Theory of Computing	**R**				**M**				
CS 495 - CS Professional Seminar	**R**								**M**
CS 421 – Database Management System Design	**Two R**						**M**	**M**	
CS 431 – Computer Networks / Data Communication							**M**	**M**	
CS 451 – Compiler Theory			**D**				**M**	**M**	

I = Introduced, D = Developed & Practiced with Feedback, M = Demonstrated at the Mastery Level appropriate for graduation in this program, P = Prerequisite to Success in Later Courses

Assessment Plan for Program Outcomes

Degree or Program Name: Computer Science
Program / Department Chair: Dr. H. Keith Edwards hedwards@hawaii.edu
Revision Date: February 19, 2017

Program Outcome (same outcomes as Deliverable #1)	Assessment Plan	Status / Progress / Results
1. Understand classical algorithmic processes and data structures and be able to perform simple analysis of algorithms	Successful completion of projects and exams in CS 321. Above national average in MFAT sub-area **Discrete Structures and Algorithms**	
2. Be proficient in one high-level programming language and have basic skill levels in a variety of programming languages	Successful completion of projects and exams in CS 321 and CS upper division courses (except CS 470 and 495). Above national average in MFAT sub-area **Programming Fundamentals**	
3. Understand the basics of logic design and computer organization and be aware of multiple architecture approaches and numerical limitations	Successful completion of projects and exams in CS 266, CS 410, and CS 407. Above national average in MFAT sub-area **Systems (Architecture, Operating Systems, Networking, Database)**	
4. Be competent in techniques of discrete mathematics and understand the theoretical foundations of computing	Successful completion of assignments and exams in CS 141, 142, and 470. Above national average in MFAT sub-area **Discrete Structures and Algorithms**	
5. Understand the steps of the software development process and the activities/products appropriate to each	Contribute to production of a successful software engineering project in CS 460-461	
6. Know the major issues in the design and implementation of major computing artifacts such as operating systems, programming languages, graphical user interfaces or systems programming, and databases, networks, or compilers	Successful completion of projects and exams in CS 430, 450,340 or 350, and 421, 431, or 451. Above national average in MFAT sub-area **Systems (Architecture, Operating Systems, Networking, Database)**	
7. Be able to adapt to changing development platforms and design/implementation tools	Successful completion of CS upper division courses (except CS 470 and 495).	
8. Communicate effectively on technical matters in both oral and written forms, work well within a team, and understand the social/ethical issues of computing	Write and present meaningful reports in CS 141, 321, 450 and 460-461. Make meaningful contributions to group work in CS 141, CS 321, and CS 460-461. Successful completion of assignments in CS 495	

Note: All these assessment mechanisms are in place. The MFAT has been given for 20 years at UH-Hilo.

Appendix 4 – Student Outcomes Assessment

Required Courses

CS150 – Introduction to Computer Science I

Must Know:

- Basic control structures - decision and looping
- Concepts of parameter passing by value and by reference
- Architecture of a computer program
- Construction and use of simple classes

Should Know:

- The scope of the computer science field; a breadth-first level knowledge of algorithms, computer organization, system software, programming languages, and theory

Must Be Able To:

- Design, implement, debug, and test simple object-oriented programs in a modern high-level language using an IDE

CS151 - Introduction to Computer Science II

Must Know:

- Class definition and implementation including container and derived classes
- Linear structures - array, string, list, stack and queue
- Sorting and searching with linear structures
- Pointers and dynamic variables
- Implementation of a linked list
- Data abstraction - function and class templates
- Recursive thinking

Should Know:

- How to create a basic graphical user interface.

Must Be Able To:

- Create classes and objects to solve problems
- Select the appropriate linear structures to solve a variety of problems
- Explain the advantages and disadvantages of different implementations of linear structures, i.e., static arrays, dynamic arrays, or linked lists
- Write programs in an object-oriented language such as C++

CS141/241 – Discrete Mathematics I & II

Must Know:

- Propositional and predicate logic
- Proof techniques: A) Direct proof B) Proof by contraposition C) Proof by contradiction
- Elementary combinatorics
- Relation and function properties

Should Know:

- Graph terminology
- Elementary group theory
- Operation of finite-state machines
- Concept of order of magnitude

Must Be Able To:

- Solve problems using strong and weak induction
- Solve linear recurrence relations
- Perform set manipulations
- Perform Boolean algebra manipulations and minimizations

CS266 - Computer Organization and Assembly Language

Must Know:

- The computer system: A) Buses B) Internal Memory C) Input/Output D) Interrupts
- The central processing unit and the control unit
- Computer arithmetic
- Various addressing modes of a computer
- The complex instruction set computer and reduced instruction set computer
    - A) Superscalar processors
    - B) Pipelining

Should Know:

- The evolution and performance of computers
- Microprogramming

Must Be Able To:

- Program in assembly language

CS321 – Data Structures and Algorithms

Must Know:

- Techniques of algorithm analysis, including A) Recurrence relations B) Decision trees
- Tree structures (Binary search trees and AVL Trees)
- Search algorithms and their efficiencies
- Sort algorithms and their efficiencies
- Hashing
- Graph algorithms
    - Minimal path
    - Spanning tree

Should Know:

- Priority queues
- Huffman codes
- Idea of NP/NP complete

Must Be Able To:

- Implement classical algorithms
- Set up testbeds for empirical efficiency results

CS407 – Numerical Analysis

Must Know:

- Floating Point Representation (IEEE 754)
- Gaussian Elimination
- Techniques for Solving Non-Linear Equations
    - Bisection Method
    - Newton's Method
    - Secant Method
- Polynomial Interpolation
- Numerical Differentiation and Integration
- Techniques for Spline Construction
- Solving Initial Value Problems

Should Know:

- Least Squares Method
- Monte Carlo Methods and Simulation
- Linear Programming

Must Be Able To:

- Implement classical from the field
- Analyze the results for errors

CS410 – Architecture

Must Know:

- Combinational logic
  - Two-level design
  - Decoders and encoders
  - Multiplexers
  - ROM
  - Programmable logic arrays and array logic
- Fundamental sequential devices (latches and Flip-Flops)

Should Know:

- Sequential logic circuits
  - Counters
  - Modulo-N counters
  - Shift registers as counters
- Microprocessor operations
  - Input/Output
  - Interrupts
- Parallel processing, Parallel random access machine (PRAM) model, and Parallel Algorithms

Must Be Able To:

- Simulate logical circuits.
- Analyze and synthesize synchronous sequential circuits
  - Mealy model
  - Moore model

CS420 – Introduction to Database Management Systems

- The significant differences between:
  A) stream files / no structure
  B) sequential files / <crlf> at end of each record
  C) relative files / array on the disk - possibly a control record
  D) indexed files / key position, length is metadata
  E) database as file access / full metadata
- The major relational database concepts
  A) table, tuple, attribute (file, record, field)
  B) metadata tables
  C) integrity (data, entity, referential)
- The basics of SQL
  A) embedded SQL (in C, C++, Java)
  B) as a tool in a database environment such as Access
- XML as a data transport and storage mechanism
- Major database indexing algorithms including B and B+ trees

Must Be Able To:

- Use embedded SQL as a file access method from within a host procedural language
- Use interactive SQL to develop simple queries to a relational database
- Implement XML-based data storage and retrieval algorithms

CS430 – Operating Systems

Must Know:

- Computer system architecture
- Basic features of an operating system
- Process scheduling
- Resource allocation
- Deadlock
- Concurrent programming
- File management
- Distributed systems

Must Be Able To:

- Explain the differences among various process scheduling methods
- Recognize whether mutual exclusion is possible among competing processes
- Explain the differences between processes and threads
- Recognize the necessary conditions that lead to a deadlock
- Simulate context switching and process scheduling such as round robin or shortest job first
- Recognize the problems and opportunities associated with distributed systems.

CS450 – Programming Languages

Must Know:

- The major features of languages representing the major programming paradigms
  - A) procedural languages (e.g., C, Ada, Algol, ...)
  - B) functional languages (Scheme, Lisp, ...)
  - C) logic programming languages (PROLOG, ...)
  - D) object-oriented languages (C++, Ada 95, Java, Smalltalk, ...)
- The major syntactic / semantic constructions used in modern programming languages
  - A) block structure, scope, overloading / dynamic allocation
  - B) primitive types, variables
  - C) data, control, and procedural abstraction
  - D) classes, objects / message passing, dynamic binding
  - E) generics / templates
  - F) recursion

Must Be Able To:

- Construct and execute programs in languages representing the major paradigms

CS460/461 – Software Engineering I and II

Must Know:

- The software development life cycle and its various models
- Requirements definition and analysis
- Software systems definition
- Software design process
- Software implementation process
- Quality assurance
- Validation and verification
    - Testing
    - Program Inspections and Reviews
- Configuration management
- Software Project management
- Software documentation

Must Be Able To:

- Explain and define the various aspects of the software development process
- Participate in the development of a team project by assuming one or more responsibilities in the development team
- Present both orally and in written form the development effort
- Help complete the team project on time and within budget

CS470 – Theory of Computing

Must Know:

- The Chomsky language hierarchy
- The hierarchy of computational models (Finite-state machines, Pushdown automata, Linear bounded automata Turing machines)
- The relation of language classes and machines as recognizers/acceptors
- The role of nondeterminism
- Church-Turing Thesis
- Undecidability

Should Know:

- Normal forms for context-free grammars
- Closure properties for language classes

Must Be Able To:

- Use productions to generate words in a language
- Demonstrate machine recognition of a language
- Use the Pumping Lemma for regular and context-free languages

CS495 – Professional Seminar

Must Know:

    - Obligations of a CS Professional (Continued education, Ethical standards)
- Professional Societies and Codes of Ethics
- Analysis of social implications of systems and the systems development process

Should Know:

- Elements of career planning