COMPUTER SCIENCE 5314
PROGRAMMING LANGUAGES
(ADP TITLE: PROGRAMMING LANGUAGES)

I. CATALOG DESCRIPTION

5314 PROGRAMMING LANGUAGES

In depth investigation of the principles of programming systems, not necessarily restricted to programming languages, from the point of view of both the user and implementer. Algorithms of implementation, syntax and semantic specification systems, block structures and scope, data abstraction and aggregates, exception handling, concurrency, and applicative/functional/data-flow languages.

Pre: 3304. (3H, 3C).

II. LEARNING OBJECTIVES

Having successfully completed this course, the student will be able to

- analyze the advantages and disadvantages presented by a given programming language with respect to a particular application;
- read and write language syntax definitions in Backus-Naur Form (BNF);
- extend or modify a BNF description to include new language features;
- explain the implementation concepts behind types, variables, and subprograms;
- discuss the strengths and weaknesses of the major programming language paradigms;
- write simple programs in a functional or logic programming language
- critically evaluate programming language research;
- prepare and present a summary and critical evaluation of a published programming language research paper.

III. JUSTIFICATION

Programming Languages provide tools for expressing computations that are machine readable. This course is fundamental to a graduate level education in Computer Science because 1) it provides a basic understanding of how one designs a system to express computations from a general perspective as well as techniques for formulating non-conventional computations, and 2) provides an in depth study of language constructs found in the “classical” programming languages. This course complements a course related to the design and implementation of translators (CS 5304).

The prerequisite has been changed from 4105, which no longer exists, to 3304 since 3304 provides background in formal languages and grammars that are used to describe language constructs in 5314. The required texts have been updated to reflect current research practices and methods in computer science. Accordingly, the syllabus has been updated to include (i) new areas among traditional CS 5314 subtopics, and (ii) new and emerging paradigms in programming languages.

IV. PREREQUISITES AND COREQUISITES

This course is intended for first year graduate students who have completed an undergraduate program in Computer Science. It is assumed that the participants have an extensive working knowledge of at least two high level programming languages, an assembly language, and several paradigms. CS 3304 is a prerequisite because it provides background in formal languages and grammars, which are used to describe language constructs in 5314. Moreover, the hierarchy introduced in formal languages provide a measurement used for expressing the power of a given language.

V. TEXTS AND SPECIAL TEACHING AIDS

Required text to be chosen from:

Sebasta, Robert W. CONCEPTS OF PROGRAMMING LANGUAGES, 4TH ED. Reading, Massachusetts: Addison Wesley, 1999. xv, 670.

Louden, K. PROGRAMMING LANGUAGES: PRINCIPLES AND PRACTICE. Boston, Massachusetts: PWS Publishing Co, 1993. vii, 641.

Current articles selected from ACM SIGPLAN Notices will also be used.

VI. SYLLABUS

	Percent of Course
1. The evolution of programming languages	5
2. Review of syntax and semantic specification systems	10
3. Basic language elements name - value systems, expressions, statements, data types, aggregates, structures, blocks, scoping, and paradigms	15
4. Building blocks and Abstraction procedures, functions, modules, data abstraction and abstract data types, information hiding, inheritance, classes	15
5. Input/Output handling	5
6. Advanced Language Elements-Parallelism exception handling, concurrency, tasking, interprocess communication	20

7. Functional and Applicative Languages	15
8. Logic Programming Languages	15
	100

VII. OLD (CURRENT) SYLLABUS

	Percent of Course
1. The evolution of programming languages	5
2. Review of syntax and semantic specification systems	10
3. Basic language elements variables, expressions, statements, types, aggregates, structures, blocks, scoping, and paradigms	15
4. Building blocks-Abstraction procedures, functions, modules, data abstraction and abstract data types, inheritance, classes	15
5. Input/Output handling	5
6. Advanced Language Elements-parallelism exception handling, concurrency, tasking, interprocess communication	20
7. Special techniques and languages pattern matching (SNOBOL) functional and applicative languages (FP and LISP, resp.) data flow languages (Lucid) object oriented languages (Smalltalk)	30
	100

VIII. CORE CURRICULUM GUIDELINES

NA