End of Course Memo
CS 415 – Programming Languages
Aaron Bloomfield (Fall 2005)

Course Objectives:

1. Develop a greater understanding of the issues involved in programming language design and implementation
2. Develop an in-depth understanding of functional, logic, and object-oriented programming paradigms
3. Implement several programs in languages other than the one emphasized in the core curriculum (Java/C++)
4. Understand design/implementation issues involved with variable allocation and binding, control flow, types, subroutines, parameter passing
5. Develop an understanding of the compilation process

Assessment of Learning by Course-Objective:

Objective 1: [Develop a greater understanding of the issues involved in programming language design and implementation]
The lectures discussed these topics. Thus, the students learned this objective through the lectures, and were assessed on this on the midterm and final exam, as well as the final project report.

Objective 2: [Develop an in-depth understanding of functional, logic, and object-oriented programming paradigms]
Three of the four homeworks were based on these three programming paradigms; the first homework was in Fortran. The second homework used OCaml (functional), the third Prolog (logic), and the fourth Smalltalk (OOP).

Objective 3: [Implement several programs in languages other than the one emphasized in the core curriculum (Java/C++)]
There were five languages for which the students had to develop a program. The first four were the four listed in objective 2. The last program was their final project, which they chose the language. The languages chosen were Ada 95, Delphi, Euphoria, PHP, Pascal, PostScript, Python, and Ruby.

Objective 4: [Understand design/implementation issues involved with variable allocation and binding, control flow, types, subroutines, parameter passing]
The lectures discussed these topics. Thus, the students learned this objective through the lectures, and were assessed on this on the midterm and final exam, as well as the final project report.

Objective 5: [Develop an understanding of the compilation process]

The overall structure of the lectures generally followed the stages of a compiler. Thus, the students learned this objective through the lectures, and were assessed on this on the midterm and final exam.

Assessment of Changes Made in the Course:

As this was my first semester teaching this course, I did not make many significant changes to the course. The few changes I did make are listed below.

- Coverage of aspect oriented programming: this topic was introduced in two separate lectures, so as to give the students an understanding of what it is.
- Coverage of design patterns: this topic also was introduced throughout the end of the semester. It was not discussed in great detail, as it more properly belongs in another course (such as CS 494). But as the students had never heard of design patterns (!), I felt they should know what they are.
- Change of the functional programming language: in previous semesters it was Scheme; this semester, I used OCaml.
- Addition of a homework assignment, specifically the Fortran assignment
- Coverage of different languages: throughout the semester, I gave a few lectures on specific languages. Some were given in past iterations of the course (such as Fortran, Algol 60, Prolog, and Smalltalk), others were augmented (such as Perl), and others were new (such as C# and INTERCAL). This worked well to show how different languages solve the common programming language problems.

Other Issues:

1. Do you have concerns regarding the background of students coming into the course?

   The students are not very familiar with compilers. While this is arguably a different course, the only time a student really learns about compilers is in the PL course. I would support adding an undergraduate compilers course.

   Also, many students commented that they finally understood why computer theory (from CS 302) is useful, as I was able to show a number of applications in this course (regular expressions for a lexer, etc.). The theory courses should give a better background as to the applications of computer theory.

   Lastly, students should be exposed to design patterns in prior courses: perhaps CS 201, but definitely by the end of CS 216. It does not have to be a very in-depth exposure, but they should know what they are.

2. Are there other issues affecting student learning beyond what has been discussed elsewhere in this report? Include any other concerns you have about what students have or have not learned when they have completed the course.

None.

3. If you know of changes being made or considered in the curriculum that might affect the course, briefly describe what these are and how the course might be affected.

   Some have suggested making this course a required course – I would support that proposal.

4. List any other comments you think the Committee that monitors our degree programs should know about this course this semester.

   None, other than my comments above (especially for # 2).

Mapping of Course Objectives to BSCS Outcomes:

CS Degree Outcomes: Students who graduate with a BSCS will…	**Course Obj. 1**	**Course Obj. 2**	**Course Obj. 3**	**Course Obj. 4**	**Course Obj. 5**
(1: Math & DLD) Have demonstrated comprehension in relevant areas of mathematics (including calculus, discrete math, and probability), and in the area of logic design.					
(2: Fundamentals) Have demonstrated comprehension in fundamental topics of computing, including the intellectual core of computing, software design and development, algorithms, computer organization and architecture, and software systems.	X	X	X		X
(3: Analysis & Evaluation) Have applied knowledge of areas of computing to analyze and evaluate algorithms, designs, implementations, systems, or other computing artifacts or work-products. Application of this knowledge includes the ability to design, conduct and evaluate the results of experiments and testing activity.					
(4: Build Solutions) Have applied knowledge of areas of computing to create solutions to challenging problems, including specifying, designing, implementing and validating solutions for new problems.			X		
(5: Research Awareness) Be aware of current research activity in computing through activities including reading papers, hearing research presentations, and successfully planning and completing an individual research project in computing or its application.	X			X	
(6: Broadening) Have demonstrated comprehension of subjects in the humanities, social sciences, and the natural sciences in order to broaden a student's education beyond engineering and computing.					
(7: Social and Professional) Comprehend important social, ethical, and professional considerations related to computing practice and research, and be able to apply this knowledge when analyzing new situations.					
(8: Post-graduation) Be prepared to enter graduate programs in computing or related fields, and be prepared to begin a professional career in computing.	X	X	X	X	X
(9: Life-long Learning) Have demonstrated a self-directed ability to acquire new knowledge in computing, including the ability to learn about new ideas and advances, techniques, tools, and languages, and to use them effectively; and to be motivated to engage in life-long learning.			X		
(10: Teamwork) Have demonstrated the ability to work effectively in a development team.					
(11: Communication) Have demonstrated the ability to communicate effectively (orally and in writing) about technical issues.	X				
(12: Professional development practices) Comprehend important issues related to the development of computer-based systems in a professional context using a well-defined process to guide development.					