Computing & Software Systems 482:
Expert Systems
Fall 2010

Basic Course Information

CSS 482 will introduce you to a completely different way of programming, in which you specify rules of behavior, rather than algorithms. This is an especially powerful approach for problems that change often or where solutions involve application of human knowledge, rather than intricate calculations. Since their commercial introduction in the early 1980s, expert systems have undergone tremendous growth, representing one of the most successful application of artificial intelligence technology. Today, they are used in business, science, engineering, manufacturing, etc. Example applications include: business rules, customer support, computer configuration, fault diagnosis, computer-aided instruction, data interpretation, planning and prediction, and process control.

This course will have an additional focus on building expert systems applications as part of larger systems, including web-based and enterprise systems. Besides rule-based programming, expert systems operation, and knowledge engineering, topics will include some aspects of Java that are useful for developing these systems, such as JavaBeans, serialization, applets, servlets, J2EE, JavaServer Pages, Tomcat, web services, and XML.

Prerequisites: CSS 343. No prior Java experience is required, just a desire to learn about it. We will mostly be modifying example Java code; the bulk of the programming will be in JESS.

Lectures Mondays and Wednesday, 1:15–3:15PM, room UW1-031.

Instructor Michael Stiber stiber@u.washington.edu, room UW1-360D, phone 352-5280, office hours Mondays and Wednesdays 3:30-4:30PM.

Course Web http://courses.washington.edu/css482/.

Required Textbook Friedman-Hill, Ernest, *JESS in Action*, Manning, Greenwich, CT, 2003.

Suggested References Arnold, Ken, James Gosling, and David Holmes, *The Java Programming Language*, Fourth Edition, Prentice Hall, 2005.

Grading 30% homework + 30% midterm + 30% project + 10% participation (including project reports).

Assignments Assignments will be due at specific dates and times. I will *not* accept *any* lateness in this class — if your assignment is submitted late, it will *not* be graded, and you will receive a *zero* for that assignment. Except for special circumstances, such as medical and other emergencies, *no* exceptions will be made to this policy. You are more than welcome to submit work before the due date.

Your name, student number, and email address should be written on your hard copy submissions. Please strive to either write/draw clearly or use a computer and high-quality printer to prepare your documentation; I cannot give you credit for what I cannot read.

Special needs The University of Washington is committed to providing equal opportunity and reasonable accommodation in its services, programs, activities, education and employment for individuals with disabilities. If you believe that you have a disability and would like academic accommodations, please contact Disability Support Services at 425.352.5307, 425.352.5303 TDD, 425.352.3581 FAX, or at dss@uwb.edu. DSS will be happy to provide assistance. You will need to provide documentation of your disability as part of the review process.

Collaboration You are expected to do your work on your own (unless, of course, it is a group project). If you get stuck, you may discuss the problem with other students, provided that you don't copy from them. Assignments must be written up independently. You may always discuss any problem

with the instructor. You are expected to subscribe to the highest standards of honesty. Failure to do this constitutes plagiarism. Plagiarism includes copying assignments in part or in total, debugging computer programs for others, verbal dissemination of algorithms and results, or using solutions from other students, solution sets, other textbooks, etc. without *crediting these sources by name*. Any student guilty of plagiarism will be subject to disciplinary action.

Class attendance I strongly encourage you to come to class (and, in fact, a portion of your grade will depend on active participation, as well as project reports). You will be held responsible for *all* material covered in class, regardless of its presence (or lack thereof) in the textbook.

Class Ettiquette Please turn off cell phones during class. I have no problem with you using a laptop to take notes, but if you are using it for anything else, please sit at the back of the class to avoid distracting other students. Please use the restroom before class to avoid disturbing everyone as you enter and leave the room; *under no circumstances will I allow anyone to leave the room during an exam and return to finish it.*

Problems If you have problems with anything in the course, please come and see me during office hours, make an appointment to see me at some other time, or send email. I want to make you a success in this course. Deliverables represent hard deadlines; this is to prevent your schedule from slipping so much that you won't be able to complete the class. **I will *not* give out grades of "incomplete"** except in circumstances in which you are making good progress and an extraordinary, unexpected event has made it impossible for you to finish the quarter. A large amount of work in another class is *not* an extraordinary, unexpected event.

Programming Assignments

I assume that everyone is past the point where I need to carefully check the details of your program execution. In other words, I expect you to turn in a fully documented program, including reasonably commented and formatted source code, design documentation sufficient for me to understand how your program works, and execution logs sufficient to convince me that your program does indeed work (that it satisfies the requirements outlined in the assignment document). Please submit your assignment electronically to the CSS 482 drop box.

Projects and Project Proposals

The purpose of this project is to design and implement an expert system that performs some real and interesting function (small though it may be). For logistical reasons, I desire to limit this class to approximately five projects, therefore, depending on enrollment, you may need to form teams of three or four. Your project should be accompanied by *detailed* design documentation in the final report.

The project is worth 30% of your grade, and the amount of time you spend should reflect that. I would expect you to spend about 45–50 hours/person working on the project.

Proposals

On October 13, I will make appointments with each team for the following week to hear your project ideas and give you feedback regarding the scope and workload. You will have only 15 minutes, so *come prepared* to present your ideas and ask specific questions. This time is *not* meant to fish around for ideas from me: you should already have your own by then (you are encouraged to see me well in advance of this date to discuss and brainstorm ideas).

Project proposals are due on October 27; each team should hand in only one copy. You will receive a grade for your proposal, which will count towards your grade for the course as one-sixth of your project grade (5% of your overall grade). So, a team should have spent on the order of 7–8 hours/person on the

project *design. All proposals and reports must be typed or typeset.* It is acceptable to submit hand-drawn figures, but I encourage you to use a drawing program.

The proposal itself should be 3–5 pages of 10-point text. It is your responsibility to justify in this proposal that your project is worth doing and that you should receive a good grade in this course if you do it well. It should be specific enough about *what* you will do and *how* you will do it that it could be used as a contractual agreement with me regarding your project content. Components of your proposal should include (but are not limited to) any of the following that are relevant:

Title The title should be short and descriptive.

Authors Team member names.

Summary A brief description of what the project will do, suitable for someone unfamiliar with the application area.

Background An introduction to the application area, including how your project fits into the existing state of the art. You should have done a literature search and read some *real* (i.e., non-web) reference material before writing this.

Techniques Course concepts and algorithms covered by the project. This section is the core justification that successful completion of this project should result in a good grade for you.

System Operation Outline What is your idea of how the system will work (from the user's point of view)? Should include things like UI sketches.

Design A high-level design (at the level of a context diagram and top-level system block diagram). It is not necessary to have a detailed design complete yet, but enough of the design should be done that the reader can determine what the scope of the work will be.

Fallbacks Identify high-risk aspects of this project. Discuss possible alternative approaches (if something goes wrong).

Annotated Bibliography By proposal time, you should have at least three quality references. Each reference should be accompanied by an analysis of its source, including who publishes it, for what audience is it published, and the impact that papers published in that venue have on the field. A reference librarian should be able to help you with this (if there is a call for this, I will try to have someone come to class to talk about these matters). If you have identified a human expert for your project, then you should summarize their qualifications. If you are working on a research paper, then you should have at least five quality references, along with annotations summarizing the topics that each work covers.

Presentations/Demos

Here are suggestions for project presentations and demonstrations:

- Don't put too many words/sentences on a single slide. Five sentences is about the most that should appear.
- Don't make your slides the complete text of your talk. The visual aid(s) you use should serve to amplify what you have to say.
- Conversely, don't just read from your slides. The audience can do that (and you *don't* want them to just read; you want them to pay attention to you).
- Think of what you can put on the slides that adds value to what you have to say. Pictures are one example of this kind of additional information.

- Just as you shouldn't read your slides, don't just read your notes. You should be prepared enough that you don't need notes (beyond maybe some 3x5 cards to jog your memory, especially for the first slide or two). Reading from your notes indicates to the audience that you didn't think this was important enough to spend much preparation time on, but nevertheless you think they should spend their time listening to it. *Practice your talk.*
- Don't just paraphrase something you read elsewhere; *understand* what you read and convey that to your audience. Give the audience something more than they could have gotten for themselves. Show that you understand what you're talking about, so they will feel their time was well-spent listening to you.
- Put everything the audience needs to understand a slide on that one slide; don't flip back and forth between slides. For example, if you have a diagram and some necessary text, put them on the same slide. (But a diagram without the need for text is even better.)
- Don't just tail off at the end. Plan an end that has finality to it.
- Go through your demonstration beforehand. Make sure that it works flawlessly. Test it on exactly the machine that you will use for the demo. The may mean coming in a few days beforehand to try it out on the e-podium machine in our classroom. If you will be using your laptop, please let me know so that I can get a laptop cable *before* the demo day so that you can try your laptop out on the e-podium.

Project Reports

Your report should include the software you have developed. I would like to put all these projects on the course web site (if you'd rather that your project not be there, please let me know). To facilitate this, please set up your submission for placement on the web. At minimum, you should include an HTML `index.html` file, which lists all other files (with *relative* HTML links), provides a brief description of your project for someone unfamiliar with this class, and explains how to run your program. You should include your report in HTML or some other format. Please submit this to the CSS 482 drop box.

As a general guideline, your written report should be 10–20 pages long, and should be submitted by midnight on December 13. It should be typed or typeset in a 10–12 point font. Figures, code, etc. are *not* counted in this page limit, and should be numbered and included throughout the report. A group should produce a single report. Your report should be an expanded and completed version of your proposal, with the following changes and additions:

- The *Fallbacks* section is not necessary.
- The *System Operation Outline* should be replaced by a *User Guide*.
- The *Design* should be complete. It should be extensive enough for someone else to continue development of the project.
- You should add the following:

  Operation Examples Illustrate the sytem in operation with real execution examples. This could be part of the user guide. A screencast video would be appropriate here; please contact me for how to submit this, as it will almost certainly be too large to submit via Catalyst.

  Alternative Approaches Discusses other possible ways to implement this.

  Limitations What your project *doesn't* do, that it might have, given time and motivation.

  Problems Problems encountered and solutions applied (or, if not solved, possible avenues for solution).

  Future Work Suggested extensions to your program.

Cutting across these topics, your report should make it clear what you learned, relating to both the course material and any unexpected problems or issues that gave you new insight.

Tentative Course Schedule

Date	Topics	Reading	Assignment
9/29	Welcome; The what and why of expert systems; A case study;	Chapters 1–4	homework 1 assigned
10/4	Facts and rules; The anatomy of an expert system	Chapters 6, 7	homework 1 due; homework 2 assigned
10/6	Fact workshop		
10/11	JESS and Java; expert systems as components	Chapter 5	homework 2 due; homework 3 assigned
10/13	Rule workshop		sign up for project conferences
10/18	Computational complexity of expert systems	Chapter 8	homework 3 due; homework 4 assigned; project conferences this week
10/20	Knowledge engineering; the expert system development process	Chapters 9–1	
10/25	Design pattern: diagnosis and backward chaining	Chapters 12, 13	homework 4 due; homework 5 assigned
10/27	Diagnosis workshop		project proposals due
11/1	Java inside JESS: facts connected to the real world	Chapters 14–16	homework 5 due
11/3	Midterm		
11/8	JESS inside Java: embedding JESS in web applications	Chapters 17–19	homework 6 assigned
11/10	Web app workshop		
11/15	Intelligent agents: the Wumpus World		homework 6 due; homework 7a assigned
11/17	Agent workshop		
11/22	Wumpus World competition 1		homework 7a due; homework 7b assigned
11/24	Expert systems in context: an overview of AI		
11/29	Wumpus World competition 2		homework 7b due
12/1	Expert systems in context: knowledge representation, logic, and reasoning beyond logic		
12/6	Flex time		
12/8	Project presentations		
12/13	Project reports due		