NORTHERN ARIZONA UNIVERSITY
College of Engineering, Forestry & Natural Sciences
Department of Electrical Engineering and Computer Science

Course Syllabus

CS 477 – Advanced User Interfaces		Spring 2016
Credits: 3 credits	**Pre-reqs:** CS 386 with C or better	**Co-Reqs:** N/A
Section#: 1	**Co-convened with:** CS477 and CS577	**Cross-listed with:** N/A

Academic Catalog Description: Explores design and construction of modern graphical user interfaces, including event models, client-server interaction, and interface design and usability evaluation.

Course Purpose: This elective course provides in-depth coverage of the theory and practice of user interface design and evaluation. Upon successful completion of this course, you should have an understanding of the principles of user-centered requirements acquisition, design and construction of modern graphical user interfaces, and experience in applied usability analysis. This course is essential preparation for those looking to lead or participate on projects including significant graphical user interface elements. Specific learning goals include:

- Theoretical Foundations of Human-Computer Interaction, Principles of Design
- Applied interface design: models, guidelines, heuristics
- Expertise in graphical interface software programming
- Expertise in design and execution of effective usability testing regimes
- Design and effective use of a modern usability testing facility

Detailed Information for this offering

Time and Location: **11:30 – 12:20, MWF. Rm 224 Engineering**

Course Website: http://www.cefns.nau.edu/~edo/Classes/CS477_WWW/

Readings and Materials:

Course Textbook: Designing the User Interface, by Ben Shneiderman, Addison-Wesley

Recommended: Handbook of Usability Testing: How to Plan, Design, and Conduct Effective Tests, Jeff Rubin, Dana Chisnell, 2nd Edition, Get it on Amazon.

Instructor's Name: **Dr. Eck Doerry**
Office Building/Room Number: **Rm 259 Engineering**
Email: **Eck.Doerry@nau.edu**

Instructor Availability:

Office hours:	**MW 13:00 – 14:20** **TH 14:30 -- 16:00**
Other:	• Although you should try hard to make it to scheduled office hours, I am also available at other times by appointment. To schedule, see me before or after class or

	send email. • Email is appropriate for **short** questions; longer questions/discussions should be handled in person. • If my office door is open or cracked outside of office hours, feel free to knock - I may be available (but no guarantees). Talking to and helping students is what makes my job fun, so if you are having problems please find some way to come see me as soon as you can.

Course Structure and Evaluation Mechanisms:

General expectations: The course web site is where all the action is. In my classes, all homeworks, programming assignments, solutions, announcement, etc. appear on the class website. This means specifically:

- You will very rarely have anything handed to you in hard copy. If you need it, it's on the course website!
- Many important class announcements will come by email. I expect you to check your email at least once a day, preferably more often.

Evaluation Mechanisms: Designing excellent interfaces is essentially a hands-on process, requiring you to not only know the design theory and the process steps, but to be able to apply that theory in a wide variety of interface design contexts. In many ways, user interface design and development has strong similarities to architecture and even art; there are guidelines to follow, but the skill evolves from practice and reflective discussion. As such, this course will be driven primarily by an evolving series of projects. We will certainly have many days of lecture to lay down the theoretical foundations and describe relevant approaches and processes, but we will also have many "studio days", in which project teams report on what they have done, and we collectively evaluate and reflect on that progress. With this in mind, there are three mechanisms by which your course grade will be determined:

Projects: Three to four projects will be assigned to give you hands-on experience with all aspects of interface design and evaluation. All of them will involve a strong element of programming…but only as a necessary tool to realize the GUIs we are developing.

Exams: There will be one midterm during the semester, aimed at testing your knowledge of design approaches, models, and other background information from lecture. The final exam will consist of the final presentation of the large final class project.

Class participation: Interaction -- both with the instructor and with other students -- is crucial for understanding and integrating the ideas presented in lecture. To emphasize this, a significant number of points are set aside to reward students who take an active role in the class. Participation points are also used to credit quizzes (in-class or on BBlearn) and other formative exercises.

Grading System:

Weighting of Deliverables: The following percentages are used in weighting total points earned on programming, exams, and participation: • Projects, including related assignments = 70% • Midterm = 25% • Participation/Quizzes = 5%	**Grading Scale**: 90-100% = A 80-89% = B 70-79% = C 60-69% = D under 60% = F
Notes: • Simply completing what is required is enough to earn a "C". To get an "A" or a "B" you must show	

additional (i.e. above average or outstanding, respectively) analytic insight, clarity of presentation, and creativity. For more detail on what is expected for each grade level, refer to the ASEE's Guidelines for Engineering Grading and Written Presentation Evaluation Rubric linked on the course website.

Peer Evaluation and Grading: Most of our projects will be done as small teams. It is important for all team members to contribute reliably and definitively to the productivity of the team in order for the team to succeed. To help evaluate this participation, we will be using a peer evaluation approach to gain insight into team dynamics, and to fairly distribute project points based on effort invested. In general, the peer eval system will be used to calculate an "effort factor" for each team member, **which will serve to individually modify the team for all team deliverables for each member.** In this way, high-performing team members will be rewarded, slackers will be penalized. See the discussion of how peer evaluation works on the course website for details.

Policy on Graduate Student grading: Since this course is cross-listed as a graduate class, I will grade graduate students differently in several ways. Graduate students will generally be held to a higher standard than undergraduates, expected to produce higher quality, more complete work products to receive the same grade as undergraduates. Graduate students will also be held to higher expectations of quality on technical writing, with hypotheses and arguments supported with references to appropriate literature. Finally, graduate students will be assigned an overall survey paper on a topic in User Interfaces of their choice (must be approved by me), which will be due at the end of Reading Week.

Class Outline or Tentative Schedule:

See Tentative Course Schedule outlined on Class Website

Class Policies:

Attendance: Attendance is required. You are responsible for all material covered during the lectures whether you attend or not. If you must miss a class, be sure to get the notes from another student. Late arrivals are disruptive --- plan to arrive five minutes before the start of class.

Electronic Device usage: All cell phones, PDAs, music players and other electronic devices must be turned off (or in silent mode) during lecture, and may not be used at any time. Laptops are allowed for note-taking only during lectures; no surfing or other use is allowed. I devote 100% of my attention to providing a high quality lecture; please respect this by devoting 100% of your attention to listening and participating.

Late work and Make-ups: Unless otherwise noted, all assigned work is due at the beginning of class on the date they are due! Programming assignments will be submitted electronically as well as in hardcopy. The following specific policies apply:

- **Quizzes:** No make-ups, no late work accepted
- **Exams:** Make-ups only when scheduled/approved in advance or with proper documentation (note from physician, etc.), as required by NAU policy.
- **Written Homeworks:** No late work accepted, no make-ups.
- **Projects and Programming Work:** Late work accepted with 10% deduction per half-day late, within limits. See Late work and Make-up Policies on course website for details.

Grade Challenges: Although I try hard to grade as accurately and fairly as I can, mistakes do occur. If you feel that I owe you some points, or would like to discuss an evaluation, I encourage you to stop by office hours. To avoid loss of context, any grade disputes must be brought to my attention no later than 5 business days after the assignment was returned.

Homework Submission and Format: All homework must be submitted in hardcopy (no email submissions accepted!) unless specifically stated otherwise. Submissions should be clearly collated, with nicely annotated content, and stapled. For details on formatting and submission requirements. See Formatting Guidelines on the class website. For programming assignments, electronic submission of the code/application may also (in addition to packet) be required for grading. The official submission time for programming assignments is based on the timestamp generated by the electronic upload.

Academic Dishonesty: Cheating will not be tolerated and may result in serious sanctions, including immediate failure in the course. Serious incidents of academic dishonesty will also for brought to the attention of the university and may result in expulsion. Aside from group projects, all work in this class is meant to be an

individual effort by the person receiving the grade. Any variation from this is considered cheating and all parties involved (giving or receiving) will be sanctioned.

Additional clarification on cheating for coding assignments: Sometimes students are not clear on what is or is not cheating in regards to programming. The programs you turn in for grading should have been written and typed in by you without referencing another's work (you may refer to books, of course --- but you may not copy substantial sections of code from any book, the internet, or any other medium). It is often helpful to verbally discuss problems with others to clarify the problem to be solved and discuss possible solutions; this is acceptable and even encouraged. But you should never at any time give another student a copy of your code or accept code from another student, either physically or electronically. Some examples of cheating include but are not limited to:

- Turning in someone else's code (perhaps with minor modifications) as your own. This includes code that you found on the internet, reference texts, or elsewhere.
- Turning in fabricated output for a non-functional program (or editing output from a partially-functional program) in an effort to fool the grader into thinking the program works as specified.
- Using fragments (e.g. functions/methods) from another's code in your code, passing them off as you own.

Occasionally, it is acceptable to use a peripheral utility code written by someone else but you should always (a) clearly document the source of the code fragment and (b) check with me first.

Other Important Course Information:

Vital Skills and other Hints for success in CS396

Be active, not passive.

One of the skills you should develop is an ability to read difficult material on your own. Try to read at least lightly through the material in the assigned reading before the related lectures. If you see a reference to something you don't know (a term, a programming language, etc.), take initiative and go track it down: search the web, scan back through the book, or come talk to me.

Avoid hacking.

Writing a program can be a funny thing. Some people are very fast programmers, others are quite slow. One common complaint I hear is that someone spent XX hours on a program and still couldn't finish. Others might spend only a couple of hours. This disparity emphasizes the difference between programming and merely "hacking". It is important to make sure you understand what has to be done, the concepts associated with the assignment, and that you have a plan or outline for your program -- before you write a single line of code. All of this will help to decrease the time you spend in front of the computer wondering why something doesn't work.

Don't procrastinate.

Don't delay programming assignments until the last minute. Unlike writing a paper for an english class (which is done when you want it to be), a program is a living thing, full of errors that must be corrected before it is done. How long this takes is completely unpredictable. So be sure to leave yourself enough time --- time to think carefully about the design before you program, and time to resolve whatever problems do arise. Remember, whatever can go wrong will and at the least opportune time. My willingness to help you with your assignment outside of office hours greatly diminishes in the hours immediately before it is due.

University Policies:

This course is conducted in accordance with all applicable university policies which can be found at: http://nau.edu/OCLDAA/_Forms/UCC/SyllabusPolicyStmts2-2014/ including Safe Environment, Students with Disabilities, Academic Contact Hours, Academic Integrity, Research Integrity, Sensitive Course Materials, and Classroom Disruption Policy. The student handbooks are also valuable resources for other policies. The undergraduate student handbook is at http://nau.edu/student-life/student-handbook/.