Brooklyn College
Department of Computer & Information Sciences

CISC 7540 [*763X] Software Methodology

37½ hours plus conference and independent work; 3 credits

Techniques for the design, implementation, maintenance, and management of very large software systems. The relation between size and complexity. Goals and measurements. Design and implementation strategies. Testing, validation, and proofs of correctness. Language aspects. Design and implementation tools. Asynchronous and real-time systems. Project management.

Prerequisite: Computer and Information Science 717.1X and two courses chosen from among Computer and Information Science 758X, 759X and 765X.

This course will focus on the study of large software systems from a number of perspectives and sources. Software systems theory, history, and lessons learned from the relatively short history of large software systems will be presented

Techniques for the design. implementation, maintenance, testing, revision, and management of complex large software systems will be presented, discussed, and explored.

The relationships between size, complexity, reliability, and flexibility of software systems will be considered. Methods for software systems measurement, system design strategies, implementation, and goals, will also be the centers of focus.

We live in a world which is dominated by dependency on computers and the software which runs them; Software has been called "the Achilles Heel" of the computer industry. The dependencies which our world has slipped into are neither planned for nor necessarily desirable. The fundamental questions which the instructor would like answered:

1. Are today''s software systems safe?

In other words can we depend on the software systems which control transportation systems, medical systems, financial systems, communication systems. Would you let your life depend on these systems? How can we measure a software system's reliability?

1a. As the world becomes more dependent on software, does life become safer?

This is a necessary correlate to Question 1 - it questions whether the current trends toward large software systems should continue.

2. How do we measure the efficiency, safety and reliability of a piece of software?

3. How should successful software systems be tested, designed, developed, and maintained?

Required Textbooks:

1. The Mythical Man-Month (1995), Fred Brooks, Addison - Wesley

2. Software Runaways (1998), Robert Glass, Prentice Hall

3. Software Metrics (1995, or latest edition), Norman Fenton, International Thomson Computer Press.

Supplemental (Recommended) Texts

* 1. Normal Accidents (2nd ed. 1999), Charles Perrow, Princeton University Press).

* 2. Software Testing in the Real World (1995), Edward Kit, Addison-Wesley.

3. Safeware (1995), Nancy Leveson, Addison-Wesley.

4. Computer-Related Risks (1995), Peter Neumann, Addison-Wesley.

5. Software Failure: management failure (1996), Stephen Flowers.

STUDENT WORK

The basis of work for the course will be two projects, one an individual project **(15%)** and the other a group project **(20%)**.

The individual project will entail the study of a large software system from an analytical perspective. The goal of this analysis will be to apply the techniques for software measurement learned from the Fenton text (Software Metrics) to an existing software system. Here the key concepts will be ***measurement scales, attributes, products, processes, models, management, complexity and prediction.*** A 10-12 page research paper (10%) including full academic reference style will be expected coupled with a class presentation (5%) before **mid-term examinations (25%)**.

The Group Project will be worth 20% of the course grade involving similar analytical work to the individual project, but here the focus will be on Software Testing (the text by Edward Kit). It is expected that there will be small groups comprised of 3-5 students whose individual tasks will be to perform the various forms of testing discussed in the text. However in this instance each individual member of the group will be responsible for once aspect of the software testing process. There will be a final paper of about 20 pages for each Group **(15%)** and a Group Project Presentation **(5%)**. Each member of a group must make clear what aspect of the testing process he/she is responsible for. The software system being tested can either be an existing system, a beta-version of a system under development, or an existing software system or project which students would like to further develop and extend. It is not recommended that students develop a large software system from scratch during the course, although in certain circumstances this may be permissible. **Progress reports** on the group projects will be expected in October and November.

Late projects and homework assignments (below) will not be accepted. Final grades will be based on a curved scale (if necessary) but students must obtain a course percentage of at least 60% in order to pass the course (passing grades for graduate students start at ‘C’.

In addition to a **Midterm (25%)** and **Final Examination (30%)** there will also be small homework assignments (usually for discussion) and several occasional quizzes based on reading assignments. **Since this course is a graduate class which only meets once per week for two hours, attendance is considered compulsory. Absence from more than two class meetings would be sufficient reason for the instructor to withdraw a student from the course**.

<u>Summary of Course Grading</u>

Individual Project:	**15%**
Midterm:	**25%**
Group Project:	**20%**
Homework, Quizzes, Attendance:	**10%**
Final Examination:	**30%**