Quantitative Methods and Experimental Design in CS
A Required Course for a Ph.D. in CS

Executive Summary

Daniel A. Menasce
Department of Computer Science
George Mason University
The Volgenau School of IT & Engineering
Fairfax, VA, USA
menasce@cs.gmu.edu

1. MOTIVATION

Several years ago, our department created its Ph.D. in Computer Science program. We had several discussions about the structure of the program and requirements and examined other Ph.D. in CS programs in the country for guidance. In designing the program, we also recognized that Computer Science involves many different areas (e.g., artificial intelligence, software engineering, operating systems, databases, data mining, computer communication networks, computer graphics, distributed systems, and many others) which, like engineering, are aimed at building systems and/or algorithms to solve a given problem in an efficient manner. Most of the research work in these areas follows a common paradigm: design and *evaluate* a new system, method, or algorithm. We also noticed that most Computer Science curricula do not emphasize or teach techniques and methods for evaluating and comparing systems and methods.

It is also not uncommon to see published papers that display experimental results in an inadequate manner. For example, some may display average values of some performance metric obtained through experimentation or simulation without confidence intervals being computed and reported. More often than not, two systems are compared based on how they perform on average without any consideration to confidence intervals.

Based on these observation, we decided that our Ph.D. students should be equipped to design and conduct experiments that met sound scientific standards. That lead to the design of the only required course in our Ph.D. in CS program: CS 700 - Quantitative Methods and Experimental Design in CS.

2. DESIGNING THE COURSE

We had the following criteria in mind when designing the course:

1. *Example-driven.* The course should be example-driven with all examples taken from computer science situations. The approach to teach a technique should follow the steps of a) presenting a motivating example, b) introducing a technique to solve the problem, c) solving the problem at hand, and d) presenting similar problems for the students to solve.

2. *Hands-on.* The course should be taught in a way that provides students with hands-on experience in applying the techniques taught in class. To achieve this goal, the course is taught in a classroom in which each student has a computer connected to the Internet. The instructor has a console from which he/she can select any student's computer monitor to be displayed at all computer monitors and projected to the entire class. Using this environment, the instructor assigns a problem to be worked on in class using previously prepared input data made available at the course's web site. Then, the instructor randomly selects a student to show to the class how he/she solved the problem using the technique just learned. This leads to interesting and interactive discussions.

3. *Emphasis on Practice.* The course should emphasize the practical aspects of the methods presented and stress the assumptions under which they can be used rather than going into details about their theoretical underpinning. It would not be possible to teach a one-semester course covering such a wide range of topics and allowing for significant in-class problem solving while at the same time delving into the theory behind the topics.

4. *Relevance to Doctoral Research.* Students should see an immediate application of the techniques learned to their area of doctoral research. To that end, as part of the course, each student must develop a large-scale project that involves using the methods taught in the course to evaluate a system, method, or algorithm that they are dealing with in their doctoral research.

Projects are presented to the class in workshop style at the end of the semester. Students are encouraged to take this course during the early stages of their doctoral research as soon as they have identified a research topic.

The course as designed provides an integrated treatment to the models and practices of experimental computer science. Topics covered include scientific evaluation methods applied to computing, workload characterization, forecasting of performance and quality metrics of systems, uses of analytic and simulation models, design of experiments, interpretation and presentation of experimental results, hypothesis testing, and statistical analyses of data. More specifically, the course covers the following topics:

- Review of basic concepts in Probability and Statistics.
- Summarizing measured data.
- Computing confidence intervals for the mean, variance, proportion, and a future value.
- Comparing systems using sample data and confidence intervals.
- Hypothesis testing.
- Single and two-factor ANOVA.
- Simple linear regression models.
- Curvilinear regression and transformations.
- Design of experiments: factorial experiment design and one-factor experiments.
- Distribution fitting methods.
- Discrete event simulation
- Analysis of simulation results.
- Basic concepts in performance modeling.
- Basic single queuing systems: various M/G/1 results and G/G/1 approximations.
- Multiclass open queuing networks and single-class closed queuing networks.

3. EXPERIENCE WITH THE COURSE

I developed and taught this course for the first time in the Spring 2001. Since then, I have alternated in teaching it with my colleague Sanjeev Setia. The experience in teaching the course has been most rewarding. We have seen many of the course projects turn into conference papers of high quality due to the care and scientific approach to evaluation.

We have also seen an interesting variety of projects developed by students for the course. A small sample of topics follows:

- Evaluation of bi-directional routing in Chord
- Performance of a genetic programming based intensional query systems
- Analysis and comparison of four image compression techniques
- Recognizing hand gestures from silhouettes
- Quantitative analysis of battery charging algorithms for a real-time embedded system
- Statistical analysis of vector comparison functions
- The analysis of Hamming distance in genetic algorithms and FCBE algorithms
- Comparison of agent and machine learning methods a quantitative analysis
- A comparison of precision targeting techniques in image analysis
- A performance comparison of data access mechanisms for the migration from legacy to Web applications
- Performance Evaluation of the FC-EDF algorithm for real-time systems
- Analysis of flash mobs in BitTorrent

4. CONCLUDING REMARKS

The CS 700 course has instilled a culture of sound experimental Computer Science in our Ph.D. students. After taking this course, our students have a better understanding of a) why adequate experimental design is important in evaluating their new designs, methods, and techniques; b) how measurement data should be treated, summarized, and analyzed; c) existing analytic performance models; and d) how simulation should be used and how the results should be analyzed and reported.

We fill that the hands-on nature of the course and its relevance to the student's research area through the project help incorporate the tools and techniques presented in the course into the research arsenal of our doctoral students.